

УДК 004.738.5

ПЫТКОВ РОМАН ЕВГЕНЬЕВИЧ

«Мобильное приложение для защищённого
хранения пользовательских данных»

Квалификационная работа на степень

«БАКАЛАВР» по направлению 09.03.04

Южный федеральный университет,

ИКТИБ, кафедра МОП ЭВМ — 2026 г.,

769 с.

АННОТАЦИЯ

Пояснительная записка посвящена разработке мобильного приложения для защищённого хранения пользовательских данных (программный продукт Wallenc). Описаны анализ предметной области и аналогов, формирование требований, проектирование архитектуры и пользовательского интерфейса, программная реализация на Kotlin (Android, Jetpack Compose, Room, Hilt), тестирование и краткая экономическая оценка.

Реализованы иерархия VaultsManager → vault → storage → файлы (один LocalVault на устройстве, удалённые vault по OAuth), клиентское AES-шифрование, служебные метаданные в Room и проектный контур синхронизации без передачи ключей провайдеру. Исходный код размещён в приватном репозитории GitLab ЮФУ. Полный листинг исходных файлов — приложение А; программная документация — приложение Б.

UDC 004.738.5

PYTKOV ROMAN EVGENIEVICH

«Mobile application for secure storage of user data»

Qualification work for the degree of

BACHELOR in the direction of 09.03.04

Southern Federal University,

ICTIS, the Department of Mathematical
Support and Application of Computers (MOP

EVM) — 2026,

769 p.

ANNOTATION

The thesis is devoted to a mobile application for secure storage of user data (the Wallenc software product). Data is encrypted on the device before upload; decryption is performed only when the user enters a valid key. The work includes analysis of analogues, requirements, architecture and UI design, Kotlin implementation for Android, testing, and a brief economic assessment.

The application uses MVVM and Clean Architecture. Main features are local and remote vault management, client-side AES encryption, Room metadata storage, and Yandex OAuth integration.

Keywords: mobile application, client-side encryption, Android, vault, OAuth, Room.

СОДЕРЖАНИЕ

Введение	13
1 Анализ требований и предметной области	15
1.1 Анализ предметной области	15
1.2 Разработка требований к программному продукту	16
1.2.1 Назначение и цели создания системы	16
1.2.2 Функциональные требования	17
1.2.3 Нефункциональные требования	19
1.2.4 Требования к программно-аппаратной платформе	19
1.3 Аналоги	19
1.3.1 Аналог Google Files Secure Folder	19
1.3.2 Аналог Proton Pass / Proton Drive	19
1.3.3 Аналог Bitwarden	20
1.3.4 Аналог Cryptomator	20
1.3.5 Сводная оценка	20
1.4 Стек технологий	20
1.5 Обзор методов и подходов к защите данных в мобильных приложениях	21
1.5.1 Клиентское шифрование и модель zero-knowledge	21
1.5.2 Авторизация и взаимодействие с внешними провайдерами без собственного сервера	21
1.6 Обзор рынка и обоснование выбора решения Wallenc	21
1.7 Формирование технического задания	22
2 Проектирование архитектуры системы	23
2.1 Схема бизнес-процессов предметной области	23
2.1.1 Организационная структура	23
2.1.2 Карта процессов	23

2.1.3	Диаграмма BPMN	23
2.1.4	Карта систем	25
2.2	Диаграмма DFD	25
2.3	Объектно-ориентированный анализ и проектирование (UML)	27
2.3.1	Диаграмма прецедентов	27
2.3.2	Диаграмма классов	27
2.3.3	Доменная иерархия хранения данных	28
2.3.4	Диаграмма развёртывания	29
2.4	Проектирование локальной базы данных	30
2.5	Проектирование подсистемы синхронизации	31
2.6	Нефункциональные архитектурные решения	31
2.7	Вывод	32
3	Проектирование пользовательского интерфейса мобильного приложения	33
3.1	Подготовка проекта и аналитика	33
3.1.1	Анализ конкурентов и определение сильных и слабых сторон .	33
3.1.2	Ограничения и допущения проектирования UI	33
3.2	Исследования пользовательских сценариев	33
3.2.1	Потребности и барьеры пользователя	33
3.2.2	Полезные сценарии из аналогов	33
3.3	Проектирование пользовательских потоков	33
3.3.1	User Story	33
3.3.2	Customer Journey Map	34
3.3.3	Пользовательский сценарий	34
3.4	Проработка прототипа и особенности дизайна	36
3.5	Требования к эргономике и доступности	46
3.6	Вывод	46

4 Программная реализация	47
4.1 Разработка программных модулей	47
4.1.1 Модуль криптографической защиты данных	47
4.1.2 Модуль управления vault и шифрованием	47
4.1.3 Модуль адаптеров хранилищ	49
4.1.4 Модуль синхронизации хранилищ	49
4.1.5 Алгоритм согласования журналов синхронизации	49
4.1.6 Использование средств ИИ при разработке	52
4.2 Разработка мобильного приложения на Kotlin (Android)	53
4.2.1 Слой domain	53
4.2.2 Слой data	53
4.2.3 Слой presentation	54
4.3 Взаимодействие подсистем и итоговая архитектура	54
4.3.1 Модуль :vault-contracts	55
4.3.2 Модуль :domain-vault	55
4.3.3 Модуль :task-runtime	56
4.3.4 Модуль :infrastructure-android	56
4.3.5 Сборка и зависимости	56
4.4 Детализация реализации по модулям Gradle	56
4.4.1 Модуль :domain	56
4.4.2 Модуль :usecases	56
4.4.3 Модуль :domain-vault	57
4.4.4 Модуль :ui	57
4.4.5 Модуль :infrastructure-android	57
4.4.6 Модуль :task-runtime	57
4.4.7 Модуль :vault-contracts и :app	57
4.4.8 Журнал разработки и контроль версий	57

5	Тестирование программного обеспечения	59
5.1	План тестирования	59
5.1.1	Цели и задачи испытаний	59
5.1.2	Объект и уровни тестирования	59
5.1.3	Матрица тестовых сценариев	60
5.1.4	Критерии начала и окончания	61
5.1.5	Среда и инструменты	61
5.2	Модульные тесты (JUnit)	61
5.2.1	Криптография и доменные ошибки	64
5.2.2	Синхронизация, 2FA и use cases	65
5.2.3	Модуль :domain-vault	65
5.2.4	Модуль :ui	65
5.2.5	Модуль :task-runtime	66
5.3	Инструментальные тесты (androidTest)	66
5.4	Ручное и UI-тестирование	67
5.5	Отчёт о результатах тестирования	68
5.6	Вывод	69
6	Оценка результатов и экономические показатели	70
6.1	Обзор рынка программного обеспечения и аналогов	70
6.2	Оценка экономических затрат на разработку проекта	70
6.3	Модель применения и окупаемости	71
6.4	Вывод	71
7	Вариативная профессиональная компетенция (ВПК-2)	72
7.1	Связь компетенции с предметом ВКР	72
7.2	Прикладные задачи анализа данных и принимаемые решения	72
7.3	Модели, обучение и развёртывание	73
7.4	Этика, ограничения и вывод по ВПК-2	74

Заключение	76
Список использованных источников	77
Приложение А Листинги исходного кода проекта Wallenc	79
Приложение А.1 Система сборки	79
Приложение А.2 Модуль :app	105
Приложение А.3 Модуль :domain	186
Приложение А.4 Модуль :usecases	241
Приложение А.5 Модуль :ui	312
Приложение А.6 Модуль :domain-vault	577
Приложение А.7 Модуль :infrastructure-android	701
Приложение А.8 Модуль :vault-contracts	722
Приложение А.9 Модуль :task-runtime	731
Приложение А.10 Модуль :other	740
Приложение Б Программная документация	741
Приложение Б.1 Обзор программного продукта Wallenc	741
Приложение Б.2 Техническое задание (выдержка)	741
Приложение Б.3 Программа и методика испытаний	742
Приложение Б.4 Отчёт о результатах испытаний	742
Приложение Б.5 Руководство пользователя	742
Приложение Б.5.1 Установка	742
Приложение Б.5.2 Первый запуск и storage в локальном vault	742
Приложение Б.5.3 Шифрование storage	744
Приложение Б.5.4 Открытие и закрытие зашифрованного vault	746
Приложение Б.5.5 Переименование и удаление	748
Приложение Б.5.6 Удалённые vault и Яндекс	750
Приложение Б.5.7 Секреты и 2FA внутри storage	753
Приложение Б.5.8 Фоновые задачи	757

Приложение В Скриншоты пользовательского интерфейса	758
---	-----

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящем документе применяются следующие обозначения и сокращения:

Таблица 1 —

Обозначение	Расшифровка
API	Application Programming Interface — программный интерфейс приложения
DAO	Data Access Object — объект доступа к данным
E2E	End-to-end — сквозное шифрование
MVVM	Model — View — ViewModel
OAuth	протокол авторизации открытого типа
UI	User Interface — пользовательский интерфейс
CRUD	Create, Read, Update, Delete — создание, чтение, изменение, удаление
ПЗ	Пояснительная записка
ТЗ	Техническое задание
ВКР	Выпускная квалификационная работа
JUnit	Фреймворк модульного тестирования Java/Kotlin
IT	Instrumented Test — инструментальный тест Android

ВВЕДЕНИЕ

Современные пользователи хранят личные и рабочие данные в облачных сервисах и на съёмных носителях, однако инфраструктура провайдера не всегда может считаться доверенной. Утечки, компрометация учётных записей и юрисдикционные риски делают актуальным подход, при котором конфиденциальность обеспечивается на стороне клиента до размещения данных во внешнем хранилище [1,2].

Актуальность темы обусловлена распространением мобильных приложений для хранения файлов и секретов, а также ограниченностью готовых решений: многие продукты привязаны к собственному backend, закрытой экосистеме или узкой предметной области [3–5].

Цель работы — повысить конфиденциальность пользовательских данных при работе с недоверенными хранилищами за счёт разработки мобильного приложения для защищённого хранения пользовательских данных (Wallenc) без собственного сервера приложения, с иерархией vault → storage → файлы и клиентским шифрованием.

Для достижения цели были поставлены следующие **задачи**:

- а) выполнить анализ предметной области и сравнительный обзор аналогов, сформировать требования к программному продукту;
- б) спроектировать архитектуру системы, модель данных и пользовательские сценарии;
- в) реализовать программные модули приложения Wallenc на платформе Android (Kotlin);
- г) провести тестирование программного обеспечения и оформить программную документацию;
- д) выполнить оценку экономических затрат и перспектив применения результатов.

Объект исследования — методы и средства клиентской защиты данных в мобильных приложениях. **Предмет исследования** — проектные и программные решения приложения Wallenc.

Методы исследования: анализ нормативной и технической документации, сравнительный анализ программных аналогов, объектно-

ориентированное проектирование (UML, BPMN, DFD), прототипирование пользовательского интерфейса, программная реализация и тестирование [6,7].

Практическая база. Работа выполнена в рамках производственной практики в ООО НМФ «Нейротех» (09.02.2026–06.05.2026) по направлению 09.03.04 «Программная инженерия» на кафедре математического обеспечения и применения ЭВМ (МОП ЭВМ). Научный руководитель — Беликов А. Н. (кафедра системного анализа и телекоммуникаций, САИТ); руководитель от организации — Алексеев Д. М.

Научная новизна заключается в сочетании единого VaultsManager, модели vault/storage, клиентского шифрования и адаптерного доступа к провайдерам без собственного сервера приложения, с проектным контуром синхронизации зашифрованных данных без передачи ключей провайдеру.

Практическая значимость — использование результатов при дальнейшей разработке продукта и в учебных проектах по мобильной разработке и информационной безопасности.

Пояснительная записка состоит из введения, семи глав, заключения, списка использованных источников и трёх приложений (листинги исходного кода, программная документация, скриншоты интерфейса). В главе 1 обоснована актуальность и приведено сравнение аналогов; глава 2 описывает архитектуру и модель Room; глава 3 — UX и пользовательские сценарии; глава 4 — реализацию по модулям, алгоритм синхронизации журналов и применение ИИ-ассистента при разработке; глава 5 — тестирование; глава 6 — экономическую оценку; глава 7 — вариативную профессиональную компетенцию по анализу данных и решениям с использованием ИИ.

1 Анализ требований и предметной области

1.1 Анализ предметной области

В рамках анализа предметной области рассмотрены подходы к хранению чувствительных данных в мобильных приложениях и облачных хранилищах. Выделены требования: конфиденциальность при хранении и передаче; отсутствие необходимости доверять инфраструктуре хранилища; устойчивость к компрометации удалённого провайдера; разделение логики хранения и криптографической защиты; удобные сценарии создания хранилища, шифрования, открытия и работы с содержимым.

Сформирован вывод о приоритете клиентских криптографических механизмов, унифицированного доступа к разным типам хранилищ и архитектуры с чётким разделением слоёв.

Рассмотрены угрозы STRIDE в применении к клиентскому хранилищу: подмена файлов на диске провайдера (tampering) нейтрализуется шифрованием; утечка метаданных минимизируется за счёт шифрования имён путей (опция `encryptPath`); отказ в обслуживании облака не блокирует локальный доступ при наличии копии vault. Модель доверия: пользователь доверяет только собственному устройству и корректности реализации криптомодуля; облако и сеть считаются активными противниками для ciphertext.

Контекст взаимодействия участников показан на рисунке 1.

Контекстная диаграмма Wallenc

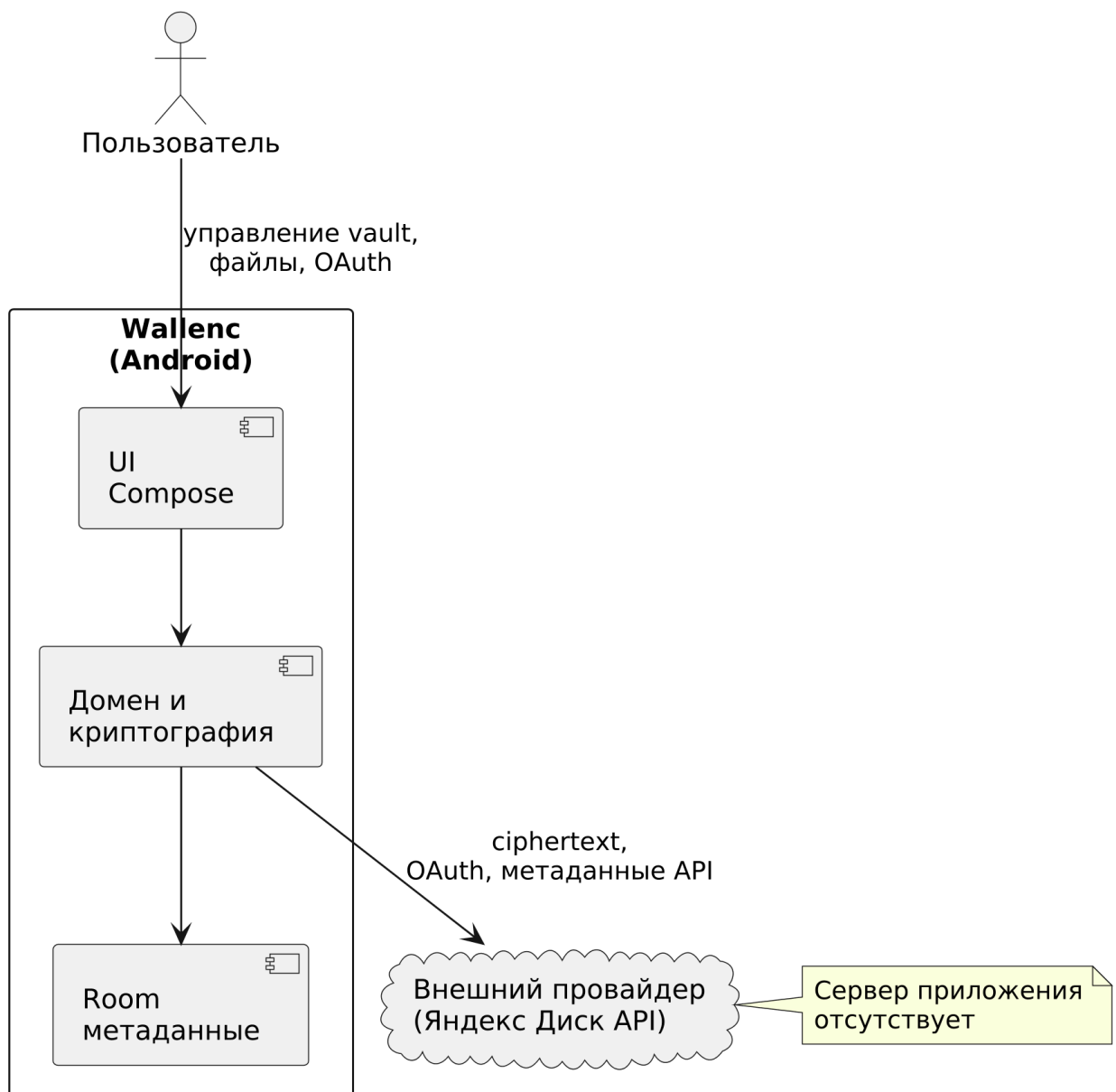


Рисунок 1 — Контекстная диаграмма: пользователь, приложение Wallenc и внешний провайдер

1.2 Разработка требований к программному продукту

1.2.1 Назначение и цели создания системы

Назначение системы Wallenc — предоставление пользователю мобильного клиента для работы с иерархией **vault** → **storage** → **файлы**: один локальный vault на устройстве, удалённые vault по учётным записям провайдера; внутри каждого vault пользователь управляет отдельными storage с обязательным клиентским шифрованием до выгрузки во внешнее хранилище.

Цели создания: обеспечить единую модель vault и storage; минимизировать доверие к провайдеру; поддержать расширение списка провайдеров через адаптеры; сохранить служебные метаданные в локальной БД без хранения пользовательского контента в открытом виде.

1.2.2 Функциональные требования

Функциональные требования сведены в таблице 2.

Таблица 2 — Свод функциональных требований

Код	Требование
ФР-1	Создание, просмотр, переименование и удаление storage в локальном vault (LocalVault — один на устройстве)
ФР-2	Включение шифрования storage, проверка ключа, открытие и закрытие зашифрованного представления
ФР-3	Просмотр и операции с файлами внутри storage; текстовые секреты и 2FA
ФР-4	OAuth-авторизация (Яндекс), регистрация удалённых vault и листинг их storage
ФР-5	Синхронизация: группы хранилищ, журнал коммитов, фоновый Worker без передачи ключей
ФР-6	Очередь фоновых задач: шифрование, синхронизация, отображение прогресса

1.2.2.1 Управление storage в локальном vault

Пользователь создаёт storage, просматривает список, переименовывает и удаляет их в единственном LocalVault. Служебные каталоги и системные пути не отображаются в пользовательском представлении.

1.2.2.2 Шифрование и открытие storage

При включении шифрования формируются параметры StorageEncryptionInfo; открытие выполняется только после успешной проверки ключа. Повторное шифрование одного storage до завершения предыдущей операции блокируется.

1.2.2.3 Работа с содержимым storage

Операции чтения и записи выполняются через единый интерфейс файлового доступа независимо от типа хранилища. Внутри открытого storage доступны текстовые секреты и генерация TOTP для 2FA (рис. 20, 21).

1.2.2.4 Удалённые хранилища и авторизация во внешних провайдерах

Реализован поток OAuth 2.0 для Яндекса [8,9] (рис. 2).

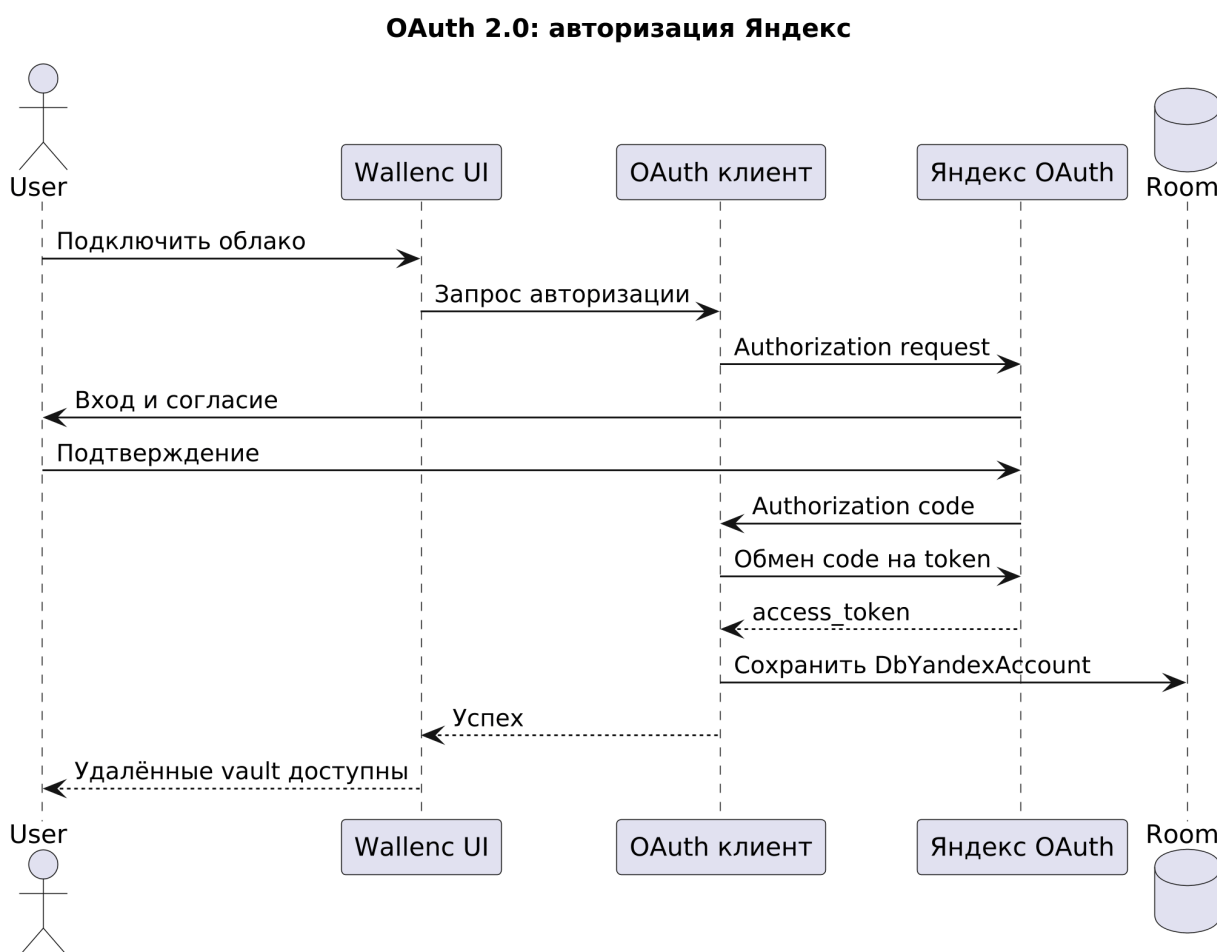


Рисунок 2 — Диаграмма последовательности OAuth Яндекс

1.2.2.5 Синхронизация зашифрованных данных

Спроектирован механизм фоновой синхронизации: в Room хранятся записи с UUID хранилищ; по таймеру запускается сервис, сравнивающий

истории коммитов локального и удалённого представления без передачи ключей на сервер провайдера (рис. 11–13).

1.2.3 Нефункциональные требования

К системе предъявляются требования по производительности (асинхронные операции на Coroutines), безопасности (AES на клиенте, минимизация утечек через имена путей), расширяемости (модульная структура Gradle) и устойчивости к гонкам при длительных операциях шифрования.

1.2.4 Требования к программно-аппаратной платформе

Минимальная платформа — Android с поддержкой Jetpack Compose; для OAuth и удалённых операций требуется сетевое подключение. Объём оперативной памяти должен быть достаточен для фоновых задач шифрования и Room.

1.3 Аналоги

1.3.1 Аналог Google Files Secure Folder

Google Files Secure Folder локально прячет и защищает файлы в отдельной папке по PIN или графическому ключу внутри Android [3]. Пользователь получает понятный «сейф» без настройки облака, однако сценарий ограничен устройством: отсутствует полноценная кроссплатформенная синхронизация зашифрованного vault между провайдерами, а модель хранилища не универсальна для подключения внешних API.

1.3.2 Аналог Proton Pass / Proton Drive

Экосистема Proton обеспечивает end-to-end шифрование для паролей, заметок и файлов с облачной синхронизацией. Для Wallenc релевантен опыт прозрачной для пользователя защиты и синхронизации, однако продукт жёстко привязан к инфраструктуре Proton, часть функций и объёмов зависит от тарифа, а роль «универсального клиента» к произвольному внешнему хранилищу ограничена.

1.3.3 Аналог Bitwarden

Менеджер секретов с шифрованием и синхронизацией. Ограничения: ориентация на учётные данные, а не файловый vault [4].

1.3.4 Аналог Cryptomator

Клиентское шифрование файловых vault в облаке (zero-knowledge). Ограничения: фокус на файлах, ограничения интеграций по платформам [5].

1.3.5 Сводная оценка

Таблица 3 — Сравнительная оценка аналогов

Критерий	Secure Folder	Proton	Bitwarden	Cryptomator	Wallenc
Собственный backend приложения	—	+	+/-	—	—
E2E / клиентское шифрование	+/-	+	+	+	+
Файловый vault	+	+	—	+	+
OAuth внешнего провайдера	—	+/-	+/-	+/-	+
Переносимость провайдеров	—	—	+/-	+	+
Unit-тесты без сервера	н/д	н/д	+/-	+/-	a) (68)

Обзор подтверждает актуальность концепции Wallenc: безопасность без собственного сервера и переносимая архитектура хранилищ.

1.4 Стек технологий

Для реализации выбраны Kotlin, Android SDK, Jetpack Compose, Coroutines/Flow, Hilt, Room, AES на клиенте; модульная

структура: :app, :domain, :usecases, :ui, :domain-vault, :infrastructure-android, :vault-contracts, :task-runtime [7,10–13].

Kotlin обеспечивает выразительную доменную модель и безопасность типов. **Jetpack Compose** декларативно описывает UI и состояние экранов vault. **Coroutines/Flow** используются для асинхронного шифрования, обращения к DAO и отображения прогресса без блокировки главного потока. **Hilt** связывает реализации интерфейсов domain с Android-инфраструктурой. **Room** персистентно хранит метаданные. Криптографические операции выполняются в доменном слое с опорой на стандартные API Java/Android и рекомендации NIST по AES [1].

Выбор стека согласован с целями практики: все компоненты поддерживаются Google и сообществом, документированы на русском и английском языках, применимы в промышленной разработке мобильных приложений.

1.5 Обзор методов и подходов к защите данных в мобильных приложениях

1.5.1 Клиентское шифрование и модель zero-knowledge

Данные шифруются до отправки во внешнее хранилище; провайдер не получает ключ расшифрования. Используется AES [1]; проверка ключа выполняется через `Encryptor.checkKey` без расшифровки всего содержимого.

1.5.2 Авторизация и взаимодействие с внешними провайдерами без собственного сервера

Применяется OAuth 2.0: токены доступа хранятся локально в Room (`DbYandexAccount`); ключи шифрования не передаются на сторону провайдера [8].

1.6 Обзор рынка и обоснование выбора решения Wallenc

Рынок мобильных хранилищ демонстрирует поляризацию: закрытые экосистемы с удобным UX (Proton, Google) и открытые криптоклиенты с ручной настройкой (Cryptomator). Wallenc занимает промежуточную нишу —

мобильный zero-knowledge vault с OAuth к популярному провайдеру (Яндекс) и расширяемой регистрацией типов хранилищ. Для корпоративного заказчика (Нейротех) важны: отсутствие затрат на сервер приложения, возможность аудита кода, формализованное тестирование (68 unit-тестов, см. гл. 5).

Перспективы коммерциализации связаны не с продажей облака, а с лицензированием клиента или внутренним внедрением. Барьер входа — ответственность пользователя за ключ; в ПЗ это отражено в руководстве пользователя (прил. Б) и предупреждениях в UI.

1.7 Формирование технического задания

Структура ТЗ оформлена по ГОСТ 7.32–2017: цели, этапы, функциональные и нефункциональные требования, порядок приёмки. Приоритет отдан ядру хранения и шифрования; расширения (2FA, текстовые секреты, синхронизация) зафиксированы как дополнительные функции второго этапа практики. Полный текст ТЗ — в приложении Б.

2 Проектирование архитектуры системы

2.1 Схема бизнес-процессов предметной области

2.1.1 Организационная структура

Участники процесса: **пользователь, мобильное приложение Wallenc, внешний провайдер хранения (облачный API)**. Сервер приложения отсутствует.

2.1.2 Карта процессов

Основные процессы: создание storage в vault → опциональное шифрование → открытие → работа с файлами → закрытие; для удалённых vault — OAuth → привязка учётной записи → листинг storage провайдера → проектная синхронизация.

2.1.3 Диаграмма BPMN

На рисунке 3 представлена диаграмма BPMN основного процесса работы с vault.

БPMN: жизненный цикл vault

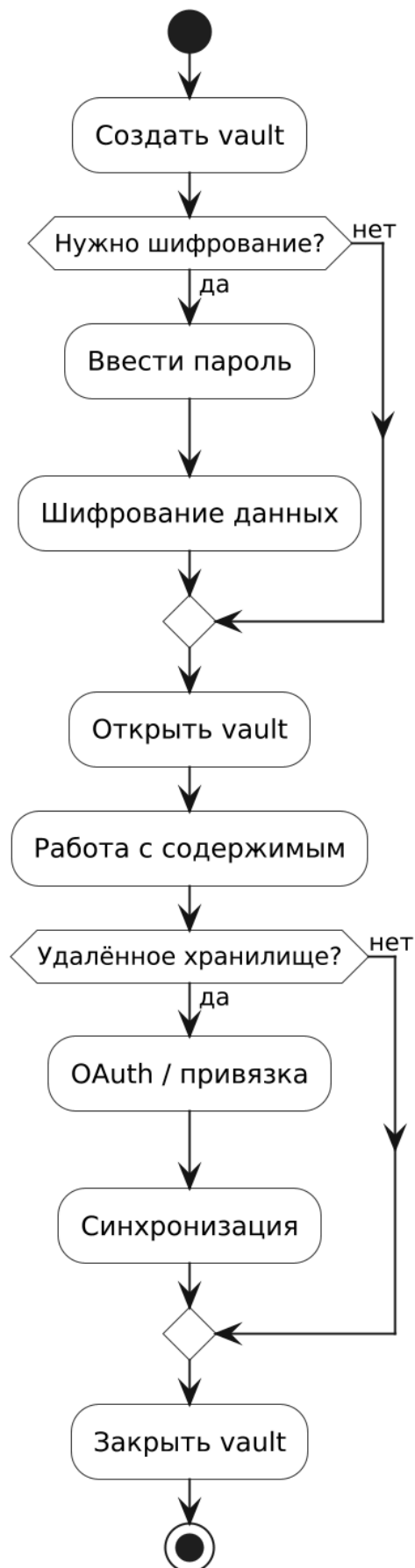


Рисунок 3 — BPMN: создание vault, шифрование, открытие, работа с
содержимым

2.1.4 Карта систем

Wallenc выступает посредником между пользователем и файловыми API провайдера, дополняя взаимодействие локальной БД Room и криптографическим модулем.

2.2 Диаграмма DFD

DFD уровня 0 (рис. 4) отражает потоки между UI, доменной логикой, криптографией, адаптерами хранилищ, Room и внешним API.

DFD уровень 0: Wallenc

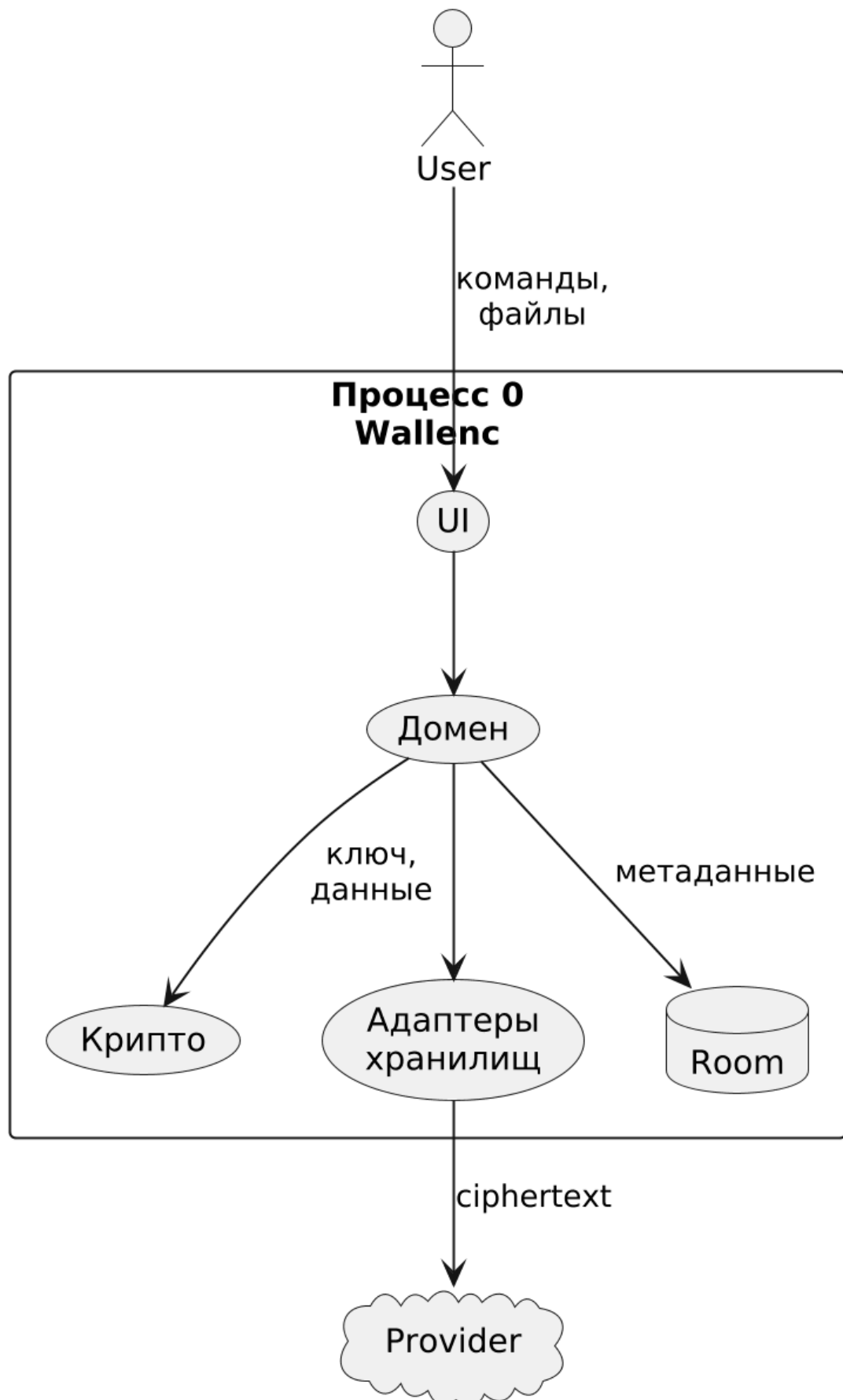


Рисунок 4 — DFD уровень 0: потоки данных в Wallenc

2.3 Объектно-ориентированный анализ и проектирование (UML)

2.3.1 Диаграмма прецедентов

Прецеденты включают управление vault, шифрование, работу с содержимым, OAuth и проектную синхронизацию (рис. 5).

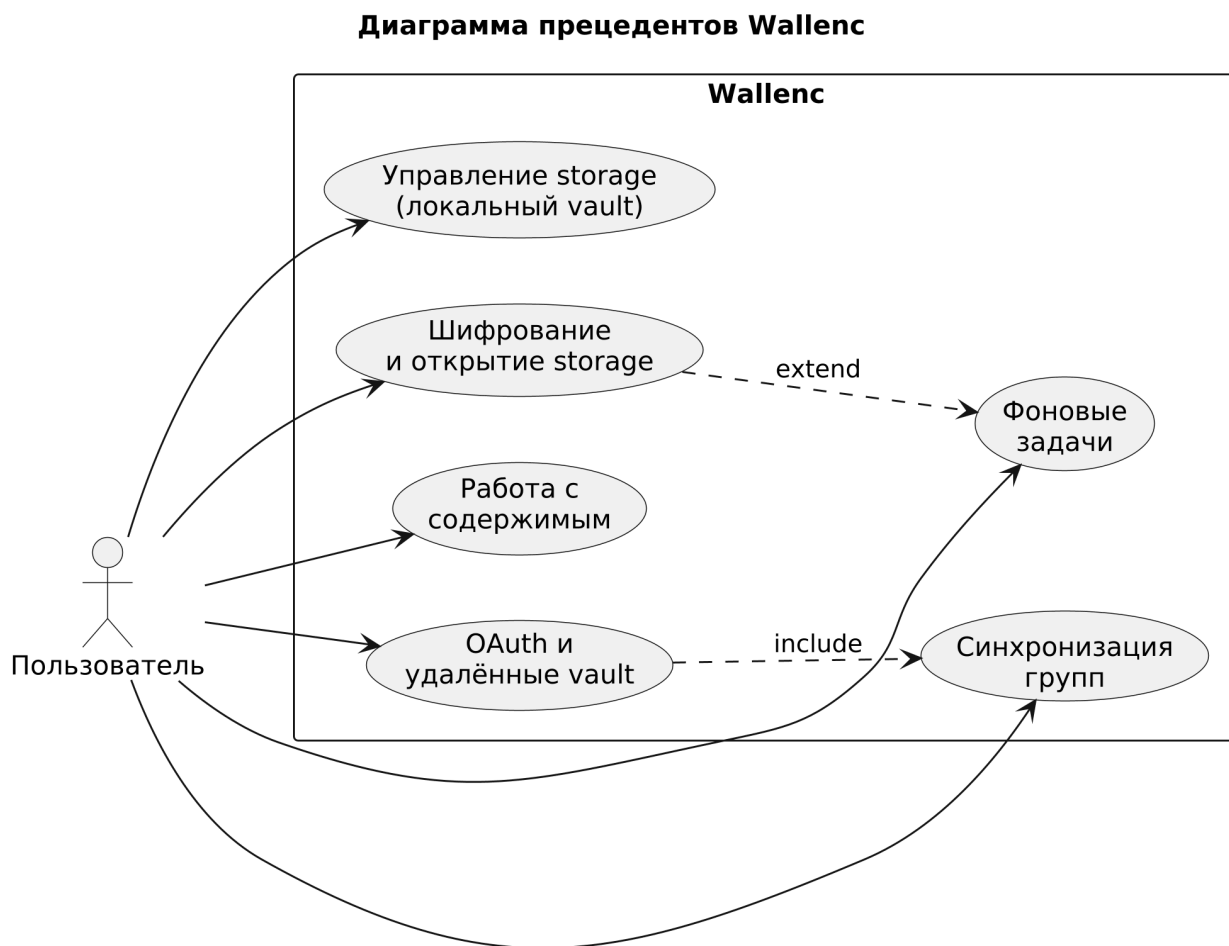


Рисунок 5 — Диаграмма прецедентов Wallenc

2.3.2 Диаграмма классов

Доменная модель модуля :domain (интерфейсы хранилищ, use case, сущности шифрования) приведена на рисунке 6.

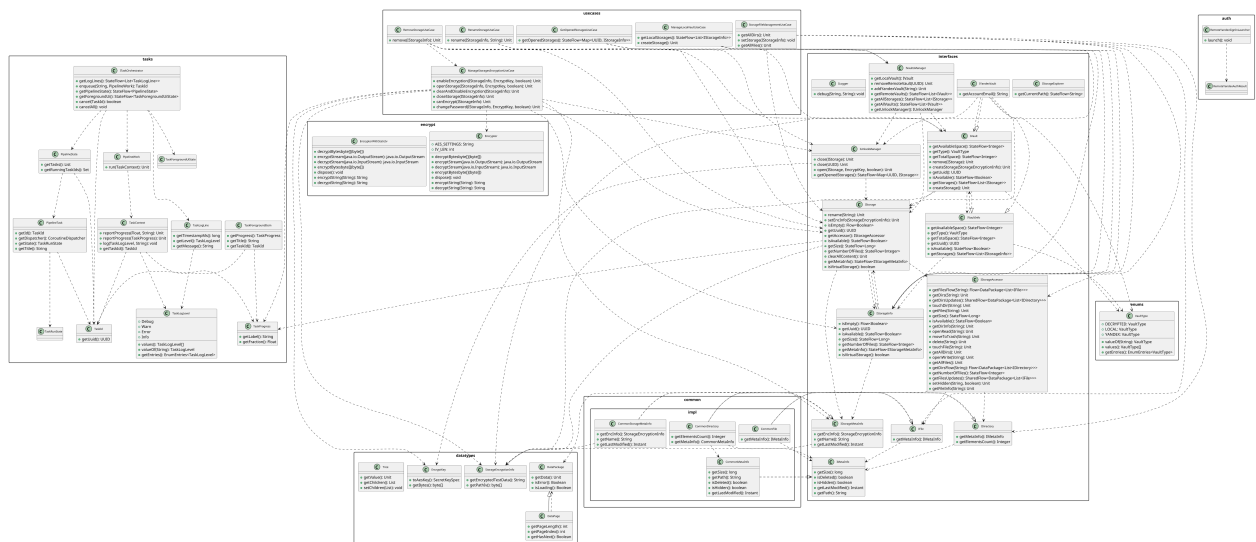


Рисунок 6 — Диаграмма классов модуля domain

2.3.3 Доменная иерархия хранения данных

Модель данных приложения строится на трёх уровнях (см. IVaultsManager, IVault, IStorage на рис. 6):

- а) **VaultsManager** — единая точка доступа: реактивный список всех vault (vaults), агрегированный список storage (allStorages) и IUnlockManager для открытых storage;
- б) **IVault** — контейнер, объединяющий набор storage. **LocalVault** в приложении один (корень на устройстве); удалённые vault (например, YandexVault) добавляются по одному на привязанную учётную запись OAuth;
- в) **IStorage** — хранилище, которое пользователь создаёт, переименовывает и удаляет в интерфейсе; внутри — файлы и каталоги, доступные через IStorageAccessor (с опциональным клиентским шифрованием).

Пользовательские сценарии «создать локальный vault» в UI соответствуют созданию нового **storage** внутри единственного LocalVault, а не появлению второго локального vault.

Служебные сущности Room показаны на рисунке 7.

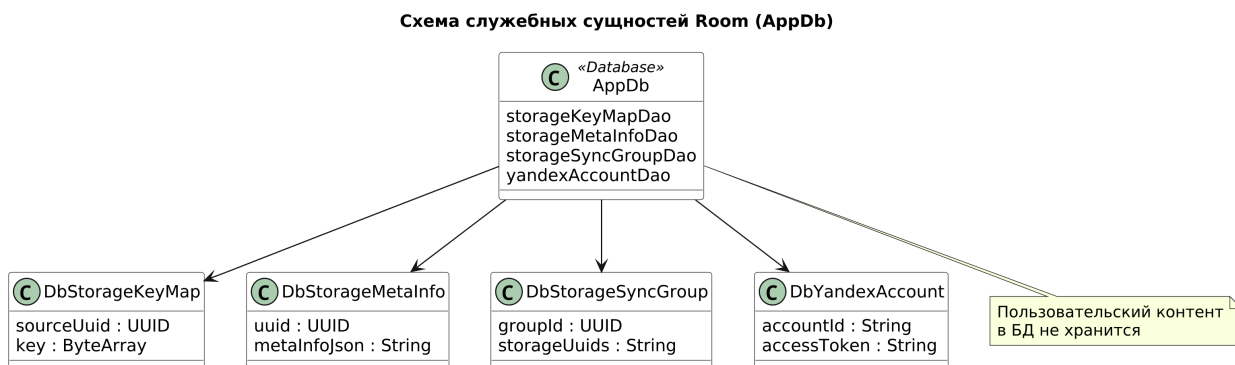


Рисунок 7 — Схема служебных сущностей Room

2.3.4 Диаграмма развёртывания

Компоненты развёртывания: устройство Android (приложение, Room), облачный API Яндекса (рис. 8).

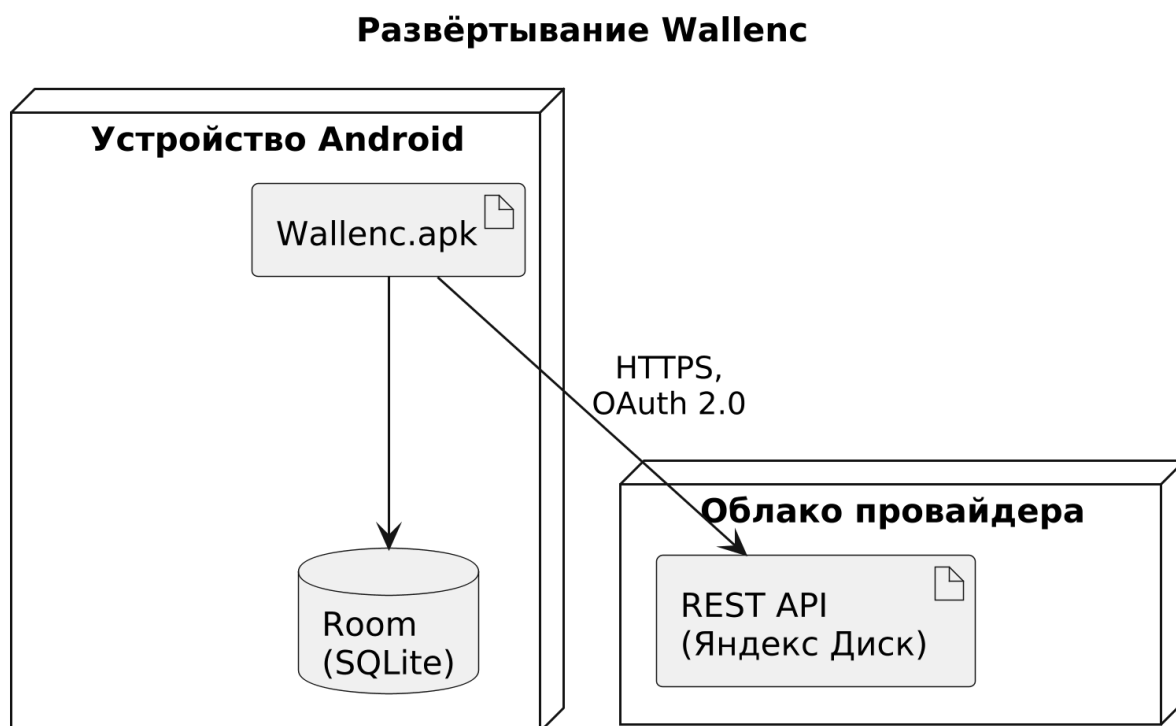


Рисунок 8 — Диаграмма развёртывания

Архитектурные слои MVVM + Clean Architecture и соответствие модулям Gradle — на рисунке 9.

Clean Architecture и модули Gradle

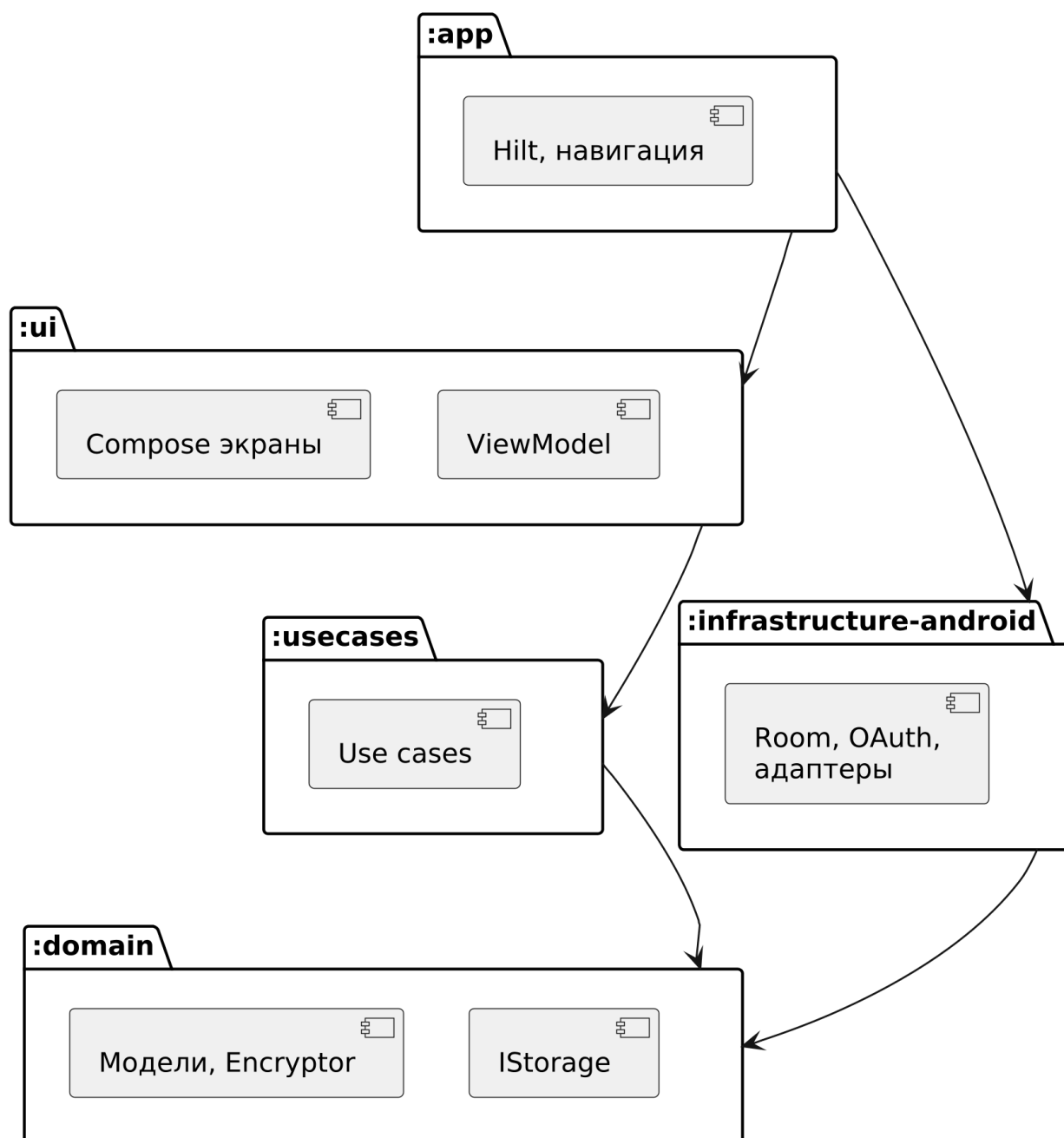


Рисунок 9 — Слои Clean Architecture и модули проекта

2.4 Проектирование локальной базы данных

Служебная БД Room хранит ключи шифрования для пар «исходный storage → зашифрованное представление» (`DbStorageKeyMap`), сериализованные метаданные storage (`DbStorageMetaInfo`), учётные записи Яндекс (`DbYandexAccount`) и группы синхронизации (`DbStorageSyncGroup`). Пользовательский контент в открытом виде в БД не сохраняется —

только параметры, необходимые для восстановления состояния приложения при следующем запуске. Доступ инкапсулирован в DAO и репозиториях модуля :infrastructure-android.

Таблица 4 — Сущности Room и назначение

Сущность	Назначение	Связь с тестами
DbStorageKeyMap	Ключ для sourceUuid storage (связь с зашифрованной копией)	Интеграция
DbStorageMetaInfo	Сериализованные метаданные IStorage (имя, путь, шифрование)	Интеграция
DbYandexAccount	OAuth access token, идентификатор аккаунта	YandexAccountRepositoryTest
DbStorageSyncGroup	Группа UUID для синхронизации	StorageSyncEngineTest

2.5 Проектирование подсистемы синхронизации

Подсистема синхронизации спроектирована как набор независимых операций над журналом изменений каждого Storage. Алгоритм выбирает «победителя» по ревизии, копирует или удаляет файлы на целевом хранилище, не расшифровывая данные на сервере. Блокировки предотвращают одновременную синхронизацию одной группы; при отмене задачи блокировки снимаются (покрыто unit-тестами syncGroupCooperativeCancellationReleasesLocks и др., гл. 5). Подробное описание реализации и выбора источника для каждого пути — в гл. 4 (§ алгоритм согласования журналов, рис. 23).

2.6 Нефункциональные архитектурные решения

Таблица 5 — Нефункциональные решения

Атрибут	Решение
Производительность	Потоковое шифрование, фоновые задачи в :task-runtime

Продолжение таблицы 5

Атрибут	Решение
Безопасность	AES, проверка ключа без полного descrypt, скрывание служебных путей
Расширяемость	vault-contracts, регистрация провайдеров
Сопровождаемость	Модульные тесты 68 + листинги в приложении А
Надёжность	Room-транзакции, восстановление журнала после сбоя записи

2.7 Вывод

Спроектирована клиентская архитектура с разделением domain, infrastructure и presentation, единым интерфейсом vault и заделом под синхронизацию без передачи ключей. Диаграммы зафиксированы в приложении Г.

3 Проектирование пользовательского интерфейса мобильного приложения

3.1 Подготовка проекта и аналитика

3.1.1 Анализ конкурентов и определение сильных и слабых сторон

По результатам сравнения аналогов (табл. 3) для Wallenc выделены сильные стороны: единый UI для локальных и удалённых vault, явное отображение состояния шифрования, диалоги подтверждения деструктивных операций. Слабые стороны конкурентов (привязка к экосистеме, узкая предметная область) учтены при проектировании навигации.

3.1.2 Ограничения и допущения проектирования UI

Полевые интервью пользователей не проводились; сценарии восстановлены по аналогам и отчётам этапов практики. Допущение: пользователь понимает риск потери ключа шифрования.

3.2 Исследования пользовательских сценариев

3.2.1 Потребности и барьеры пользователя

Потребности: хранить файлы в «сейфе» без доверия к облаку; синхронизировать между устройствами. Барьеры: сложность OAuth, страх потери пароля, неочевидность состояния «vault открыт».

3.2.2 Полезные сценарии из аналогов

Заимствованы: список хранилищ с индикацией статуса (Cryptomator, Bitwarden); локальная защита папки (Secure Folder); пошаговый мастер шифрования.

3.3 Проектирование пользовательских потоков

3.3.1 User Story

Таблица 6 — User Story Wallenc

ID	Формулировка	Критерий приёмки
US-1	Создаю storage для файлов в локальном vault на устройстве	Storage в списке (рис. 5)
US-2	Включаю шифрование vault паролем	Статус encrypted, тест T-8
US-3	Открываю зашифрованный vault ключом	Доступ к контенту, тест T-9
US-4	Подключаю Яндекс и удалённый vault	OAuth OK, тест T-10
US-5	Вижу прогресс фоновых задач	Экран задач, тест T-11
US-6	Храню TOTP и текстовые секреты в vault	TwoFaTotpTest, UI IT

3.3.2 Customer Journey Map

CJM сценария «защитить и открыть vault» представлен на рисунке 10.

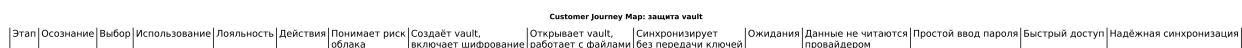


Рисунок 10 — Customer Journey Map: защита и открытие vault

3.3.3 Пользовательский сценарий

Диаграммы потоков: старт приложения и фоновая синхронизация (11), жизненный цикл vault (12), навигация с главного экрана (13).

Wallenc — старт приложения и фоновая синхронизация

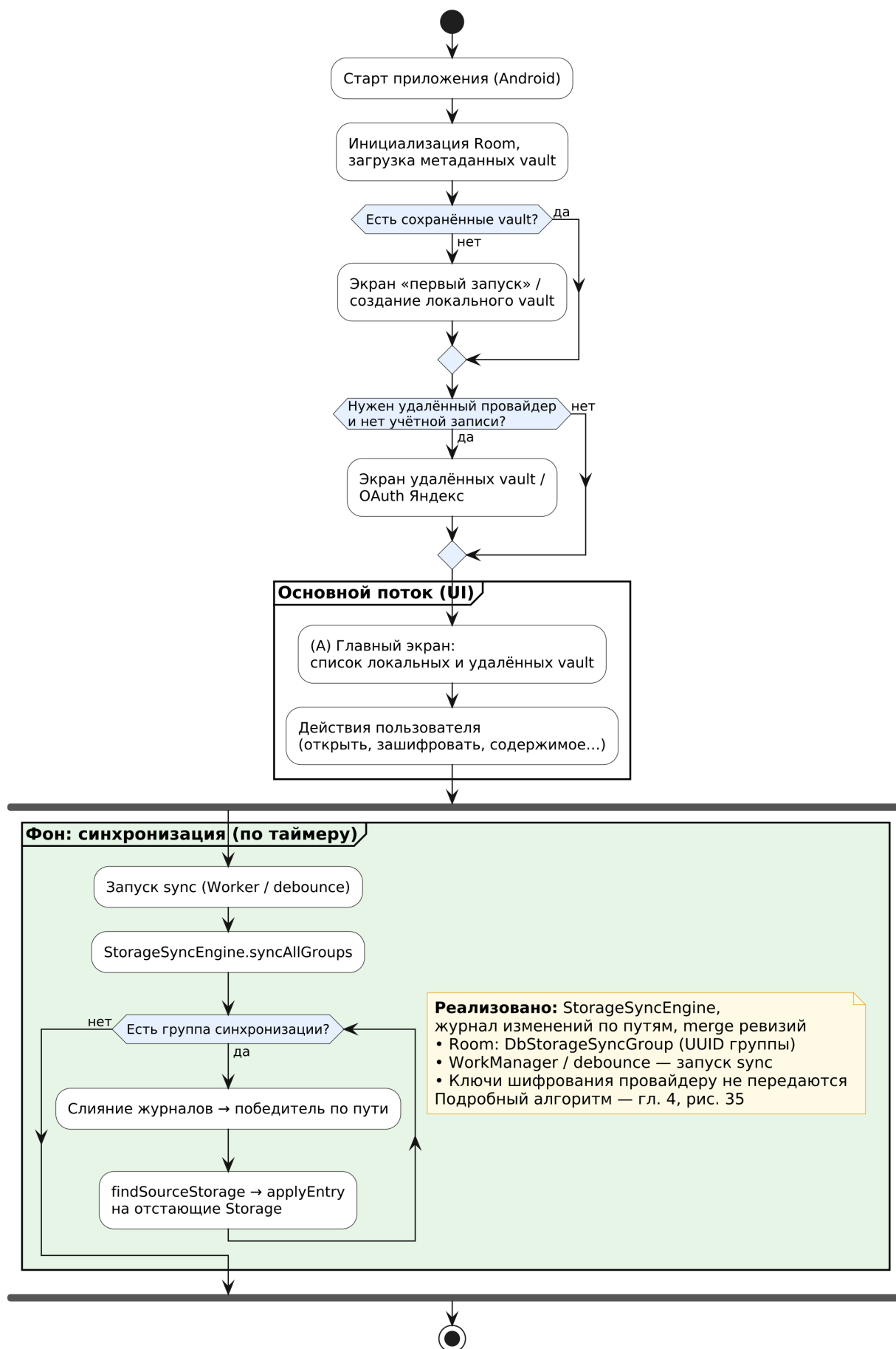


Рисунок 11 — Старт приложения и фоновая синхронизация (проект)

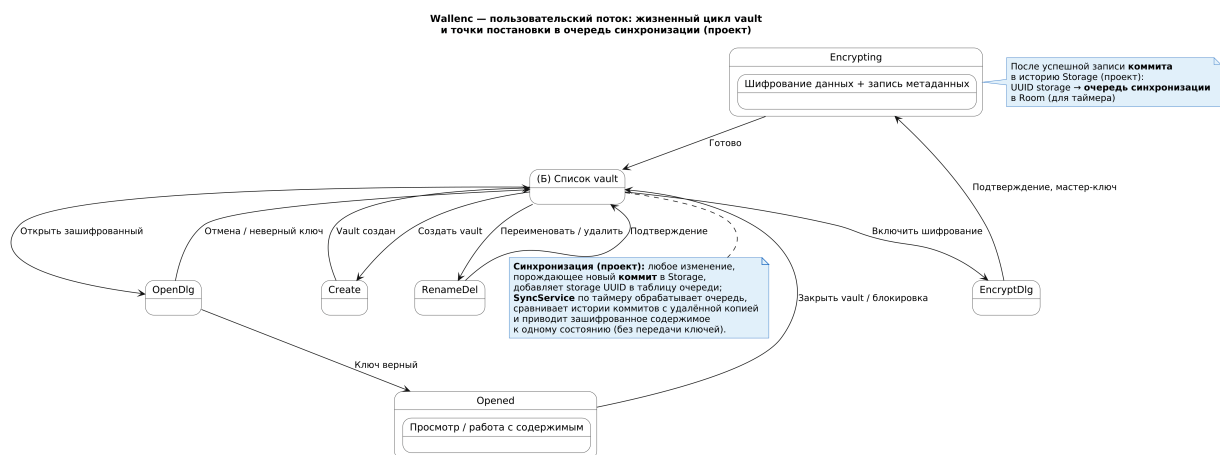


Рисунок 12 — Жизненный цикл vault и очередь синхронизации

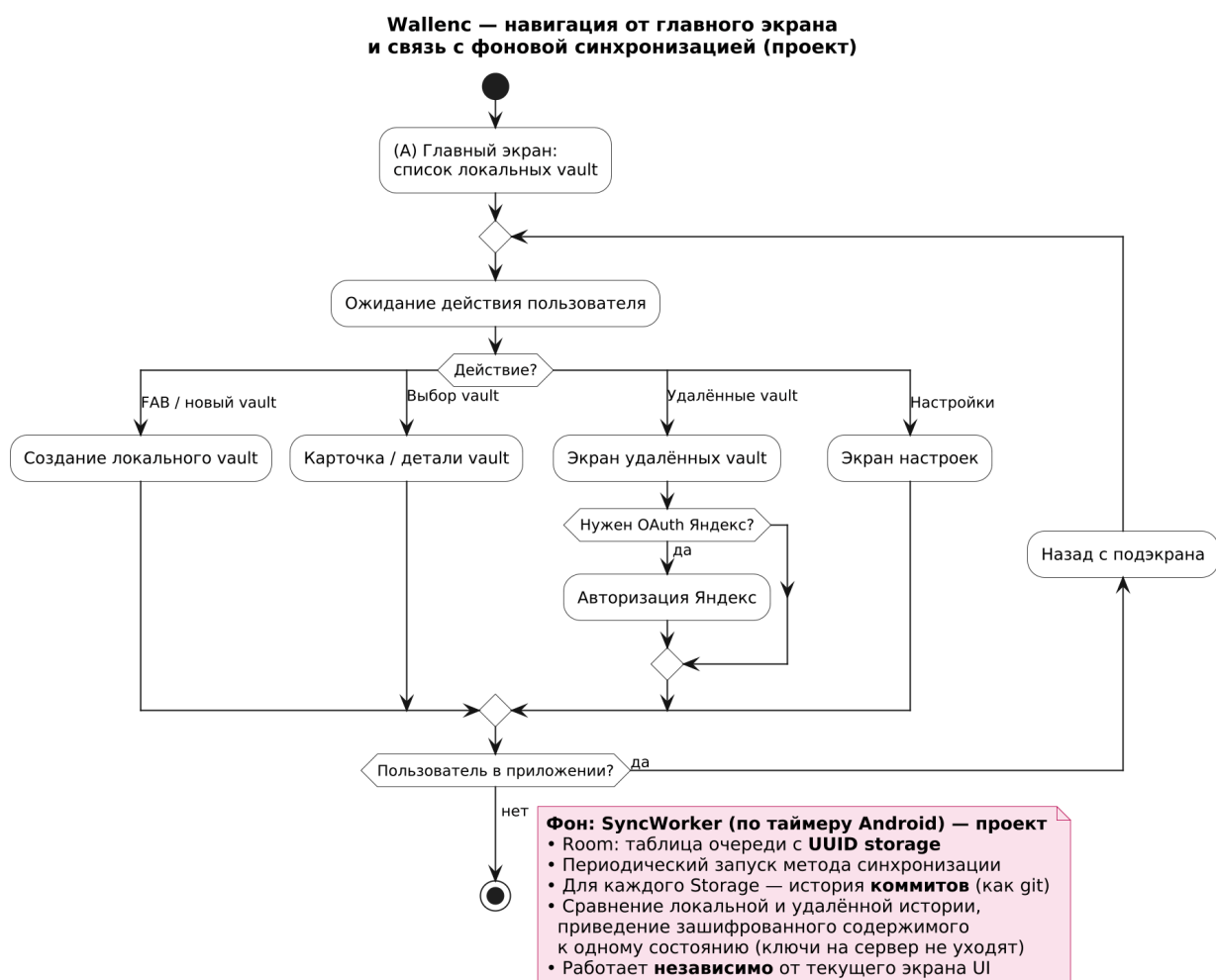


Рисунок 13 — Навигация с главного экрана и SyncWorker

3.4 Проработка прототипа и особенности дизайна

Интерфейс реализован на Jetpack Compose [10]. Экраны локальных и удалённых vault, диалоги шифрования, OAuth, а также разделы текстовых

секретов и 2FA внутри storage показаны на рис. 14–19 и 20–21 (подробно — приложение В и руководство пользователя в приложении Б).

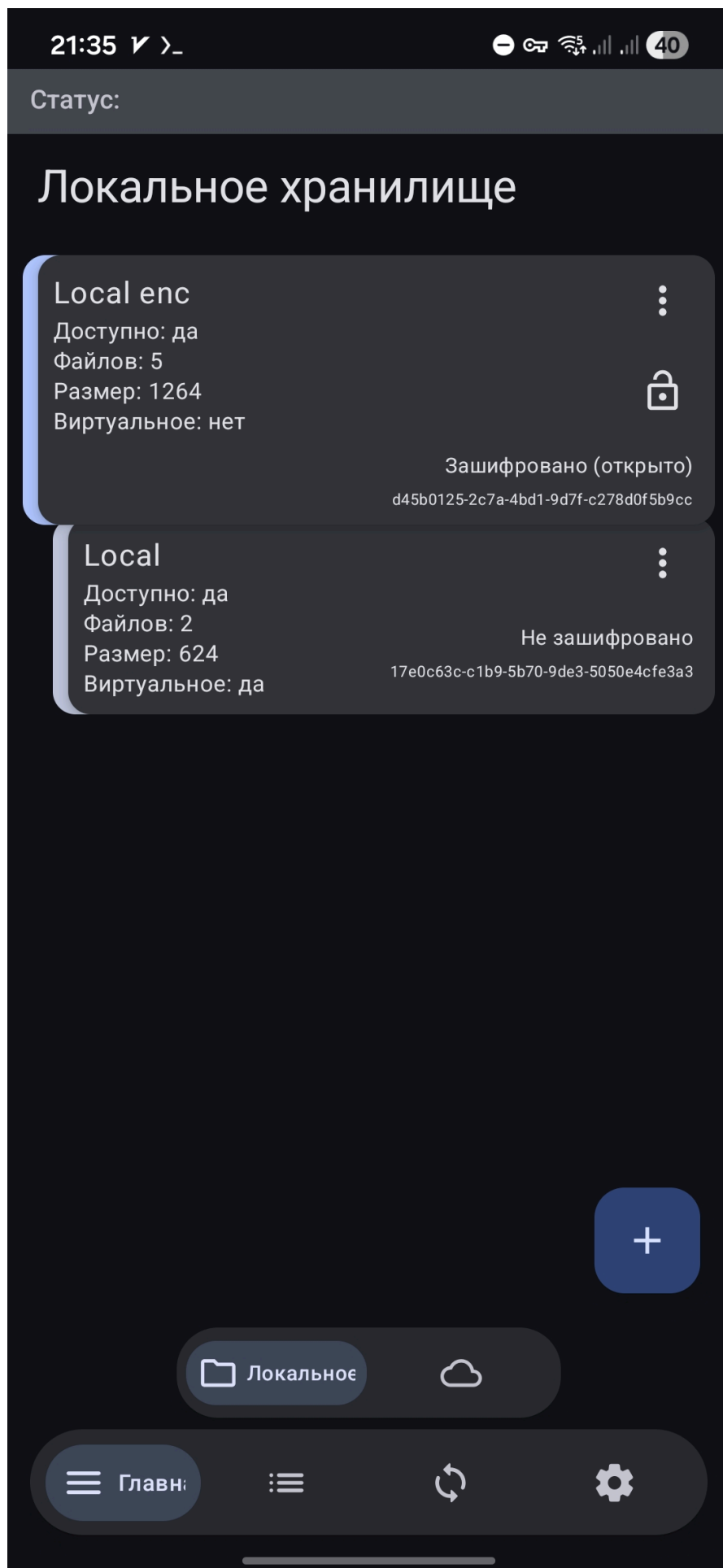


Рисунок 14 — Список storage в локальном vault (экран «локальные vault»)

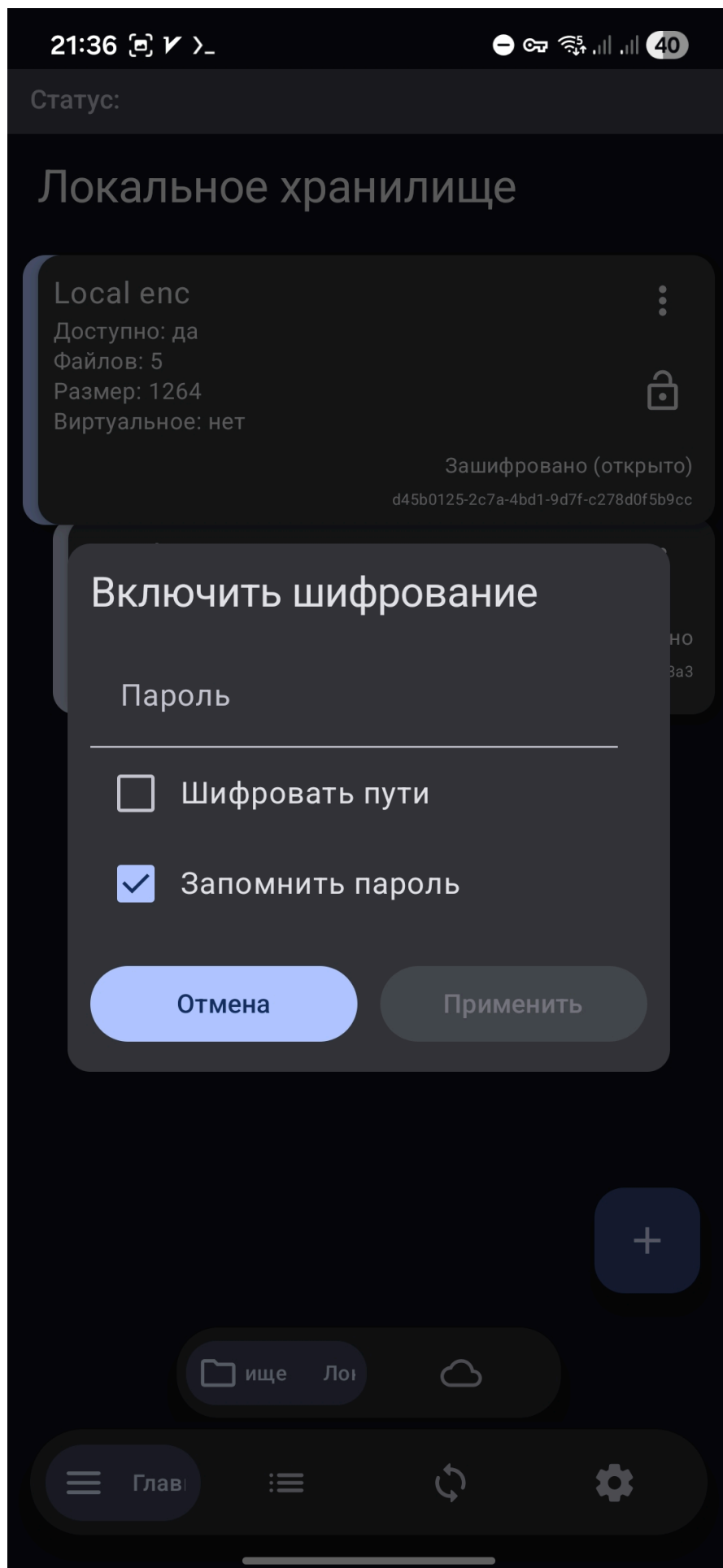


Рисунок 15 — Диалог включения шифрования

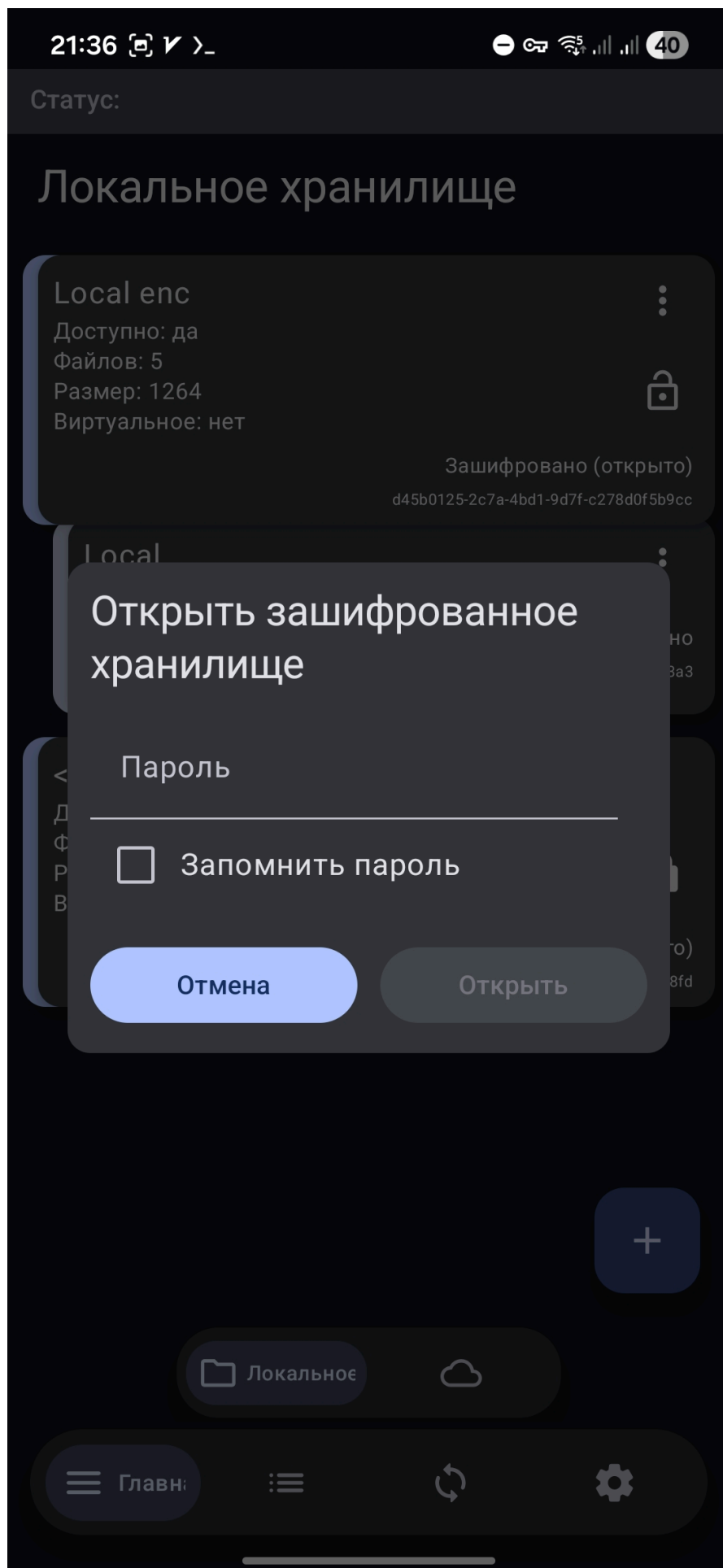


Рисунок 16 — Диалог открытия и закрытия vault
40

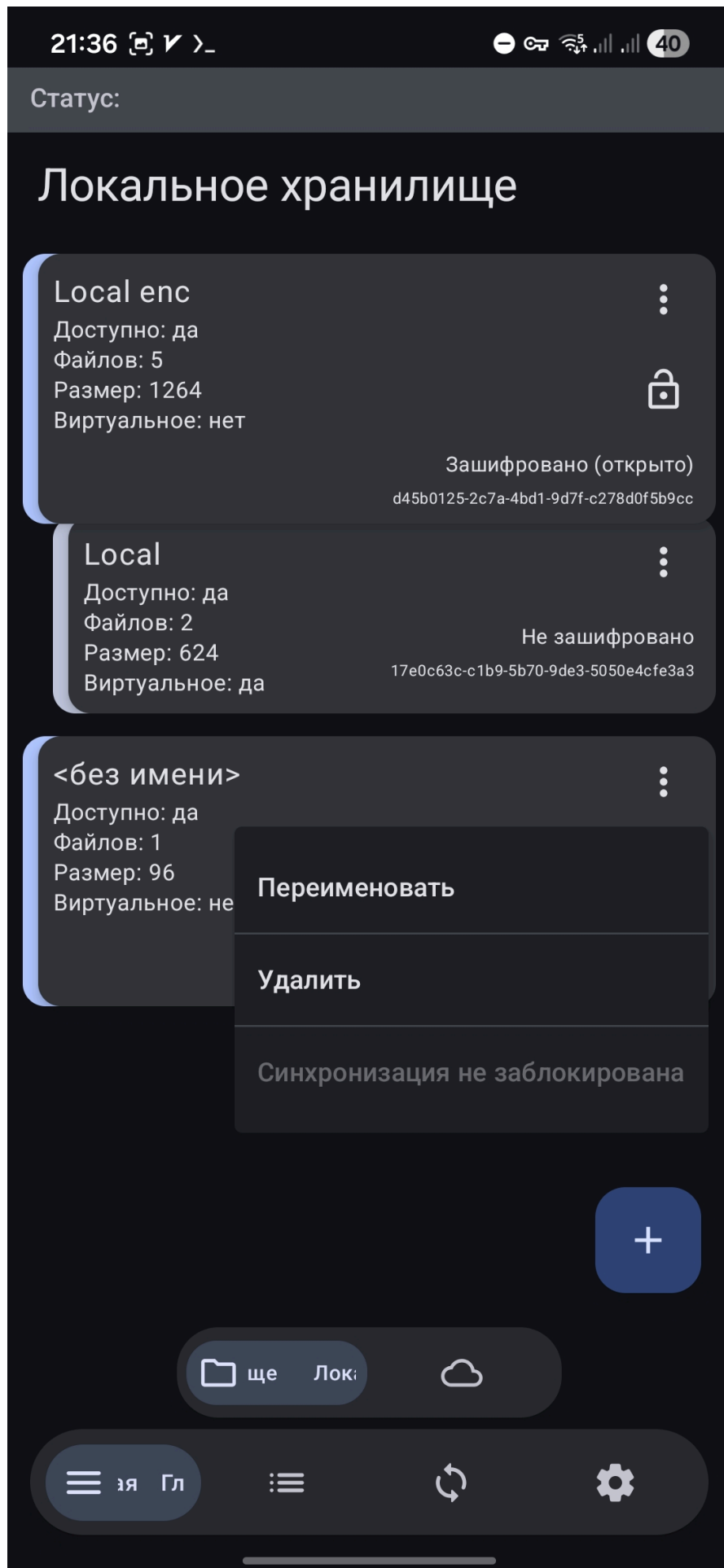


Рисунок 17 — Диалог переименования и удаления
41



Рисунок 18 — Экран удалённых vault
42

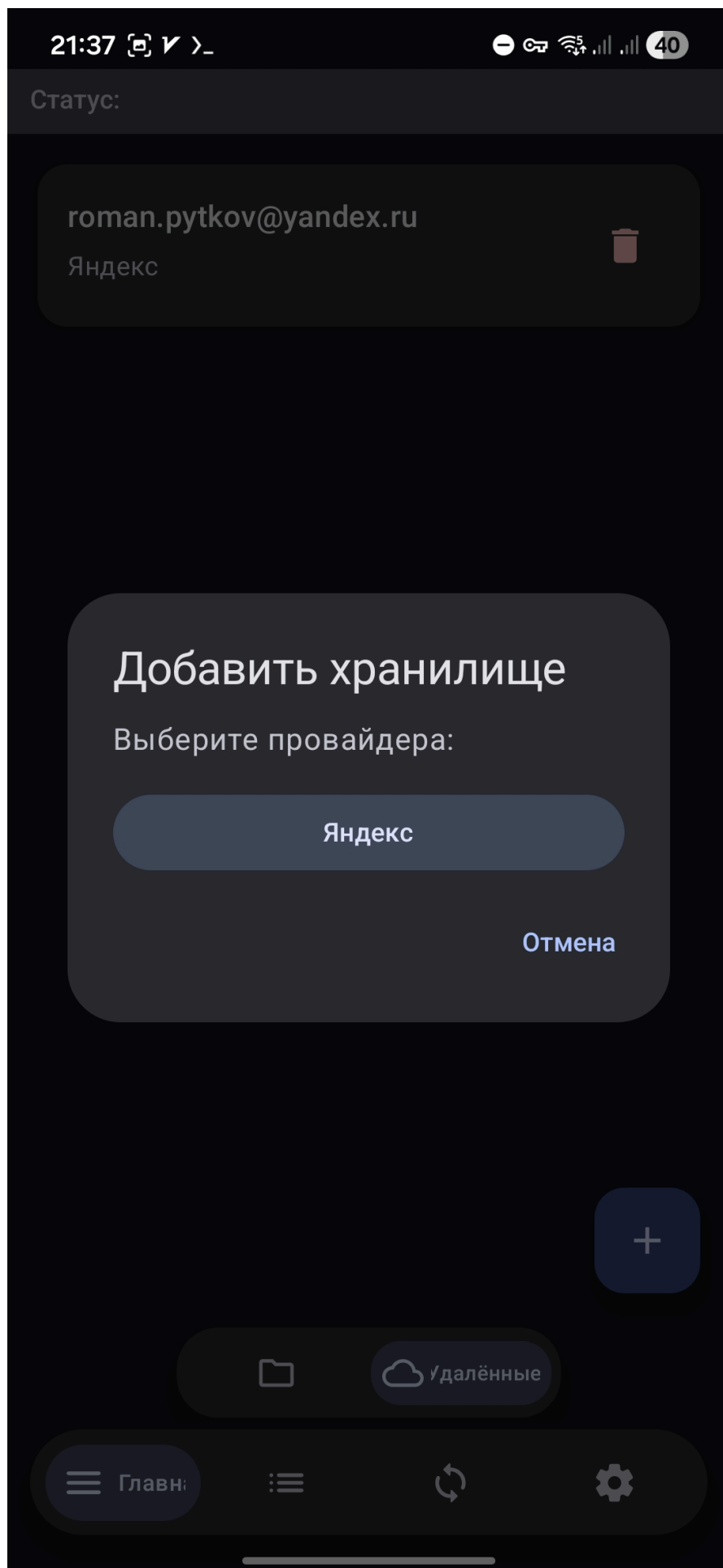


Рисунок 19 — Добавление удалённого vault, OAuth Яндекс

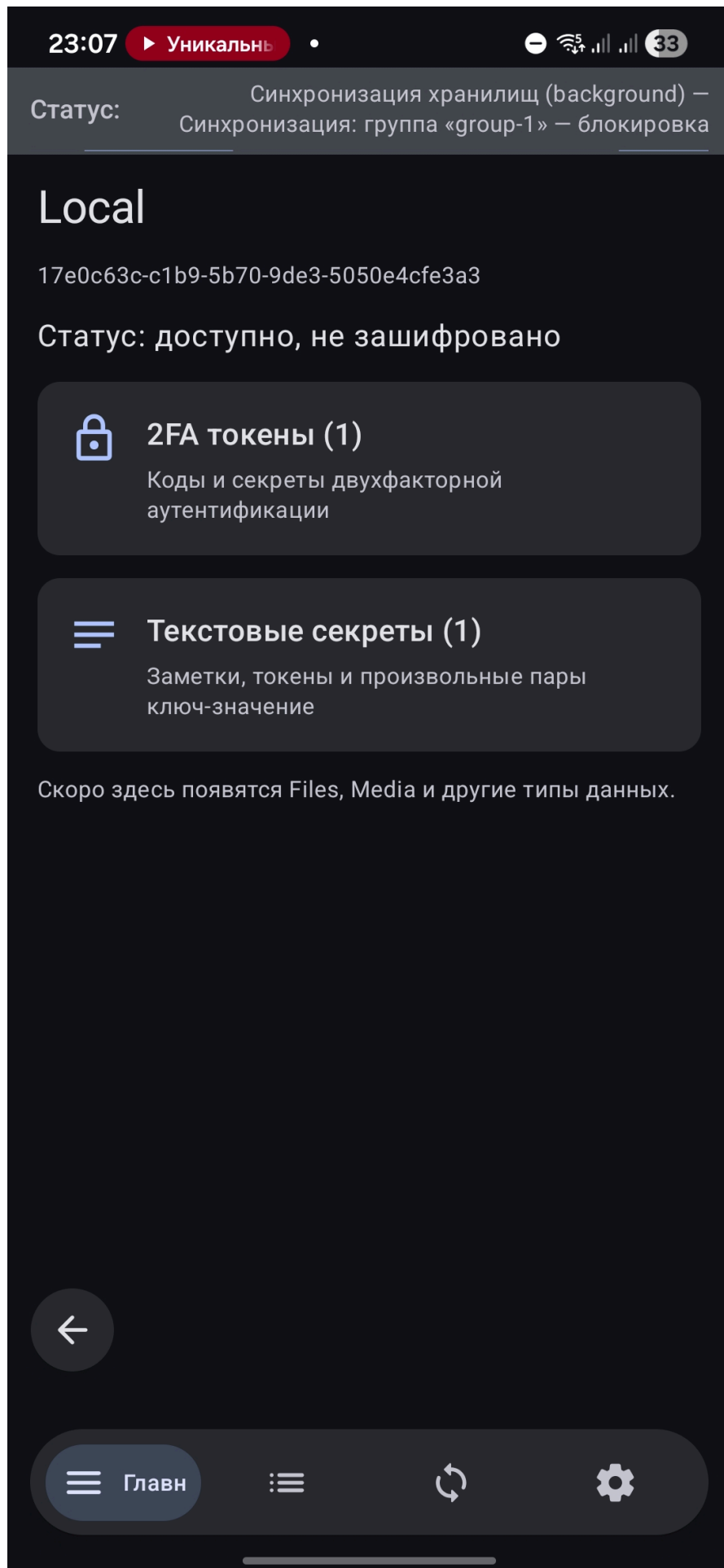


Рисунок 20 — Экран storage: разделы «Секреты» и «2FA»

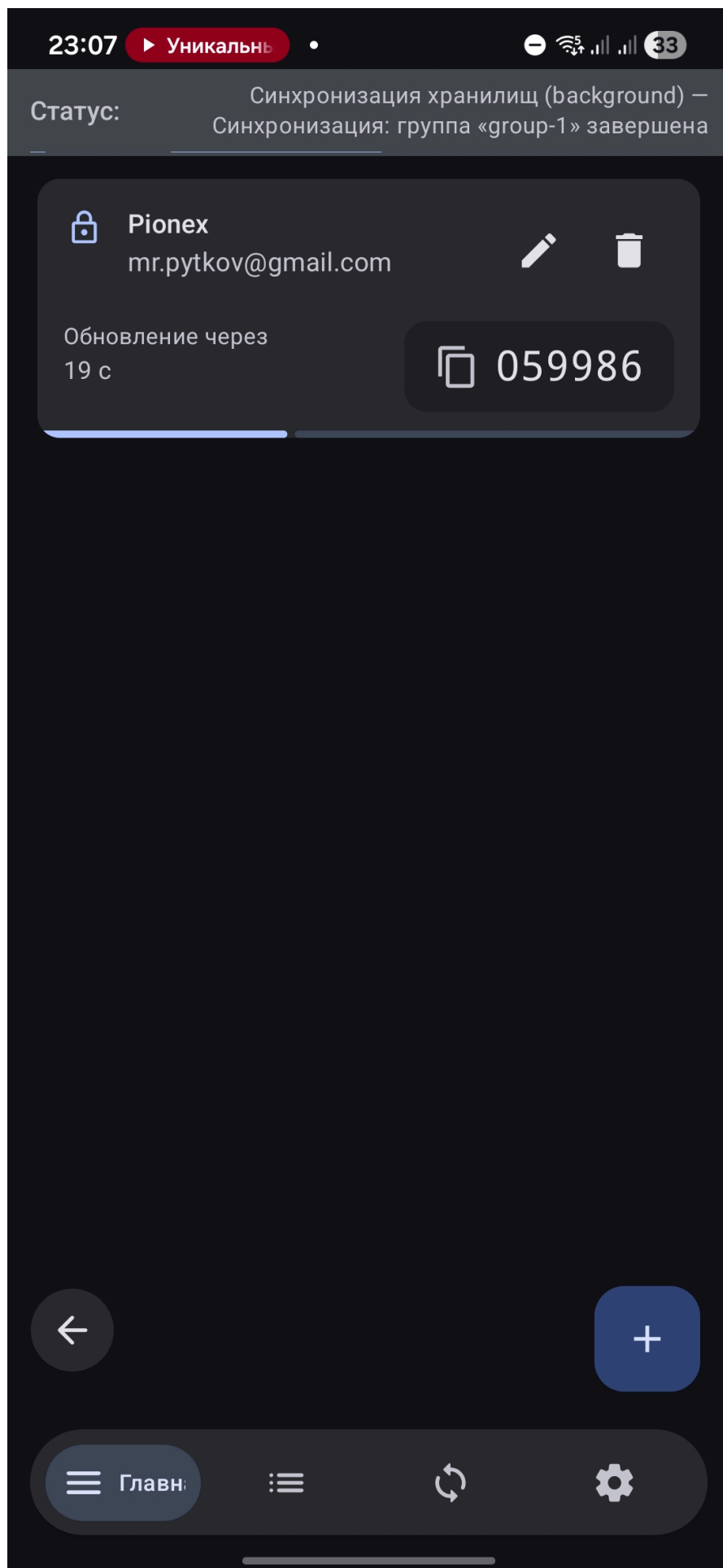


Рисунок 21 — Экран 2FA: список с одним TOTP-токеном

3.5 Требования к эргономике и доступности

Интерфейс должен отображать состояние vault без перехода в технические экраны: признаки «зашифровано», «открыто», «выполняется операция» (isBusy). Диалоги деструктивных действий (удаление, отключение шифрования) требуют явного подтверждения. Тексты сообщений об ошибках OAuth и неверном ключе формулируются нейтрально, без раскрытия внутренних путей и имён файлов.

3.6 Вывод

Спроектированы пользовательские потоки и экранная структура, согласованные с архитектурой и требованиями безопасности. Иллюстрации интерфейса приведены в приложении В.

4 Программная реализация

4.1 Разработка программных модулей

Архитектура реализации следует принятой на этапе проектирования схеме MVVM + Clean Architecture. Модули Gradle разделены по ответственности: контракты vault, доменная логика, сценарии use case, Android-инфраструктура (Room, OAuth, файловые адаптеры), UI и точка входа :app. Такое разбиение позволило параллельно развивать локальный и удалённый контуры и изолировать криптографию от представления данных.

4.1.1 Модуль криптографической защиты данных

Класс Encryptor формирует StorageEncryptionInfo, проверяет ключ и выполняет шифрование/дешифрование на клиенте. Unit-тесты подтверждают корректность для верного и неверного ключа (гл. 5).

4.1.2 Модуль управления vault и шифрованием

Use case ManageStoragesEncryptionUseCase инкапсулирует проверку canEncrypt, включение шифрования и открытие хранилища. ViewModel предотвращает повторный запуск шифрования для занятого storage.

Фрагмент логики включения шифрования:

```
fun enableEncryption(storage: IStorageInfo, password: String,
encryptPath: Boolean) {
    val key = EncryptKey(password)
    viewModelScope.launch {
        when (manageStoragesEncryptionUseCase.canEncrypt(storage)) {
            ManageStoragesEncryptionUseCase.CanEncryptResult.Allowed
-> {
                manageStoragesEncryptionUseCase.enableEncryption(storage, key,
encryptPath)
                manageStoragesEncryptionUseCase.openStorage(storage,
key, true)
            }
        }
        ManageStoragesEncryptionUseCase.CanEncryptResult.AlreadyEncrypted ->
        { /* сообщение */ }
```

```

else -> { /* неподдерживаемая операция */ }
}
}
}

```

Блок-схема сценария — рис. 22.

Поток enableEncryption → checkKey → openStorage

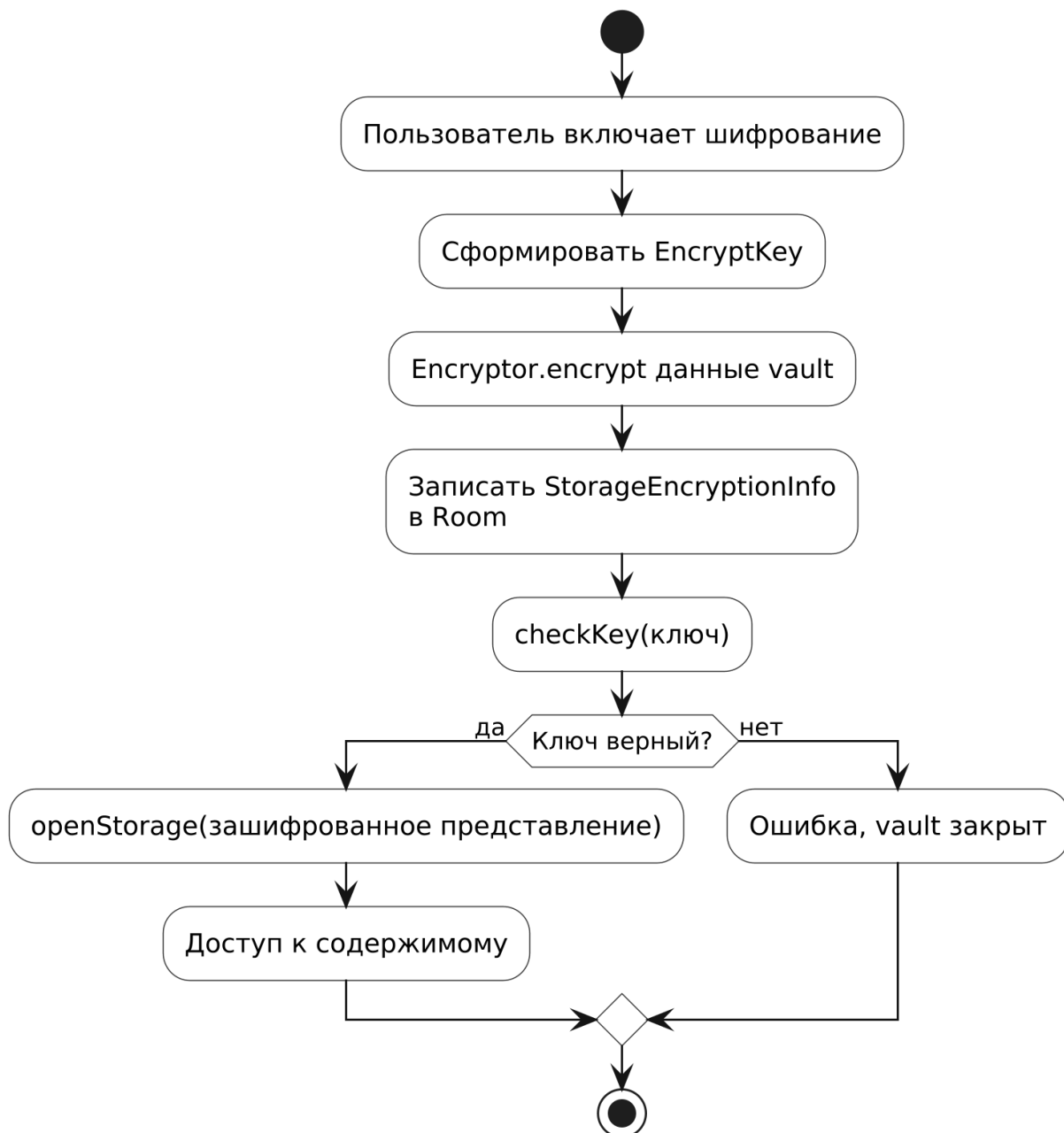


Рисунок 22 — Блок-схема: enableEncryption → checkKey → openStorage

4.1.3 Модуль адаптеров хранилищ

`VaultsManager` агрегирует один `LocalVault` и удалённые `vault`; адаптеры реализуют доступ к файлам внутри каждого `IStorage`. Регистрация удалённых `vault` — через модуль `:vault-contracts`.

4.1.4 Модуль синхронизации хранилищ

В `Room` хранится сущность `DbStorageSyncGroup` — набор `UUID Storage`, которые должны иметь согласованное состояние. Запуск синхронизации выполняется через `RunStorageSyncUseCase / WorkManager` и `debounce` при изменении файлов (рис. 11–13). Движок `StorageSyncEngine` (модуль `:usecases`) реализует согласование журналов изменений; доработка касается в основном политики фонового расписания и UX отображения прогресса.

Для каждого `Storage` ведётся журнал изменений по относительным путям пользовательских файлов. Запись журнала (`StorageSyncJournalEntry`) содержит операцию (`UPSERT`, `TRASH`, `DELETE`) и ревизию (`sequence`, `actorId`, `createdAt`). Ключи шифрования в обмен не включаются — провайдер видит только зашифрованные объекты. Модуль `:task-runtime` обслуживает длительные задачи шифрования и синхронизации без блокировки UI.

4.1.5 Алгоритм согласования журналов синхронизации

Синхронизация одной группы выполняется в несколько этапов (рис. 23).

Wallenc — алгоритм согласования журналов синхронизации (StorageSyncEngine)

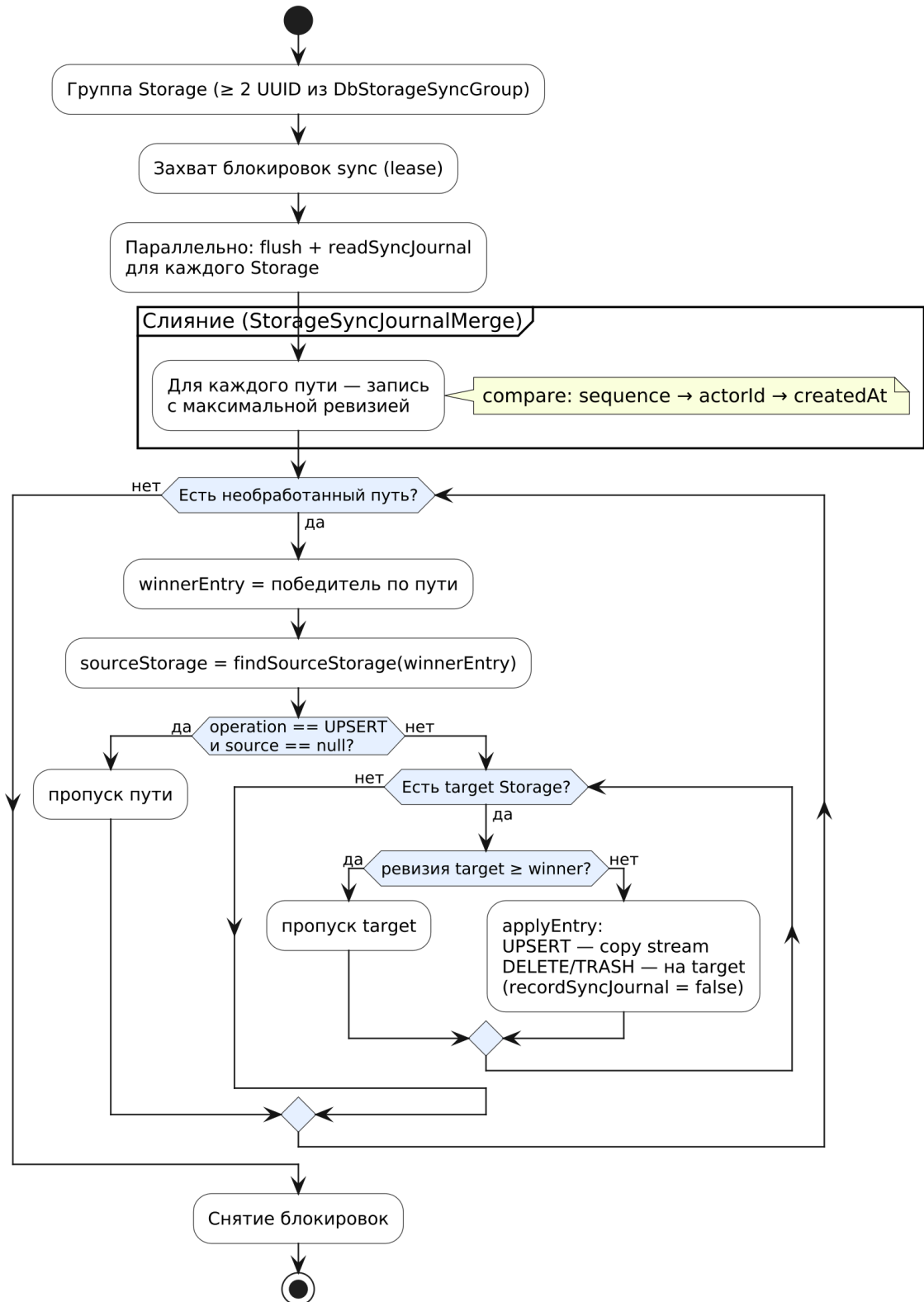


Рисунок 23 — Алгоритм согласования журналов (StorageSyncEngine)

Подготовка. По UUID из DbStorageSyncGroup загружаются объекты IStorage. Если в группе меньше двух хранилищ или несовместимы параметры шифрования, синхронизация пропускается. На каждом accessor

запрашивается блокировка `sync` (lease на удалённом диске — `best-effort`; внутри процесса группа сериализуется `Mutex`). Параллельно для каждого `storage` вызываются `flushPendingSyncJournal()` и `readSyncJournal()`; служебные пути отфильтровываются (`StorageSyncPaths.isSyncableUserPath`).

Слияние журналов. Объект `StorageSyncJournalMerge` объединяет журналы в отображение «путь → победитель». Для каждого пути остаётся запись с наибольшей ревизией. Сравнение реализовано функцией `compareEntries`: сначала `revision.sequence`, при равенстве — `actorId`, затем `createdAt`. Такой порядок обеспечивает детерминированный выбор одной записи при конкурентных изменениях на разных устройствах.

Выбор источника для пути. После слияния для каждой пары (путь, `winnerEntry`) вызывается `findSourceStorage`:

- а) при `UPSERT` — первый `Storage`, у которого запись по этому пути **совпадает** с победителем (`compareEntries == 0`): отсюда читаются байты для копирования;
- б) при `DELETE` или `TRASH` — первый `Storage`, где путь ещё присутствует в журнале (или любой `storage` группы, если записей нет): отсюда инициируется удаление на целях.

Если для `UPSERT` источник не найден, путь пропускается (файл уже отсутствует на всех носителях).

Распространение на цели. Для каждого `target` в группе, отличного от источника, сравнивается ревизия записи на цели с `winnerEntry`. Если цель уже не слабее победителя (`compareEntries(target, winner) >= 0`), шаг пропускается. Иначе вызывается `applyEntry`: для `UPSERT` — потоковое копирование `openRead` → `openWrite`; для `DELETE` / `TRASH` — `delete` или `moveToTrash`. Все операции выполняются с `recordSyncJournal = false`, чтобы не порождать цикл повторной синхронизации. Ошибки отдельных путей учитываются счётчиком `applyFailures`, отмена кооперативная (проверка `syncGeneration`, снятие блокировок в `finally`).

Фрагмент сравнения ревизий и выбора источника:

```
private fun compareEntries(a: StorageSyncJournalEntry, b:
StorageSyncJournalEntry): Int {
    val seqCmp = a.revision.sequence.compareTo(b.revision.sequence)
```

```

        if (seqCmp != 0) return seqCmp
        val actorCmp = a.revision.actorId.compareTo(b.revision.actorId)
        if (actorCmp != 0) return actorCmp
        return a.revision.createdAt.compareTo(b.revision.createdAt)
    }

private fun findSourceStorage(..., winnerEntry:
StorageSyncJournalEntry): IStorage? {
    if (winnerEntry.operation == DELETE || winnerEntry.operation ==
TRASH) {
        return storages.firstOrNull
{ entriesByStorage[it.uuid]?.get(path) != null }
        ?: storages.firstOrNull()
    }
    return storages.firstOrNull { storage ->
        val entry = entriesByStorage[storage.uuid]?.get(path) ?:
return@firstOrNull false
        compareEntries(entry, winnerEntry) == 0
    }
}

```

Ограничения модели. Механизм не является полноценным CRDT: конфликты снимаются фиксированным порядком ревизий, а не автоматическим слиянием содержимого. Содержимое файла при расхождении версий без роста sequence на одном пути не анализируется побайтно. Шифротекст передаётся как есть; расшифровка на стороне провайдера не предполагается. Корректность алгоритма проверена unit-тестами StorageSyncEngineTest (гл. 5): слияние одной записи на путь, пропуск цели с актуальной ревизией, копирование и удаление, cooperative cancellation.

4.1.6 Использование средств ИИ при разработке

Разработка Wallenc велась в два этапа. На первом этапе исполнитель самостоятельно спроектировал доменную модель (иерархия vault → storage → файлы, единый VaultsManager), навигацию между экранами, визуальный стиль UI на Jetpack Compose, границы Gradle-модулей и каркас use case-слоя. Криптографический контур (Encryptor, привязка ключей к storage), журнал синхронизации и сценарии OAuth проектировались и проверялись вручную.

На втором этапе, после готовности архитектурного каркаса, наращивание функционала выполнялось с помощью среды Cursor (модели семейства Composer): адаптеры Yandex Disk и локального storage, движок StorageSyncEngine, экраны 2FA и текстовых секретов, unit-тесты. После каждой генерации код просматривался в diff, запускались модульные тесты (`./gradlew :usecases:test` и смежные модули), критичные сценарии проверялись на устройстве.

Таблица 7 — Роли при разработке (фрагмент)

Этап	Исполнитель	Инструмент / метод
Домен, навигация, UI-концепция	Исполнитель ВКР	Ручное проектирование, Compose
Адаптеры, sync, тесты, доработка UI	Исполнитель ВКР + ревью	Cursor, Gradle test
Шифрование, OAuth, ревизии журнала	Исполнитель ВКР	Ручная ревизия, без автогенерации «вслепую»

ИИ использовался как ускоритель шаблонного и повторяющегося кода, а не как замена проектных решений. Риски (неверные сигнатуры API, лишние зависимости, утечки в логи) снижались обязательной проверкой сборки, отсутствием секретов в репозитории и правилами `.gitignore` для локальных конфигураций.

4.2 Разработка мобильного приложения на Kotlin (Android)

4.2.1 Слой domain

Модуль `:domain` содержит интерфейсы хранилищ и use case. Модуль `:usecases` связывает сценарии приложения.

4.2.2 Слой data

Модуль `:infrastructure-android` реализует Room AppDb (версия 5) с сущностями `DbStorageKeyMap`, `DbStorageMetaInfo`, `DbYandexAccount`, `DbStorageSyncGroup`:

```

@Database(
    entities = [
        DbStorageKeyMap::class,
        DbStorageMetaInfo::class,
        DbYandexAccount::class,
        DbStorageSyncGroup::class,
    ],
    version = 5,
    exportSchema = false,
)
abstract class AppDb : IAppDb, RoomDatabase()

```

4.2.3 Слой presentation

Модуль :ui и :app содержат Compose-экраны, ViewModel и навигацию. OAuth Яндекс запускается из UI удалённых vault:

```

viewModel.yandexSignIn.launch { outcome ->
    when (outcome) {
        is RemoteYandexAuthResult.Success ->
            viewModel.onYandexAuthSuccess(outcome.accessToken)
        is RemoteYandexAuthResult.Failure -> { /* ошибка */ }
        RemoteYandexAuthResult.Cancelled -> { }
    }
}

```

4.3 Взаимодействие подсистем и итоговая архитектура

Зависимости модулей Gradle показаны на рисунке 24. Полный исходный код модулей сборки приведён в приложении А.

Зависимости модулей Gradle

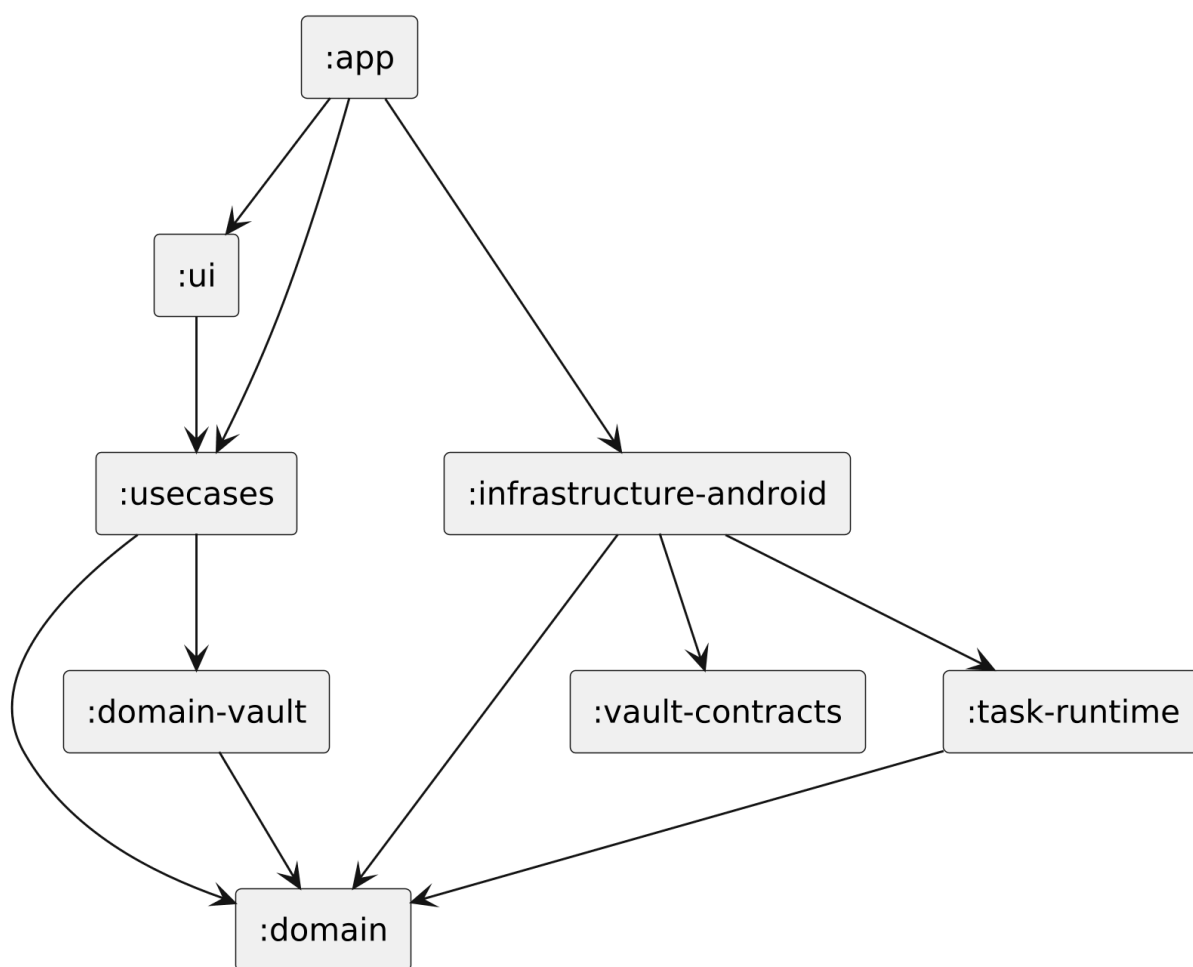


Рисунок 24 — Зависимости модулей Gradle

В основном тексте приведены показательные фрагменты; полные листинги — в приложении А.

4.3.1 Модуль `:vault-contracts`

Определяет дескрипторы vault (`VaultDescriptor`, `DescribedVault`) и контракты регистрации удалённых vault (`VaultRegistrar`, `VaultRegistration`). Реализация агрегатора — `VaultsManager` в `:domain-vault`.

4.3.2 Модуль `:domain-vault`

Содержит реализацию доступа к Yandex Disk API, маппинг сетевых исключений в доменные коды (`VaultThrowableMappingTest`), буфер журнала синхронизации. Unit-тесты репозитория используют подмену HTTP-клиента.

4.3.3 Модуль :task-runtime

TaskOrchestrator управляет очередью долгих операций: единая точка для прогресса, отмены и логов, что используется UI экрана задач (гл. 5, рис. 12).

4.3.4 Модуль :infrastructure-android

Реализует Room (AppDb v5), DAO, репозитории, OAuth-хранилище токенов, файловые адаптеры Android. Модуль — единственная точка зависимости от Android SDK в слое данных.

4.3.5 Сборка и зависимости

Корневой settings.gradle.kts фиксирует восемь включаемых модулей. Версии библиотек централизованы в gradle/libs.versions.toml. Задача test каждого модуля входит в обязательный прогон перед релизом прототипа (см. гл. 5, рис. 30).

4.4 Детализация реализации по модулям Gradle

4.4.1 Модуль :domain

Содержит чистую бизнес-логику: Encryptor, типы ключей, интерфейсы use case для хранилищ. Не зависит от Android SDK, что позволяет выполнять 12 unit-тестов на JVM без эмулятора. Класс Encryptor реализует симметричное шифрование для строк, массивов байт и потоков; метод generateEncryptionInfo формирует соль и параметры для проверки пароля без хранения пароля в открытом виде.

4.4.2 Модуль :usecases

Координирует сценарии приложения: ManageStoragesEncryptionUseCase, движок StorageSyncEngine, операции с 2FA и текстовыми секретами. Здесь сосредоточена наибольшая доля автоматических тестов (25), так как поведение синхронизации задаётся правилами журнала и блокировок, удобными для изоляции на моках файловой системы.

4.4.3 Модуль `:domain-vault`

Инкапсулирует сетевой доступ к Yandex Disk и преобразование исключений Retrofit/OkHttp в доменные коды. Тесты репозитория не обращаются к реальной сети — подставляется фейковый HTTP-слой, что обеспечивает детерминированность CI.

4.4.4 Модуль `:ui`

Предоставляет Compose-экраны, ViewModel, навигацию (WallencDeerLinks), строковые ресурсы для прогресса задач. Unit-тесты проверяют парсеры и маппинг без поднятия Activity; инструментальные тесты (androidTest) валидируют композицию экранов 2FA и секретов.

4.4.5 Модуль `:infrastructure-android`

Связывает Room, DataStore, OAuth Activity Result API, реализации репозитория. Версия схемы БД — 5; миграции отключены (exportSchema = false) на этапе прототипа. Инструментальный YandexAccountRepositoryTest подтверждает CRUD учётной записи в in-memory БД.

4.4.6 Модуль `:task-runtime`

Очередь задач с состояниями pending, running, completed, failed, cancelled. Используется при длительном шифровании каталогов и будущей синхронизации; UI подписывается на Flow прогресса.

4.4.7 Модуль `:vault-contracts` и `:app`

`:vault-contracts` задаёт точку расширения для новых провайдеров. `:app` — Hilt-модули, Application, навигационный граф, сборка APK. Точка входа не содержит бизнес-правил; они делегируются use case.

4.4.8 Журнал разработки и контроль версий

Исходный код размещён в приватном репозитории GitLab ЮФУ; в процессе разработки использовалось зеркало на личном сервере Gitea (<https://git.nullptr.top/nullptr/Wallenc>). Ветвление по этапам практики. Перед защитой

ВКР выполняется полная сборка `./gradlew assembleDebug test` и фиксация отчётов тестирования в `Report/images/` (рис. 27–32).

5 Тестирование программного обеспечения

В ходе работы было организовано тестирование Wallenc на нескольких уровнях: модульные автоматические тесты (JUnit, каталог `src/test` каждого Gradle-модуля), инструментальные тесты (`src/androidTest`), а также ручные функциональные и UI-прогоны. Программа и методика испытаний приведены в приложении Б.

5.1 План тестирования

5.1.1 Цели и задачи испытаний

Основная цель — подтвердить корректность криптографического ядра, доменной логики синхронизации и сценариев UI. Были поставлены следующие задачи:

- а) проверить `Encryptor` и проверку ключа для строк, байтов и потоков;
- б) убедиться в корректном маппинге исключений в коды ошибок;
- в) протестировать движок синхронизации (`StorageSyncEngine`, журнал, блокировки);
- г) проверить оркестратор фоновых задач;
- д) выполнить smoke-тесты навигации, deep link и 2FA/TOTP;
- е) зафиксировать результаты ручных сценариев vault, OAuth и экрана задач.

5.1.2 Объект и уровни тестирования

Таблица 8 — Объекты и уровни тестирования Wallenc

Уровень	Объект	Инструмент	Критерий успеха
Unit	Классы <code>domain</code> , <code>usecases</code> , <code>ui</code> , <code>task-runtime</code> , <code>domain-vault</code>	JUnit 4, JVM	Все тесты модуля успешны
Инструм.	<code>Room</code> , <code>Compose UI</code> , <code>OAuth</code>	<code>AndroidJUnit</code> , эмулятор	Нет падений на целевом API

Продолжение таблицы 8

Уровень	Объект	Инструмент	Критерий успеха
Ручной	Сборка app, пользовательские цепочки	Чек-лист	Сценарии T-1...T-12 пройденны
Регресс.	Синхронизация, шифрование	Повтор unit + выборочный ручной	Нет блокирующих дефектов

5.1.3 Матрица тестовых сценариев

Таблица 9 — Матрица тестовых сценариев

ID	Сценарий	Тип	Авто	Ожидаемый результат
T-1	Проверка ключа шифрования	Unit	Да	Encryptor.checkKey true/false
T-2	Шифрование/ дешифрование строки и байтов	Unit	Да	Симметрия данных
T-3	Потоковое шифрование файла	Unit	Да	Данные после decrypt равны исходным
T-4	Синхронизация группы хранилищ	Unit	Да	Копирование, удаление, trash, блокировки
T-5	2FA TOTP генерация	Unit	Да	Совпадение с эталоном Java OTP
T-6	Маппинг ошибок сети/диска	Unit	Да	Типизированные WallencException
T-7	CRUD storage в LocalVault	Ручной	Нет	Список обновлён (рис. 14)
T-8	Включение шифрования vault	Ручной	Нет	Статус «зашифровано» (рис. 15)
T-9	Открытие/закрытие vault	Ручной	Нет	Доступ только с ключом (рис. 16)
T-10	OAuth Яндекс	Ручной / IT	Частично	Токен в Room (рис. 19)

Продолжение таблицы 9

ID	Сценарий	Тип	Авто	Ожидаемый результат
T-11	Экран задач и уведомления	Ручной	Частично	Прогресс и завершение (рис. 12–13)
T-12	Compose: секреты и 2FA	IT	Да	Отображение без падений (рис. 20–21)

5.1.4 Критерии начала и окончания

Начало: собраны модули проекта; выполняется `./gradlew test`; для инструментальных тестов доступен эмулятор API 26+.

Окончание: все 68 unit-тестов в `src/test` завершились успешно; инструментальные тесты пройдены на эмуляторе; ручной чек-лист T-7...T-12 выполнен; критические дефекты отсутствуют.

5.1.5 Среда и инструменты

Таблица 10 — Тестовая среда

Параметр	Значение
ОС разработки	GNU/Linux, Android Studio
JDK	OpenJDK 17 / 21
Сборка	<code>./gradlew test</code> , <code>./gradlew connectedDebugAndroidTest</code>
Устройство	Эмулятор Pixel 6 API 34; физическое устройство для OAuth

5.2 Модульные тесты (JUnit)

В проекте реализовано 68 автоматических unit-тестов в пяти модулях (`:domain` — 12, `:domain-vault` — 10, `:usecases` — 25, `:ui` — 15, `:task-runtime` — 6). Тесты выполняются на JVM при сборке.

Таблица 11 — Реестр модульных unit-тестов

№	Модуль	Метод	Проверяемое поведение
1	domain	<code>mapsFileNotFoundException</code>	исключение преобразуется в типизированную ошибку Wallenc

Продолжение таблицы 11

№	Модуль	Метод	Проверяемое поведение
2	domain	mapsGenericExceptionToUnknown	исключение преобразуется в типизированную ошибку Wallenc
3	domain	mapsIOExceptionToIoFailed	исключение преобразуется в типизированную ошибку Wallenc
4	domain	preservesWallencException	сохранение уже типизированного Wallenc Exception
5	domain	test bytes encryption with the same key	симметрия шифрования и дешифрования при верном ключе
6	domain	test bytes encryption with the wrong key	дешифрование с неверным ключом завершается ошибкой
7	domain	test correct key for StorageEncryptionInfo	верный ключ проходит проверку checkKey
8	domain	test incorrect key for StorageEncryptionInfo	верный ключ проходит проверку checkKey
9	domain	test stream encryption with the same key	симметрия шифрования и дешифрования при верном ключе
10	domain	test stream encryption with the wrong key	дешифрование с неверным ключом завершается ошибкой
11	domain	test string encryption with the same key	симметрия шифрования и дешифрования при верном ключе
12	domain	test string encryption with the wrong key	дешифрование с неверным ключом завершается ошибкой
13	domain-vault	diskInfoParsesResponse	разбор ответа API diskInfo
14	domain-vault	diskInfoThrowsAuthExceptionOn401	AuthException при HTTP 401
15	domain-vault	flushRestoresPendingOnWriteFailure	откат буфера журнала при сбое записи
16	domain-vault	listReturnsEmptyEmbeddedOn404	пустой список при HTTP 404
17	domain-vault	mapsFileNotFoundToStorageFileNotFound	исключение преобразуется в типизированную ошибку Wallenc
18	domain-vault	mapsHttpExceptionToNetworkHttpFailed	исключение преобразуется в типизированную ошибку Wallenc
19	domain-vault	mapsIllegalStateNotAFile	исключение преобразуется в типизированную ошибку Wallenc
20	domain-vault	mapsMissingOAuthTokenIoToTokenMissing	исключение преобразуется в типизированную ошибку Wallenc
21	domain-vault	mapsSocketTimeoutToOperationTimedOut	исключение преобразуется в типизированную ошибку Wallenc
22	domain-vault	mapsYandexDiskAuthToAuthFailed	исключение преобразуется в типизированную ошибку Wallenc
23	task-runtime	cancelAllMarksRunningTaskCancelled	жизненный цикл фоновой задачи
24	task-runtime	cancelMarksTaskCancelled	жизненный цикл фоновой задачи
25	task-runtime	enqueueCompletesTask	жизненный цикл фоновой задачи
26	task-runtime	failRecordsFailedState	жизненный цикл фоновой задачи
27	task-runtime	logAppendsLine	log appends line
28	task-runtime	progressUpdatesRunningState	жизненный цикл фоновой задачи
29	ui	clearContentProgress_mapsToStringRes	маршрутизация, deep link или подписи UI
30	ui	mapsFeatureStorageNotFound	исключение преобразуется в типизированную ошибку Wallenc

Продолжение таблицы 11

№	Модуль	Метод	Проверяемое поведение
31	ui	mapsStorageIncorrectKey	исключение преобразуется в типизированную ошибку Wallenc
32	ui	mapsUnknown	исключение преобразуется в типизированную ошибку Wallenc
33	ui	matchesTasksAndSettingsHosts	matches tasks and settings hosts
34	ui	matchesWallencViewIntent	маршрутизация, deep link или подписи UI
35	ui	parsesStandardTotpUri	корректность TOTP/OTP: parses standard totp uri
36	ui	rejectsMissingSecret	разбор и валидация входных данных
37	ui	rejectsNonOtpauthScheme	корректность TOTP/OTP: rejects non otpauth scheme
38	ui	rejectsUnrelatedIntent	разбор и валидация входных данных
39	ui	startTestTaskEnqueuesWork	постановка тестовой задачи в очередь orchestrator
40	ui	storageHomeRouteCarriesVaultAndStorageIds	маршрутизация, deep link или подписи UI
41	ui	syncNoGroups_mapsToStringRes	сценарий синхронизации: no groups_maps to string res
42	ui	textSecretsRoutesCarryRequiredArguments	маршрутизация, deep link или подписи UI
43	ui	vaultTask_mapsToStringRes	маршрутизация, deep link или подписи UI
44	usecases	buildTwoFaCodeStateMatchesJavaOtpForKnownSecret	корректность TOTP/OTP: build two fa code state matches java otp for known secret
45	usecases	buildTwoFaCodeStateReturnsNullForInvalidSecret	build two fa code state returns null for invalid secret
46	usecases	deleteWithRecordSyncJournalFalseDoesNotBumpSequence	удаление без записи в журнал не увеличивает sequence
47	usecases	isSyncableUserPathExcludesEncDirAndJournal	пользовательский путь исключает служебные каталоги
48	usecases	mergeKeepsSingleEntryPerPath	слияние журнала оставляет одну запись на путь
49	usecases	openReadDoesNotChangeJournal	чтение без записи не изменяет журнал синхронизации
50	usecases	storageWithEncInfoIsIncompatible	хранилище с шифрованием несовместимо в одной группе sync
51	usecases	storageWithoutEncInfoIsCompatible	хранилище без метаданных шифрования совместимо с синхронизацией
52	usecases	syncAllGroupsReportsNoGroupsWhenEmpty	сценарий синхронизации: all groups reports no groups when empty
53	usecases	syncGroupCooperativeCancellationReleasesLocks	снятие блокировок при отмене задачи пользователем
54	usecases	syncGroupCopiesFileFromSourceToTarget	копирование файла с источника на целевое хранилище в группе
55	usecases	syncGroupDeleteRemovesFileOnTarget	удаление файла на целевом хранилище при синхронизации
56	usecases	syncGroupReleasesLocksAfterSuccessfulSync	снятие блокировок после успешной синхронизации
57	usecases	syncGroupReleasesLocksWhenJournalEmpty	снятие блокировок при пустом журнале
58	usecases	syncGroupReleasesLocksWhenJournalReadFails	снятие блокировок при ошибке чтения журнала

Продолжение таблицы 11

№	Модуль	Метод	Проверяемое поведение
59	usecases	syncGroupSkippedWhenFewerThanTwoStorages	синхронизация пропускается, если в группе меньше двух хранилищ
60	usecases	syncGroupStopsWhenLockCannotBeAcquired	остановка при невозможности захватить блокировку группы
61	usecases	syncGroupTrashSoftDeletesOnTarget	мягкое удаление (trash) на целевом хранилище
62	usecases	syncSkipsWhenTargetRevisionAlreadyWinner	пропуск синхронизации, если ревизия цели уже новее
63	usecases	textSecretsCrudWorksWithOptionalLabels	CRUD-операции и сохранение данных
64	usecases	textSecretsInvalidJsonFallsBackToEmptyList	text secrets invalid json falls back to empty list
65	usecases	totpPeriodProgressIsContinuousWithinPeriod	корректность TOTP/OTP: totp period progress is continuous within period
66	usecases	totpSecondsUntilRefreshCountsDownWithinPeriod	корректность TOTP/OTP: totp seconds until refresh counts down within period
67	usecases	twoFaCrudWorksAndPersists	CRUD-операции и сохранение данных
68	usecases	twoFaInvalidJsonFallsBackToEmptyList	two fa invalid json falls back to empty list

5.2.1 Криптография и доменные ошибки

Класс `EncryptorTest` проверяет сценарии AES: `checkKey`, шифрование строк, байтовых массивов и потоков с верным и неверным ключом (строки 5–14 табл. 11). `WallencExceptionMappingTest` покрывает преобразование файловых и сетевых исключений.

Прогон `./gradlew :domain:test` — на рис. 25.

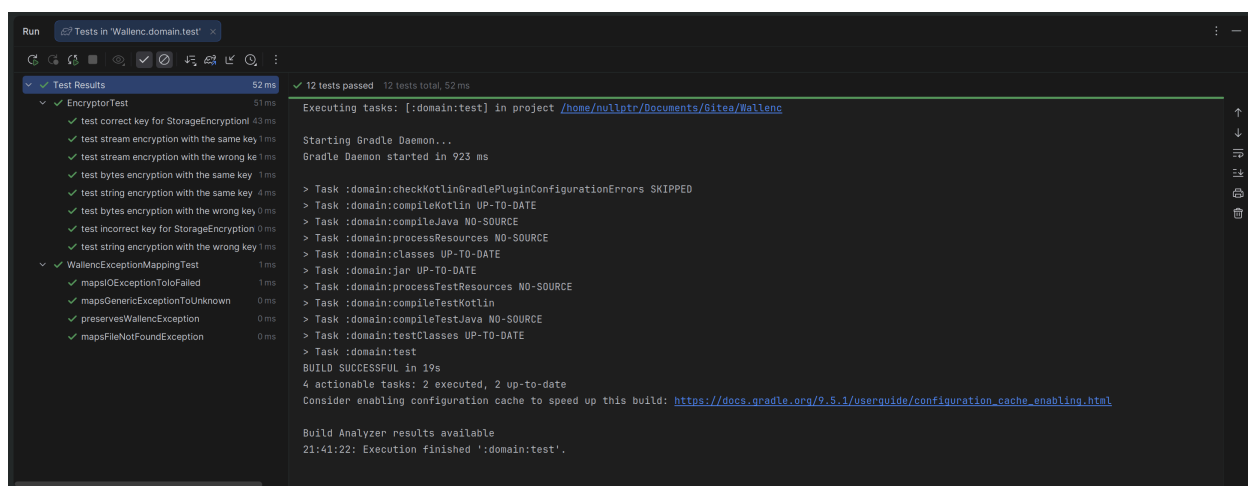


Рисунок 25 — Отчёт Gradle: модуль `:domain`, задача `test`

5.2.2 Синхронизация, 2FA и use cases

`StorageSyncEngineTest` моделирует группы синхронизации, копирование и удаление файлов, soft-delete, отмену и блокировки (строки 52–64 табл. 11); отдельно проверяются слияние журнала (`mergeKeepsSingleEntryPerPath`) и пропуск цели с актуальной ревизией (`syncSkipsWhenTargetRevisionAlreadyWinner`) — см. алгоритм в гл. 4. `TwoFaTotpTest` сверяет TOTP с эталоном Java OTP. `StorageDomainUseCasesTest` проверяет CRUD текстовых секретов и 2FA.

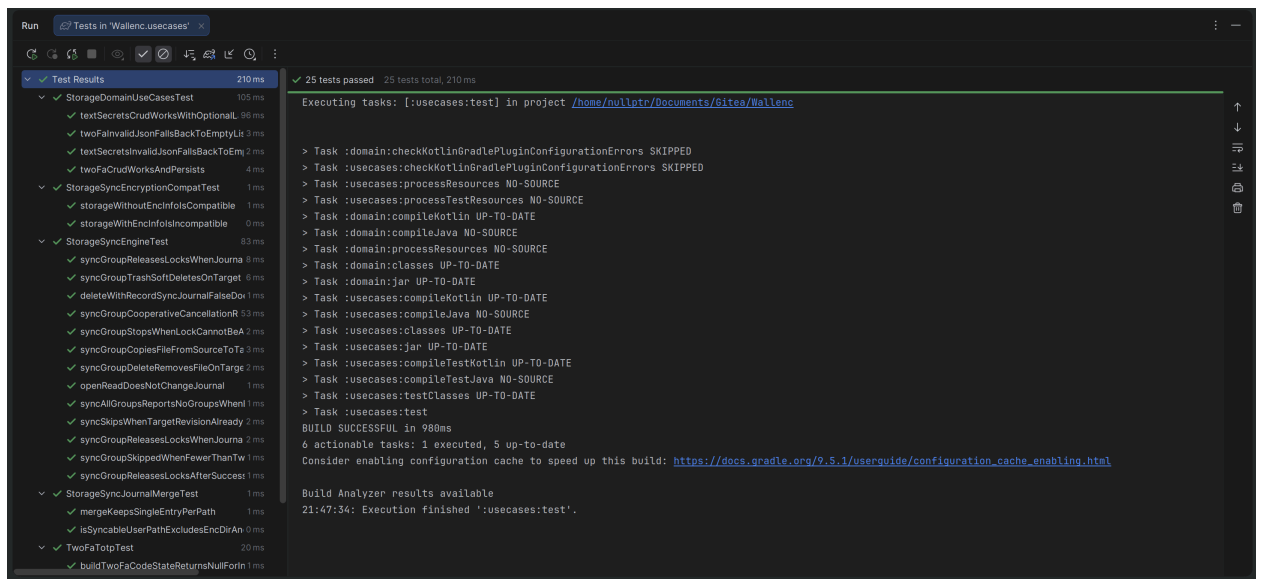


Рисунок 26 — Отчёт Gradle: модуль :usecases

5.2.3 Модуль :domain-vault

`YandexDiskRepositoryTest` использует мок HTTP: разбор `diskInfo`, пустой список при 404, `AuthException` при 401. `VaultThrowableMappingTest` покрывает сетевые и файловые ошибки vault.

5.2.4 Модуль :ui

Проверены чистые функции навигации, deep link, подписи уведомлений, парсинг OTP URI и постановка задачи в очередь (`TaskPipelineViewModelTest`).

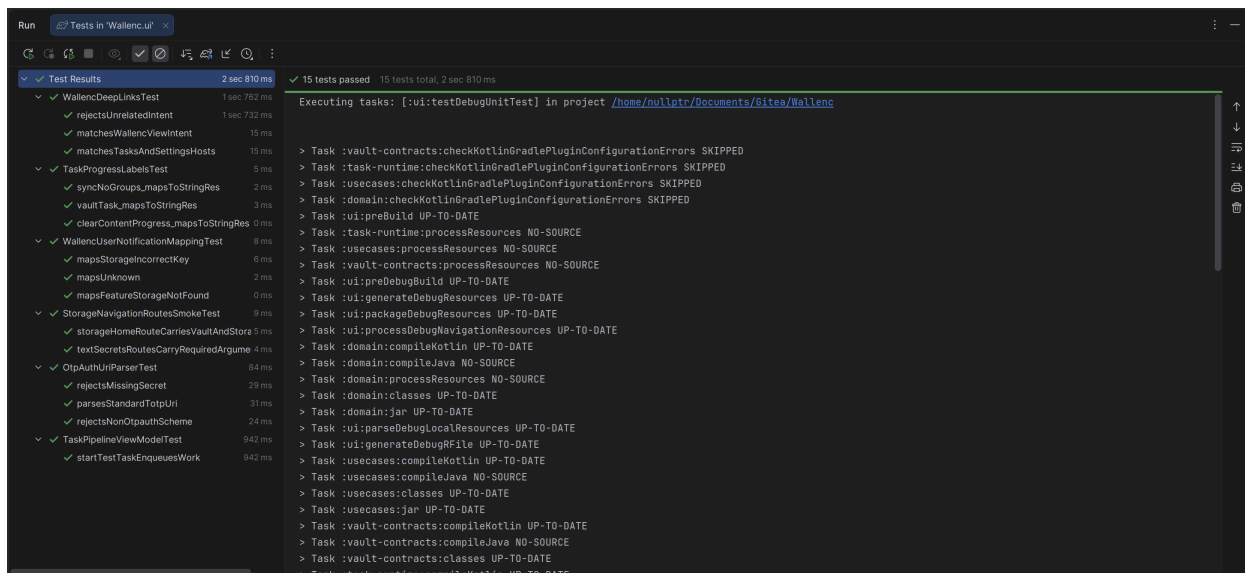


Рисунок 27 — Отчёт Gradle: модуль :ui

5.2.5 Модуль :task-runtime

TaskOrchestratorTest проверяет enqueue, progress, fail, cancel и cancelAll.

5.3 Инструментальные тесты (androidTest)

Таблица 12 — Инструментальные тесты androidTest

Модуль	Класс	Назначение	Методов
:ui	TwoFaTokensScreenContentTest	Compose: экран 2FA токенов	2
:ui	TextSecretsScreenContentTest	Compose: текстовые секреты	2
:infra	YandexAccountRepositoryTest	Room in-memory: аккаунт Яндекс	3
:app	YandexDiskLiveIntegrationTest	Живой API (при наличии токена)	3

Запуск: `./gradlew connectedDebugAndroidTest`. Результат — рис. 28. Отрисовка экранов секретов и 2FA подтверждена скриншотами 20–21.

```

nullptr@thinkbook ~/Documents/Gitea/Wallenc r main ++ ./gradlew connectedDebugAndroidTest
Starting 3 tests on Medium_Phone_API_36.1(AVD) - 16
Finished 3 tests on Medium_Phone_API_36.1(AVD) - 16
Starting 4 tests on Medium_Phone_API_36.1(AVD) - 16
Finished 4 tests on Medium_Phone_API_36.1(AVD) - 16
Starting 3 tests on Medium_Phone_API_36.1(AVD) - 16
Medium_Phone_API_36.1(AVD) - 16 Tests 1/3 completed. (0 skipped) (0 failed)
Medium_Phone_API_36.1(AVD) - 16 Tests 2/3 completed. (0 skipped) (0 failed)
Finished 3 tests on Medium_Phone_API_36.1(AVD) - 16

BUILD SUCCESSFUL in 55s
193 actionable tasks: 19 executed, 174 up-to-date
Consider enabling configuration cache to speed up this build: https://docs.gradle.org/9.5.1/userguide/configuration\_cache\_enabling.html
nullptr@thinkbook ~/Documents/Gitea/Wallenc r main ++

```

Рисунок 28 — Gradle connectedDebugAndroidTest

5.4 Ручное и UI-тестирование

Ручные прогоны выполнялись по чек-листу T-7...T-12 на эмуляторе и физическом устройстве.

Таблица 13 — Протокол ручного тестирования

ID	Шаг	Статус	Фактический результат	Иллюстрация
T-7	Создать storage в LocalVault	OK	Storage в списке	14
T-8	Включить шифрование	OK	Статус encrypted	15
T-9	Открыть/закрыть vault	OK	Контент только при открытом vault	16
T-10	OAuth Яндекс	OK	Запись в DbYandexAccount	19
T-11	Фоновая задача шифрования	OK	Прогресс на экране задач	рис. 12
T-12	Уведомление о завершении	OK	Notification отображён	рис. 13

Чек-лист ручного UI-тестирования Wallenc				
ID	Шаг	Статус	Фактический результат	Рис.
T-7	Создать storage в LocalVault	OK	Storage в списке	5
T-8	Включить шифрование storage	OK	Статус encrypted	6
T-9	Открыть / закрыть storage	OK	Контент при открытом storage	7
T-10	OAuth Яндекс	OK	Запись в DbYandexAccount	10
T-11	Фоновая задача шифрования	OK	Прогресс на экране задач	12
T-12	Уведомление о завершении	OK	Notification отображён	13
Платформа: Android (эмулятор / устройство), 2026				
#				

Рисунок 29 — Чек-лист ручного UI-тестирования

5.5 Отчёт о результатах тестирования

По итогам `./gradlew test` все 68 unit-тестов завершились со статусом PASSED. Инструментальные тесты `:ui` подтвердили отрисовку экранов секретов и 2FA; тесты Room — персистентность учётной записи Яндекс.

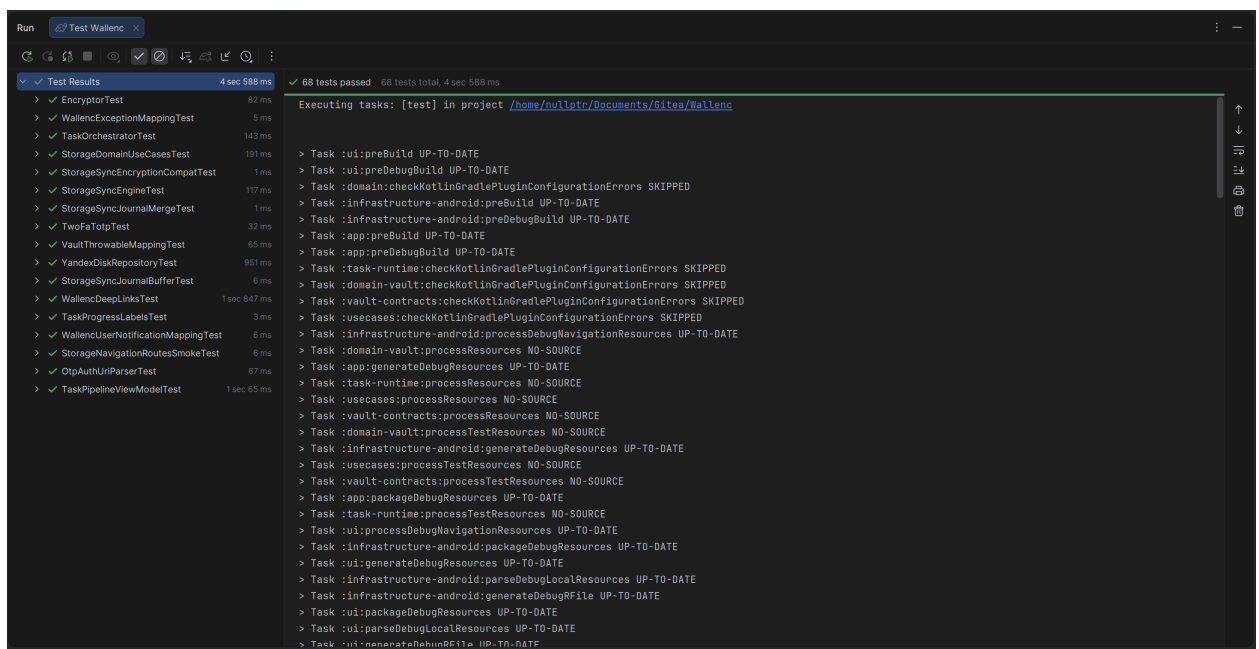


Рисунок 30 — Сводка Gradle test по модулям

Таблица 14 — Трассировка требований → тесты

ФР	Тесты	Комментарий
ФР-1	T-7, StorageDomainUseCasesTest	Storage в LocalVault и CRUD секретов
ФР-2	EncryptorTest, T-8, T-9	Покрытие AES
ФР-3	TextSecretsScreenContentTest	UI + domain

Продолжение таблицы 14

ФР	Тесты	Комментарий
ФР-4	YandexDiskRepositoryTest, T-10	HTTP-мок и ручной OAuth
ФР-5	StorageSyncEngineTest	Синхронизация групп
ФР-6	TaskOrchestratorTest, T-11	Очередь и экран задач

5.6 Вывод

План тестирования выполнен: автоматизированное покрытие охватывает криптографию, синхронизацию, задачи, парсинг OTP и обработку ошибок; ручные сценарии подтвердили пригодность UI для vault и OAuth. Результаты обосновывают готовность прототипа Wallenc к демонстрации и развитию в рамках ВКР.

6 Оценка результатов и экономические показатели

6.1 Обзор рынка программного обеспечения и аналогов

Рынок клиентских решений для защищённого хранения данных представлен продуктами классов «локальный сейф», менеджеры секретов и zero-knowledge файловые клиенты (см. табл. 3, гл. 1). Ниша Wallenc — мобильный универсальный vault без собственного backend приложения с возможностью подключения внешних провайдеров через OAuth.

6.2 Оценка экономических затрат на разработку проекта

Затраты на разработку в рамках практики (09.02.2026–06.05.2026) оцениваются по трудозатратам обучающегося и использованию открытых инструментов (Android Studio, Kotlin, Typst).

Таблица 15 — Структура затрат на разработку (оценка)

Статья затрат	Ед.	Сумма, руб.
Трудозатраты (≈ 480 ч практики)	н/ч	0 (учётная работа в рамках учебного плана)
Лицензии ПО (IDE, SDK)	комплект	0
Устройство для тестирования	1	имеется у исполнителя
Итого прямых денежных затрат		≈ 0

Таблица 16 — Календарный план-график практики (фрагмент)

Период	Результат
09.02–28.02.2026	Анализ, аналоги, ТЗ, архитектура
01.03–28.03.2026	Проектирование БД, стек, OAuth-исследования
29.03–19.04.2026	Реализация ядра, UI, Room, Яндекс
20.04–05.05.2026	Тестирование, иллюстрации, отчётность
06.05.2026	Защита практики, итоговая сборка ПЗ

Календарный график работ совпадает с дневником практики: этап 1 — аналитика и проектирование (февраль–март 2026); этап 2 — реализация (29.03–19.04.2026); оформление документации — апрель–май 2026.

6.3 Модель применения и окупаемости

Сценарий применения — внедрение в ООО НМФ «Нейротех» как внутренний инструмент защищённого хранения файлов сотрудников на корпоративных или личных облачных аккаунтах при сохранении zero-knowledge модели.

Эффект: снижение риска утечки при компрометации провайдера; отсутствие затрат на собственный сервер приложения. Окупаемость для учебно-промышленного прототипа выражается не в прямой прибыли, а в сокращении рисков и возможности доработки продукта без смены архитектуры.

6.4 Вывод

Экономическая оценка показывает низкие прямые затраты на создание прототипа и потенциальную практическую ценность для организации-партнёра. Детальный рыночный анализ приведён в п. 1.3.5; полноценный раздел ТЭО в объёме отраслевых ВКР не требуется согласно рекомендациям кафедры.

7 Вариативная профессиональная компетенция (ВПК-2)

В рамках образовательной программы по направлению 09.03.04 «Программная инженерия» формируется вариативная профессиональная компетенция (ВПК-2):

Способен решать прикладные задачи анализа данных и принятия решений с использованием искусственного интеллекта.

В текущей версии Wallenc модели машинного обучения в runtime не внедрены: продукт строится на zero-knowledge модели, отсутствует доверенный сервер приложения, а выгрузка расшифрованного содержимого vault на облако для обучения противоречила бы заявленным требованиям безопасности. Ниже рассмотрено, какие прикладные задачи анализа данных и автоматизированных решений **могли бы** быть добавлены в перспективе без нарушения клиентской модели угроз.

7.1 Связь компетенции с предметом ВКР

Wallenc уже оперирует структурированными данными, пригодными для анализа: журнал синхронизации (пути, операции, ревизии, метки времени), метаданные файлов (имя, размер, тип по расширению), события фоновых задач. Это не «сырой» пользовательский контент, а агрегаты, которые можно обрабатывать локально. Компетенция ВПК-2 в контексте ВКР раскрывается постановкой задач, выбором данных и моделей, метриками качества и правилами принятия решений — даже если в MVP остаётся только проектный анализ.

7.2 Прикладные задачи анализа данных и принимаемые решения

Таблица 17 — Гипотетические задачи ИИ в экосистеме Wallenc

Задача	Данные для анализа	Тип модели	Где inference	Решение для пользователя
Классификация и теги файлов	Имя, расширение, размер; опционально	Лёгкий классификатор k-NN	On-device (TFLite)	Предложить теги, группировку в vault

Продолжение таблицы 17

Задача	Данные для анализа	Тип модели	Где inference	Решение для пользователя
	эмбеddинг после локального decrypt			
Аномалии в журнале sync	Частота операций по путям, время, actorId	Isolation Forest, пороговые правила	On-device	Предупреждение о подозрительной активности
Похожие имена / дубликаты	Нормализованные имена, размер	Эмбеddинги строк + косинусная близость	On-device	Подсказка при синхронизации или импорте
Импорт секрета со скриншота	Изображение экрана (только по действию пользователя)	OCR (ML Kit)	On-device	Заполнить поле текстового секрета

Каждая строка таблицы связывает **анализ данных** (входные признаки) с **решением** (действие системы или рекомендация), что соответствует формулировке компетенции.

7.3 Модели, обучение и развёртывание

Классификация файлов. Обучающая выборка может формироваться из синтетических имён и публичных датасетов типов файлов без доступа к реальным vault пользователей. Пайплайн: разметка классов (документ, изображение, архив) → обучение компактной сети (MobileNet-подобный backbone) → квантизация → конвертация в TensorFlow Lite [14]. Метрики: accuracy, F1 по классам на hold-out.

Детектор аномалий sync. Признаки: число UPSERT/DELETE за окно времени, доля новых путей, смена actorId. Обучение на журналах, сгенерированных unit-тестами и симуляциями StorageSyncEngineTest, плюс

размеченные «нормальные» и «атакующие» сценарии. Метрики: precision/recall для класса «аномалия». Решение: показать уведомление, не блокировать sync автоматически без подтверждения.

OCR. Google ML Kit Text Recognition [15] позволяет извлечь текст без собственного обучения; дообучение нужно только для специфичных шрифтов. Решение: предзаполнить поле секрета с возможностью редактирования.

Общая схема on-device inference приведена на рис. 31: метаданные и признаки не покидают устройство в открытом виде; обучение на расшифрованном содержимом всех пользователей в облаке для Wallenc неприемлемо. Альтернатива — **федеративное обучение** только на агрегированных градиентах по opt-in, без централизованного хранения файлов.

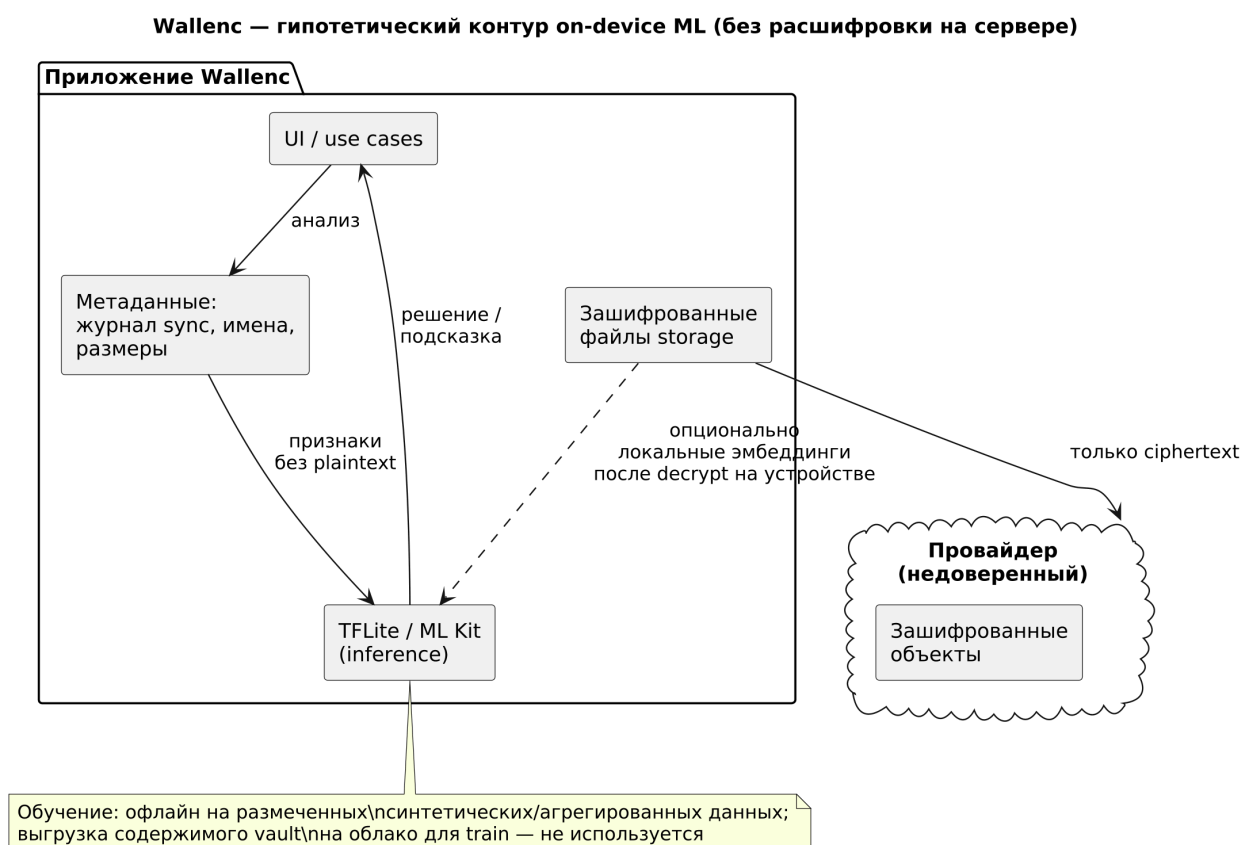


Рисунок 31 — Гипотетический контур on-device ML

7.4 Этика, ограничения и вывод по ВПК-2

Любой модуль ИИ должен быть отключаемым (opt-in), объяснять пользователю основание рекомендации (например, «похожее имя файла») и не подменять явное действие при удалении или синхронизации. Конфликт с E2E

возникает, если отправлять ciphertext или ключи на внешний API inference — в проекте такой путь не рассматривается.

Вывод. Компетенция «решать прикладные задачи анализа данных и принятия решений с использованием искусственного интеллекта» в пояснительной записке раскрыта анализом применимости ML к Wallenc: сформулированы задачи, источники данных, типы моделей, метрики обучения и границы развёртывания on-device. Внедрение в текущий MVP не выполнялось осознанно — в соответствии с целями ВКР по защищённому хранению без доверенного backend.

ЗАКЛЮЧЕНИЕ

В пояснительной записке рассмотрены анализ предметной области, проектирование и реализация мобильного приложения для защищённого хранения пользовательских данных (Wallenc).

По главе 1 сформированы требования и выполнен сравнительный анализ аналогов; обоснован выбор стека Kotlin/Compose/Room/Hilt. По главе 2 спроектированы бизнес-процессы, DFD, UML-диаграммы и модель данных Room. Глава 3 описывает пользовательские сценарии и интерфейсные решения. Глава 4 представляет реализованные модули, алгоритм StorageSyncEngine и методологию разработки с применением ИИ-ассистента Cursor; полный исходный код приведён в приложении А. Глава 5 документирует план и результаты тестирования. Глава 6 содержит краткую экономическую оценку. Глава 7 раскрывает ВПК-2: прикладные задачи анализа данных и принятия решений с использованием ИИ в перспективном развитии Wallenc.

Цель работы достигнута: разработан и протестирован прототип Android-приложения с иерархией vault → storage → файлы, клиентским шифрованием, OAuth Яндекс и реализованным движком синхронизации по журналам ревизий.

Перспективы развития: расширение фонового расписания sync; поддержка дополнительных провайдеров; опциональные on-device модели по сценариям гл. 7; расширение автоматизированных UI-тестов.

Программная документация приведена в приложении Б; иллюстрации интерфейса — в приложении В.

По тестированию подтверждено: 68 модульных unit-тестов, инструментальные тесты Compose и Room, ручной протокол из двенадцати сценариев (гл. 5).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. NIST. Advanced Encryption Standard (AES) [Электронный ресурс]. 2001. URL: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>.
2. Martin R.C. The Clean Architecture [Электронный ресурс]. 2012. URL: <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>.
3. Google. Use a PIN to lock your Safe folder in Files by Google [Электронный ресурс]. 2026. URL: <https://support.google.com/files/answer/9935263>.
4. Bitwarden Help Center [Электронный ресурс]. 2026. URL: <https://bitwarden.com/help/>.
5. Cryptomator Documentation [Электронный ресурс]. 2026. URL: <https://docs.cryptomator.org/en/latest/>.
6. ГОСТ 7.32—2017. Отчёт о научно-исследовательской работе. Структура и правила оформления. 2017.
7. JetBrains. Kotlin Documentation [Электронный ресурс]. 2026. URL: <https://kotlinlang.org/docs/home.html>.
8. Hardt D. The OAuth 2.0 Authorization Framework [Электронный ресурс]. 2012. URL: <https://datatracker.ietf.org/doc/html/rfc6749>.
9. Яндекс. OAuth для сервисов Яндекса [Электронный ресурс]. 2026. URL: <https://yandex.ru/dev/id/doc/ru/>.
10. Google. Get started with Jetpack Compose [Электронный ресурс]. 2026. URL: <https://developer.android.com/develop/ui/compose/documentation>.
11. Google. Save data in a local database using Room [Электронный ресурс]. 2026. URL: <https://developer.android.com/training/data-storage/room>.
12. Google. Hilt [Электронный ресурс]. 2026. URL: <https://developer.android.com/training/dependency-injection/hilt-android>.
13. Google. Guide to app architecture [Электронный ресурс]. 2026. URL: <https://developer.android.com/topic/architecture>.

14. Google. TensorFlow Lite [Электронный ресурс]. 2026. URL: <https://www.tensorflow.org/lite>.
15. Google. ML Kit: Text recognition [Электронный ресурс]. 2026. URL: <https://developers.google.com/ml-kit/vision/text-recognition>.

ПРИЛОЖЕНИЕ А

Листинги исходного кода проекта Wallenc

ПРИЛОЖЕНИЕ А.1

Система сборки

Исходный файл app/build.gradle.kts

```
1  import java.util.Properties
2
3  plugins {
4      alias(libs.plugins.android.application)
5      alias(libs.plugins.compose.compiler)
6      alias(libs.plugins.dagger.hilt)
7      alias(libs.plugins.ksp)
8  }
9
10 val localProps = Properties().apply {
11     val file = rootProject.file("local.properties")
12     if (file.exists()) {
13         file.inputStream().use { load(it) }
14     }
15 }
16
17 android {
18     namespace = "com.github.nullptroma.wallenc.app"
19     compileSdk = 37
20
21     defaultConfig {
22         applicationId = "com.github.nullptroma.wallenc.app"
23         minSdk = 26
24         targetSdk = 37
25         versionCode = 1
26         versionName = "1.0"
27
28         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
29         testInstrumentationRunnerArguments["yandex.oauth.token"] =
30             localProps.getProperty("yandex.test.oauth.token").orEmpty()
31         vectorDrawables {
32             useSupportLibrary = true
33         }
34
35         manifestPlaceholders["YANDEX_CLIENT_ID"] = "0854a43a284a445480c5ced2258f2069"
36     }
```

```

37
38     buildTypes {
39         release {
40             isMinifyEnabled = false
41             proguardFiles(
42                 getDefaultProguardFile("proguard-android-optimize.txt"),
43                 "proguard-rules.pro"
44             )
45         }
46     }
47     compileOptions {
48         sourceCompatibility = JavaVersion.VERSION_17
49         targetCompatibility = JavaVersion.VERSION_17
50     }
51
52     buildFeatures {
53         compose = true
54     }
55
56     packaging {
57         resources {
58             excludes += "/META-INF/{AL2.0,LGPL2.1}"
59         }
60     }
61 }
62
63 kotlin {
64     jvmToolchain(17)
65 }
66
67 dependencies {
68     // Timber
69     implementation(libs.timber)
70
71     // Yandex
72     implementation(libs.yandex.oauth)
73
74     // Hilt
75     implementation(libs.dagger.hilt)
76     ksp(libs.dagger.hilt.compiler)
77     implementation(libs.androidx.hilt.work)
78     ksp(libs.androidx.hilt.compiler)
79
80     implementation(libs.androidx.core.ktx)
81     implementation(libs.androidx.appcompat)

```

```

82     implementation(libs.androidx.datastore.preferences)
83     implementation(libs.androidx.work.runtime.ktx)
84     implementation(libs.androidx.lifecycle.runtime.ktx)
85     implementation(libs.androidx.activity.compose)
86     implementation(platform(libs.androidx.compose.bom))
87     implementation(libs.androidx.ui)
88
89     testImplementation(libs.junit)
90     androidTestImplementation(libs.androidx.junit)
91     androidTestImplementation(libs.androidx.espresso.core)
92     androidTestImplementation(libs.kotlinx.coroutines.test)
93     androidTestImplementation(libs.room.testing)
94     androidTestImplementation(platform(libs.androidx.compose.bom))
95     androidTestImplementation(libs.okhttp3)
96     androidTestImplementation(libs.retrofit)
97     androidTestImplementation(libs.retrofit.converter.jackson)
98     androidTestImplementation(libs.jackson.module.kotlin)
99     androidTestImplementation(libs.jackson.datatype.jsr310)
100
101     implementation(project(":domain"))
102     implementation(project(":usecases"))
103     implementation(project(":domain-vault"))
104     implementation(project(":infrastructure-android"))
105     implementation(project(":task-runtime"))
106     implementation(project(":ui"))
107     implementation(project(":vault-contracts"))
108 }

```

Исходный файл build.gradle.kts

```
1  plugins {
2      alias(libs.plugins.android.application) apply false
3      alias(libs.plugins.kotlin.android) apply false
4      alias(libs.plugins.compose.compiler) apply false
5      alias(libs.plugins.android.library) apply false
6      alias(libs.plugins.dagger.hilt) apply false
7      alias(libs.plugins.ksp) apply false
8      alias(libs.plugins.jetbrains.kotlin.jvm) apply false
9  }
10
11  allprojects {
12      tasks.withType<JavaCompile> {
13          options.compilerArgs.addAll(listOf("-Xlint:unchecked", "-Xlint:deprecation"))
14      }
15  }
16
```

Исходный файл domain-vault/build.gradle.kts

```
1  plugins {
2      id("java-library")
3      alias(libs.plugins.jetbrains.kotlin.jvm)
4  }
5
6  kotlin {
7      jvmToolchain(17)
8  }
9
10 dependencies {
11     // jackson
12     implementation(libs.jackson.module.kotlin)
13     implementation(libs.jackson.datatype.jsr310)
14
15     // Retrofit
16     implementation(libs.retrofit)
17     implementation(libs.retrofit.converter.scalars)
18     implementation(libs.retrofit.converter.jackson)
19
20     implementation(libs.okhttp3)
21     implementation(libs.kotlinx.coroutines.core)
22
23     testImplementation(libs.junit)
24     testImplementation(libs.kotlinx.coroutines.test)
25     testImplementation(libs.mockwebserver)
26
27     implementation(project(":domain"))
28     implementation(project(":vault-contracts"))
29 }
```

Исходный файл domain/build.gradle.kts

```
1  plugins {
2      id("java-library")
3      alias(libs.plugins.jetbrains.kotlin.jvm)
4  }
5
6  java {
7      toolchain {
8          languageVersion = JavaLanguageVersion.of(17)
9      }
10 }
11
12 kotlin {
13     jvmToolchain(17)
14 }
15
16 dependencies {
17     implementation(libs.kotlinx.coroutines.core)
18     testImplementation(libs.junit)
19     testImplementation(libs.kotlinx.coroutines.test)
20 }
```


Исходный файл gradle.properties

```
1  # Project-wide Gradle settings.
2  # IDE (e.g. Android Studio) users:
3  # Gradle settings configured through the IDE *will override*
4  # any settings specified in this file.
5  # For more details on how to configure your build environment visit
6  # http://www.gradle.org/docs/current/userguide/build_environment.html
7  # Specifies the JVM arguments used for the daemon process.
8  # The setting is particularly useful for tweaking memory settings.
9  org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
10 # When configured, Gradle will run in incubating parallel mode.
11 # This option should only be used with decoupled projects. For more details, visit
12 # https://developer.android.com/r/tools/gradle-multi-project-decoupled-projects
13 # org.gradle.parallel=true
14 # AndroidX package structure to make it clearer which packages are bundled with the
15 # Android operating system, and which are packaged with your app's APK
16 # https://developer.android.com/topic/libraries/support-library/androidx-rn
17 android.useAndroidX=true
18 # Kotlin code style for this project: "official" or "obsolete":
19 kotlin.code.style=official
20 # Enables namespacing of each library's R class so that its R class includes only the
21 # resources declared in the library itself and none from the library's dependencies,
22 # thereby reducing the size of the R class for that library
23 android.nonTransitiveRClass=true
24
```

Исходный файл gradle/libs.versions.toml

```
1  [versions]
2  agp = "9.2.1"
3  jacksonModuleKotlin = "2.21.3"
4  javaxInject = "1"
5  kotlin = "2.3.21"
6  coreKtx = "1.18.0"
7  junit = "4.13.2"
8  junitVersion = "1.3.0"
9  androidxTestCore = "1.7.0"
10 espressoCore = "3.7.0"
11 kotlinReflect = "2.3.21"
12 kotlinxCoroutinesCore = "1.11.0"
13 kotlinXSerializationJson = "1.11.0"
14 lifecycleRuntimeKtx = "2.10.0"
15 activityCompose = "1.13.0"
16 composeBom = "2026.05.00"
17 navigation = "2.9.8"
18 hiltNavigation = "1.3.0"
19 timber = "5.0.1"
20 yandexAuthSdk = "3.2.0"
21 daggerHilt = "2.59.2"
22 ksp = "2.3.7"
23 room = "2.8.4"
24 retrofit = "3.0.0"
25 okhttp = "5.3.2"
26 workRuntime = "2.11.2"
27 hiltWork = "1.3.0"
28 cameraX = "1.6.1"
29 zxing = "3.5.3"
30 javaOtp = "0.4.0"
31 appcompat = "1.7.1"
32 datastore = "1.2.1"
33 mockk = "1.14.9"
34 robolectric = "4.16.1"
35 androidxArchCore = "2.2.0"
36
37 [libraries]
38 jackson-datatype-jsr310 = { module = "com.fasterxml.jackson.datatype:jackson-datatype-
39 jsr310" }
40 jackson-module-kotlin = { module = "com.fasterxml.jackson.module:jackson-module-kotlin",
41 version.ref = "jacksonModuleKotlin" }
42 javax-inject = { module = "javax.inject:javax.inject", version.ref = "javaxInject" }
43 kotlin-reflect = { module = "org.jetbrains.kotlin:kotlin-reflect", version.ref =
44 "kotlinReflect" }
```

```

42     kotlinxcoroutines-core = { module = "org.jetbrains.kotlinx:kotlinx-coroutines-core",
    version.ref = "kotlinxCoroutinesCore" }
43     kotlinxcoroutines-test = { module = "org.jetbrains.kotlinx:kotlinx-coroutines-test",
    version.ref = "kotlinxCoroutinesCore" }
44     mockk = { module = "io.mockk:mockk", version.ref = "mockk" }
45     mockwebserver = { module = "com.squareup.okhttp3:mockwebserver", version.ref = "okhttp" }
46     room-testing = { group = "androidx.room", name = "room-testing", version.ref = "room" }
47     roboelectric = { module = "org.robolectric:roboelectric", version.ref = "roboelectric" }
48     androidx-arch-core-testing = { group = "androidx.arch.core", name = "core-testing",
    version.ref = "androidxArchCore" }
49     kotlinx-serialization-json = { module = "org.jetbrains.kotlinx:kotlinx-serialization-json",
    version.ref = "kotlinxSerializationJson" }
50     androidx-navigation-compose = { group = "androidx.navigation", name = "navigation-compose",
    version.ref = "navigation" }
51     androidx-hilt-navigation-compose = { group = "androidx.hilt", name = "hilt-navigation-
    compose", version.ref = "hiltNavigation" }
52     androidx-hilt-lifecycle-viewmodel-compose = { group = "androidx.hilt", name = "hilt-
    lifecycle-viewmodel-compose", version.ref = "hiltNavigation" }
53     timber = { module = "com.jakewharton.timber:timber", version.ref = "timber" }
54
55     # Yandex
56     yandex-oauth = { group = "com.yandex.android", name = "authsdk", version.ref =
    "yandexAuthSdk" }
57
58     # Hilt
59     dagger-hilt = { group = "com.google.dagger", name = "hilt-android", version.ref =
    "daggerHilt" }
60     dagger-hilt-compiler = { group = "com.google.dagger", name = "hilt-android-compiler",
    version.ref = "daggerHilt" }
61
62     # Room
63     room-runtime = { group = "androidx.room", name = "room-runtime", version.ref = "room" }
64     room-ktx = { group = "androidx.room", name = "room-ktx", version.ref = "room" }
65     room-compiler = { group = "androidx.room", name = "room-compiler", version.ref = "room" }
66
67     # Retrofit
68     retrofit = { group = "com.squareup.retrofit2", name = "retrofit", version.ref = "retrofit" }
69     retrofit-converter-scalars = { group = "com.squareup.retrofit2", name = "converter-scalars",
    version.ref = "retrofit" }
70     retrofit-converter-jackson = { group = "com.squareup.retrofit2", name = "converter-jackson",
    version.ref = "retrofit" }
71     okhttp3 = { module = "com.squareup.okhttp3:okhttp", version.ref = "okhttp" }
72     androidx-work-runtime-ktx = { group = "androidx.work", name = "work-runtime-ktx",
    version.ref = "workRuntime" }
73     androidx-hilt-work = { group = "androidx.hilt", name = "hilt-work", version.ref =
    "hiltWork" }
74     androidx-hilt-compiler = { group = "androidx.hilt", name = "hilt-compiler", version.ref =
    "hiltWork" }
75
76     androidx-core-ktx = { group = "androidx.core", name = "core-ktx", version.ref = "coreKtx" }

```

```

77     androidx-appcompat = { group = "androidx.appcompat", name = "appcompat", version.ref =
    "appcompat" }
78     androidx-datastore-preferences = { group = "androidx.datastore", name = "datastore-
    preferences", version.ref = "datastore" }
79     junit = { group = "junit", name = "junit", version.ref = "junit" }
80     androidx-junit = { group = "androidx.test.ext", name = "junit", version.ref =
    "junitVersion" }
81     androidx-test-runner = { group = "androidx.test", name = "runner", version.ref =
    "androidxTestCore" }
82     androidx-test-core = { group = "androidx.test", name = "core", version.ref =
    "androidxTestCore" }
83     androidx-espresso-core = { group = "androidx.test.espresso", name = "espresso-core",
    version.ref = "espressoCore" }
84     androidx-lifecycle-runtime-ktx = { group = "androidx.lifecycle", name = "lifecycle-runtime-
    ktx", version.ref = "lifecycleRuntimeKtx" }
85     androidx-activity-compose = { group = "androidx.activity", name = "activity-compose",
    version.ref = "activityCompose" }
86     androidx-compose-bom = { group = "androidx.compose", name = "compose-bom", version.ref =
    "composeBom" }
87     androidx-material3 = { group = "androidx.compose.material3", name = "material3" }
88     androidx-material-icons-extended = { group = "androidx.compose.material", name = "material-
    icons-extended" }
89     androidx-camera-core = { group = "androidx.camera", name = "camera-core", version.ref =
    "cameraX" }
90     androidx-camera-camera2 = { group = "androidx.camera", name = "camera-camera2", version.ref
    = "cameraX" }
91     androidx-camera-lifecycle = { group = "androidx.camera", name = "camera-lifecycle",
    version.ref = "cameraX" }
92     androidx-camera-view = { group = "androidx.camera", name = "camera-view", version.ref =
    "cameraX" }
93     zxing-core = { group = "com.google.zxing", name = "core", version.ref = "zxing" }
94     java-otp = { group = "com.eatthepath", name = "java-otp", version.ref = "javaOtp" }
95
96     androidx-ui = { group = "androidx.compose.ui", name = "ui" }
97     androidx-ui-graphics = { group = "androidx.compose.ui", name = "ui-graphics" }
98     androidx-ui-tooling = { group = "androidx.compose.ui", name = "ui-tooling" }
99     androidx-ui-tooling-preview = { group = "androidx.compose.ui", name = "ui-tooling-preview" }
100    androidx-ui-test-manifest = { group = "androidx.compose.ui", name = "ui-test-manifest" }
101    androidx-ui-test-junit4 = { group = "androidx.compose.ui", name = "ui-test-junit4" }
102
103
104    [plugins]
105    android-application = { id = "com.android.application", version.ref = "agp" }
106    kotlin-android = { id = "org.jetbrains.kotlin.android", version.ref = "kotlin" }
107    compose-compiler = { id = "org.jetbrains.kotlin.plugin.compose", version.ref = "kotlin" }
108    jetbrains-kotlin-serialization = { id = "org.jetbrains.kotlin.plugin.serialization",
    version.ref = "kotlin" }
109    jetbrains-kotlin-jvm = { id = "org.jetbrains.kotlin.jvm", version.ref = "kotlin" }
110    dagger-hilt = { id = "com.google.dagger.hilt.android", version.ref = "daggerHilt" }

```

```
111     ksp = { id = "com.google.devtools.ksp", version.ref = "ksp" }
112     android-library = { id = "com.android.library", version.ref = "agp" }
113     kotlin-parcelize = { id = "kotlin-parcelize" }
114
```

Исходный файл gradle/wrapper/gradle-wrapper.properties

```
1  #Sat Sep 07 01:04:14 MSK 2024
2  distributionBase=GRADLE_USER_HOME
3  distributionPath=wrapper/dists
4  distributionUrl=https\://services.gradle.org/distributions/gradle-9.5.1-bin.zip
5  zipStoreBase=GRADLE_USER_HOME
6  zipStorePath=wrapper/dists
7
```

Исходный файл gradlew

```
1      #!/usr/bin/env sh
2
3      #
4      # Copyright 2015 the original author or authors.
5      #
6      # Licensed under the Apache License, Version 2.0 (the "License");
7      # you may not use this file except in compliance with the License.
8      # You may obtain a copy of the License at
9      #
10     #      https://www.apache.org/licenses/LICENSE-2.0
11     #
12     # Unless required by applicable law or agreed to in writing, software
13     # distributed under the License is distributed on an "AS IS" BASIS,
14     # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15     # See the License for the specific language governing permissions and
16     # limitations under the License.
17     #
18
19     #####
20     ##
21     ## Gradle start up script for UN*X
22     ##
23     #####
24
25     # Attempt to set APP_HOME
26     # Resolve links: $0 may be a link
27     PRG="$0"
28     # Need this for relative symlinks.
29     while [ -h "$PRG" ] ; do
30         ls=`ls -ld "$PRG"`
31         link=`expr "$ls" : '.*-> \(.*\)$'`
32         if expr "$link" : '/.*' > /dev/null; then
33             PRG="$link"
34         else
35             PRG=`dirname "$PRG"`"/$link"
36         fi
37     done
38     SAVED=""`pwd` "
39     cd "`dirname \"$PRG\"`/" >/dev/null
40     APP_HOME=""`pwd` -P` "
41     cd "$SAVED" >/dev/null
42
43     APP_NAME="Gradle"
```

```

44 APP_BASE_NAME=`basename "$0"`
45
46 # Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to pass JVM
47 # options to this script.
48 DEFAULT_JVM_OPTS="-Xmx64m" "-Xms64m"
49
50 # Use the maximum available, or set MAX_FD != -1 to use that value.
51 MAX_FD="maximum"
52
53 warn () {
54     echo "$*"
55 }
56
57 die () {
58     echo
59     echo "$*"
60     echo
61     exit 1
62 }
63
64 # OS specific support (must be 'true' or 'false').
65 cygwin=false
66 msys=false
67 darwin=false
68 nonstop=false
69 case "`uname`" in
70     CYGWIN* )
71         cygwin=true
72         ;;
73     Darwin* )
74         darwin=true
75         ;;
76     MINGW* )
77         msys=true
78         ;;
79     NONSTOP* )
80         nonstop=true
81         ;;
82 esac
83
84 CLASSPATH=$APP_HOME/gradle/wrapper/gradle-wrapper.jar
85
86 # Determine the Java command to use to start the JVM.
87 if [ -n "$JAVA_HOME" ] ; then
88     if [ -x "$JAVA_HOME/jre/sh/java" ] ; then

```



```

89         # IBM's JDK on AIX uses strange locations for the executables
90         JAVACMD="$JAVA_HOME/jre/sh/java"
91     else
92         JAVACMD="$JAVA_HOME/bin/java"
93     fi
94     if [ ! -x "$JAVACMD" ] ; then
95         die "ERROR: JAVA_HOME is set to an invalid directory: $JAVA_HOME
96
97     Please set the JAVA_HOME variable in your environment to match the
98     location of your Java installation."
99     fi
100 else
101     JAVACMD="java"
102     which java >/dev/null 2>&1 || die "ERROR: JAVA_HOME is not set and no 'java' command
103     could be found in your PATH.
104
105     Please set the JAVA_HOME variable in your environment to match the
106     location of your Java installation."
107 fi
108
109 # Increase the maximum file descriptors if we can.
110 if [ "$cygwin" = "false" -a "$darwin" = "false" -a "$nonstop" = "false" ] ; then
111     MAX_FD_LIMIT=`ulimit -H -n`
112     if [ $? -eq 0 ] ; then
113         if [ "$MAX_FD" = "maximum" -o "$MAX_FD" = "max" ] ; then
114             MAX_FD="$MAX_FD_LIMIT"
115         fi
116         ulimit -n $MAX_FD
117         if [ $? -ne 0 ] ; then
118             warn "Could not set maximum file descriptor limit: $MAX_FD"
119         fi
120     else
121         warn "Could not query maximum file descriptor limit: $MAX_FD_LIMIT"
122     fi
123 fi
124
125 # For Darwin, add options to specify how the application appears in the dock
126 if $darwin; then
127     GRADLE_OPTS="$GRADLE_OPTS \"-Xdock:name=$APP_NAME\" \"-Xdock:icon=$APP_HOME/media/
128     gradle.icns\""
129 fi
130
131 # For Cygwin or MSYS, switch paths to Windows format before running java
132 if [ "$cygwin" = "true" -o "$msys" = "true" ] ; then
133     APP_HOME=`cygpath --path --mixed "$APP_HOME"`

```

```

132 CLASSPATH=`cygpath --path --mixed "$CLASSPATH"`
133
134 JAVACMD=`cygpath --unix "$JAVACMD"`
135
136 # We build the pattern for arguments to be converted via cygpath
137 ROOTDIRSRAW=`find -L / -maxdepth 1 -mindepth 1 -type d 2>/dev/null`
138 SEP=""
139 for dir in $ROOTDIRSRAW ; do
140     ROOTDIRS="$ROOTDIRS$SEP$dir"
141     SEP="|"
142 done
143 OURCYGPATTERN="(^($ROOTDIRS))"
144 # Add a user-defined pattern to the cygpath arguments
145 if [ "$GRADLE_CYGPATTERN" != "" ] ; then
146     OURCYGPATTERN="$OURCYGPATTERN|($GRADLE_CYGPATTERN)"
147 fi
148 # Now convert the arguments - kludge to limit ourselves to /bin/sh
149 i=0
150 for arg in "$@" ; do
151     CHECK=`echo "$arg"|egrep -c "$OURCYGPATTERN" -`
152     CHECK2=`echo "$arg"|egrep -c "^-"`           ### Determine if
an option
153
154     if [ $CHECK -ne 0 ] && [ $CHECK2 -eq 0 ] ; then        ### Added a
condition
155         eval `echo args$i`=`cygpath --path --ignore --mixed "$arg"`
156     else
157         eval `echo args$i`="\"$arg\""
158     fi
159     i=`expr $i + 1`
160 done
161 case $i in
162     0) set -- ;;
163     1) set -- "$args0" ;;
164     2) set -- "$args0" "$args1" ;;
165     3) set -- "$args0" "$args1" "$args2" ;;
166     4) set -- "$args0" "$args1" "$args2" "$args3" ;;
167     5) set -- "$args0" "$args1" "$args2" "$args3" "$args4" ;;
168     6) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" ;;
169     7) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6" ;;
170     8) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6" "$args7" ;;
171     9) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6" "$args7"
"$args8" ;;
172 esac
173 fi
174

```

```

175     # Escape application args
176     save () {
177         for i do printf %s\\n "$i" | sed "s/'/'\\\\\\'/g;ls/^/'/;\\$s/\\$/' \\\\\\\'/ " ; done
178         echo " "
179     }
180     APP_ARGS=`save "$@"`
181
182     # Collect all arguments for the java command, following the shell quoting and substitution
    rules
183     eval set -- $DEFAULT_JVM_OPTS $JAVA_OPTS $GRADLE_OPTS "\\-
Dorg.gradle.appname=$APP_BASE_NAME\\" -classpath "\\$CLASSPATH\\"
org.gradle.wrapper.GradleWrapperMain "$APP_ARGS"
184
185     exec "$JAVACMD" "$@"
186

```

Исходный файл infrastructure-android/build.gradle.kts

```
1  plugins {
2      alias(libs.plugins.android.library)
3      alias(libs.plugins.ksp)
4  }
5
6  android {
7      namespace = "com.github.nullptroma.wallenc.domain.vault.android"
8      compileSdk = 37
9
10     defaultConfig {
11         minSdk = 26
12         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
13         consumerProguardFiles("consumer-rules.pro")
14     }
15
16     buildTypes {
17         release {
18             isMinifyEnabled = false
19             proguardFiles(
20                 getDefaultProguardFile("proguard-android-optimize.txt"),
21                 "proguard-rules.pro"
22             )
23         }
24     }
25
26     compileOptions {
27         sourceCompatibility = JavaVersion.VERSION_17
28         targetCompatibility = JavaVersion.VERSION_17
29     }
30 }
31
32 dependencies {
33     implementation(project(":domain"))
34     implementation(project(":domain-vault"))
35     implementation(project(":vault-contracts"))
36
37     implementation(libs.jackson.module.kotlin)
38     implementation(libs.jackson.datatype.jsr310)
39     implementation(libs.kotlinx.coroutines.core)
40
41     implementation(libs.room.ktx)
42     implementation(libs.room.runtime)
43     ksp(libs.room.compiler)
```

```
44
45     implementation(libs.androidx.core.ktx)
46     testImplementation(libs.junit)
47     androidTestImplementation(libs.androidx.junit)
48     androidTestImplementation(libs.androidx.test.runner)
49     androidTestImplementation(libs.androidx.test.core)
50     androidTestImplementation(libs.room.testing)
51 }
52
```

Исходный файл settings.gradle.kts

```
1  pluginManagement {
2      repositories {
3          google {
4              content {
5                  includeGroupByRegex("com\\.\\.android.*")
6                  includeGroupByRegex("com\\.\\.google.*")
7                  includeGroupByRegex("androidx.*")
8              }
9          }
10         mavenCentral()
11         gradlePluginPortal()
12     }
13 }
14 plugins {
15     id("org.gradle.toolchains.foojay-resolver-convention") version "1.0.0"
16 }
17 dependencyResolutionManagement {
18     repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
19     repositories {
20         google()
21         mavenCentral()
22     }
23 }
24
25 rootProject.name = "Wallenc"
26 include(":app")
27 include(":domain")
28 include(":usecases")
29 include(":ui")
30 include(":domain-vault")
31 include(":infrastructure-android")
32 include(":vault-contracts")
33 include(":task-runtime")
34
```

Исходный файл task-runtime/build.gradle.kts

```
1  plugins {
2      id("java-library")
3      alias(libs.plugins.jetbrains.kotlin.jvm)
4  }
5
6  java {
7      toolchain {
8          languageVersion = JavaLanguageVersion.of(17)
9      }
10 }
11
12 kotlin {
13     jvmToolchain(17)
14 }
15
16 dependencies {
17     implementation(project(":domain"))
18     implementation(libs.kotlinx.coroutines.core)
19     testImplementation(libs.junit)
20     testImplementation(libs.kotlinx.coroutines.test)
21 }
22
```

Исходный файл ui/build.gradle.kts

```
1  plugins {
2      alias(libs.plugins.android.library)
3      alias(libs.plugins.compose.compiler)
4      alias(libs.plugins.dagger.hilt)
5      alias(libs.plugins.jetbrains.kotlin.serialization)
6      alias(libs.plugins.kotlin.parcelize)
7      alias(libs.plugins.ksp)
8  }
9
10 android {
11     namespace = "com.github.nullptroma.wallenc.ui"
12     compileSdk = 37
13
14     defaultConfig {
15         minSdk = 24
16
17         testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
18         consumerProguardFiles("consumer-rules.pro")
19     }
20
21     buildTypes {
22         release {
23             isMinifyEnabled = false
24             proguardFiles(
25                 getDefaultProguardFile("proguard-android-optimize.txt"),
26                 "proguard-rules.pro"
27             )
28         }
29     }
30     compileOptions {
31         sourceCompatibility = JavaVersion.VERSION_17
32         targetCompatibility = JavaVersion.VERSION_17
33     }
34
35     buildFeatures {
36         compose = true
37     }
38
39     testOptions {
40         unitTests.isIncludeAndroidResources = true
41     }
42
43     packaging {
```



```

44         resources {
45             excludes += "/META-INF/{AL2.0,LGPL2.1}"
46         }
47     }
48 }
49
50 kotlin {
51     jvmToolchain(17)
52 }
53
54
55 dependencies {
56     implementation(libs.androidx.navigation.compose)
57     implementation(libs.androidx.hilt.navigation.compose)
58     implementation(libs.androidx.hilt.lifecycle.viewmodel.compose)
59
60     // Hilt
61     implementation(libs.dagger.hilt)
62     ksp(libs.dagger.hilt.compiler)
63
64     implementation(libs.kotlinx.serialization.json)
65     implementation(libs.kotlin.reflect)
66
67     implementation(libs.androidx.core.ktx)
68     implementation(libs.androidx.lifecycle.runtime.ktx)
69     implementation(libs.androidx.activity.compose)
70     implementation(platform(libs.androidx.compose.bom))
71
72     debugImplementation(libs.androidx.ui.tooling)
73     debugImplementation(libs.androidx.ui.test.manifest)
74     implementation(libs.androidx.ui)
75     implementation(libs.androidx.ui.graphics)
76     implementation(libs.androidx.ui.tooling.preview)
77     implementation(libs.androidx.material3)
78     implementation(libs.androidx.material.icons.extended)
79     implementation(libs.androidx.camera.core)
80     implementation(libs.androidx.camera.camera2)
81     implementation(libs.androidx.camera.lifecycle)
82     implementation(libs.androidx.camera.view)
83     implementation(libs.zxing.core)
84
85     testImplementation(libs.junit)
86     testImplementation(libs.kotlinx.coroutines.test)
87     testImplementation(libs.androidx.arch.core.testing)
88     testImplementation(libs.robolectric)

```

```
89     testImplementation(libs.mockk)
90     androidTestImplementation(libs.androidx.junit)
91     androidTestImplementation(libs.androidx.espresso.core)
92     androidTestImplementation(platform(libs.androidx.compose.bom))
93     androidTestImplementation(libs.androidx.ui.test.junit4)
94
95     implementation(project(":domain"))
96     implementation(project(":usecases"))
97     implementation(project(":vault-contracts"))
98
99     testImplementation(project(":task-runtime"))
100 }
```

Исходный файл usecases/build.gradle.kts

```
1  plugins {
2      id("java-library")
3      alias(libs.plugins.jetbrains.kotlin.jvm)
4  }
5
6  java {
7      toolchain {
8          languageVersion = JavaLanguageVersion.of(17)
9      }
10 }
11
12 kotlin {
13     jvmToolchain(17)
14 }
15
16 dependencies {
17     compileOnly(libs.javax.inject)
18     implementation(project(":domain"))
19     implementation(libs.kotlinx.coroutines.core)
20     implementation(libs.kotlinx.serialization.json)
21     implementation(libs.java otp)
22     testImplementation(libs.junit)
23     testImplementation(libs.kotlinx.coroutines.test)
24 }
25
```

Исходный файл vault-contracts/build.gradle.kts

```
1  import org.gradle.api.file.DuplicatesStrategy
2  import org.gradle.api.tasks.bundling.Jar
3
4  plugins {
5      id("java-library")
6      alias(libs.plugins.jetbrains.kotlin.jvm)
7  }
8
9  java {
10     toolchain {
11         languageVersion = JavaLanguageVersion.of(17)
12     }
13 }
14
15 tasks.withType<Jar>().configureEach {
16     duplicatesStrategy = DuplicatesStrategy.EXCLUDE
17 }
18
19 dependencies {
20     implementation(project(":domain"))
21     implementation(libs.kotlinx.coroutines.core)
22     testImplementation(libs.junit)
23 }
24
```

ПРИЛОЖЕНИЕ А.2

Модуль :app

Исходный файл app/proguard-rules.pro

```
1  # Add project specific ProGuard rules here.
2  # You can control the set of applied configuration files using the
3  # proguardFiles setting in build.gradle.
4  #
5  # For more details, see
6  #   http://developer.android.com/guide/developing/tools/proguard.html
7
8  # If your project uses WebView with JS, uncomment the following
9  # and specify the fully qualified class name to the JavaScript interface
10 # class:
11 #-keepclassmembers class fqcn.of.javascript.interface.for.webview {
12 #   public *;
13 #}
14
15 # Uncomment this to preserve the line number information for
16 # debugging stack traces.
17 #-keepattributes SourceFile,LineNumberTable
18
19 # If you keep the line number information, uncomment this to
20 # hide the original source file name.
21 #-renamesourcefileattribute SourceFile
```

Исходный файл app/src/androidTest/java/com/github/nullptroma/wallenc/app/
integration/yandex/YandexDiskLiveIntegrationTest.kt

```
1 package com.github.nullptroma.wallenc.app.integration.yandex
2
3 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskApiFactory
4 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository.YandexDiskRepository
5 import kotlinx.coroutines.Dispatchers
6 import kotlinx.coroutines.runBlocking
7 import org.junit.After
8 import org.junit.Assert.assertNotNull
9 import org.junit.Assert.assertTrue
10 import org.junit.Before
11 import org.junit.Test
12 import org.junit.runner.RunWith
13 import org.junit.runners.JUnit4
14
15 /**
16  * Live-прогон Disk API. Пути только `app:/` – как в [YandexVault]; токен из Auth SDK
17  * обычно имеет `cloud_api:disk.app_folder`, без полного доступа к `disk:/` (там 403).
18  */
19 @RunWith(JUnit4::class)
20 class YandexDiskLiveIntegrationTest {
21
22     private lateinit var repository: YandexDiskRepository
23     private val testFolder = "app:/wallenc-integration-test"
24     private val probeFileName = "wallenc-probe.txt"
25     private val probePath = "$testFolder/$probeFileName"
26     private val probePayload = "wallenc-integration-probe".encodeToByteArray()
27
28     @Before
29     fun setUp() {
30         YandexTestCredentials.assumePresent()
31         val token = YandexTestCredentials.oauthToken()!!
32         repository = YandexDiskApiFactory.createRepositoryWithToken(token, Dispatchers.IO)
33         runBlocking {
34             runCatching { repository.createFolder(testFolder) }
35             runCatching {
36                 repository.uploadBytes(probePath, probePayload, overwrite = true)
37             }
38         }
39     }
40
41     @After
```

```

42     fun tearDown(): Unit = runBlocking {
43         runCatching { repository.delete(probePath, permanently = true) }
44     }
45
46     @Test
47     fun diskInfoReturnsQuota() = runBlocking {
48         val info = repository.diskInfo()
49         assertNotNull(info.totalSpace)
50         assertTrue(info.totalSpace!! > 0)
51     }
52
53     @Test
54     fun listTestFolderDoesNotThrow() = runBlocking {
55         val result = repository.list(testFolder, limit = 10, offset = 0)
56         assertNotNull(result)
57     }
58
59     @Test
60     fun uploadAndDownloadRoundTrip() = runBlocking {
61         val path = "$testFolder/roundtrip-${System.currentTimeMillis()}.bin"
62         try {
63             repository.uploadBytes(path, probePayload, overwrite = true)
64             val downloaded = repository.openDownloadStream(path).use { it.readBytes() }
65             assertTrue(downloaded.contentEquals(probePayload))
66         } finally {
67             runCatching { repository.delete(path, permanently = true) }
68         }
69     }
70 }
71

```

Исходный файл app/src/androidTest/java/com/github/nullptroma/wallenc/app/
integration/yandex/YandexTestCredentials.kt

```
1 package com.github.nullptroma.wallenc.app.integration.yandex
2
3 import androidx.test.platform.app.InstrumentationRegistry
4 import org.junit.Assume.assumeFalse
5
6 object YandexTestCredentials {
7     fun oauthToken(): String? =
8         InstrumentationRegistry.getArguments().getString("yandex.oauth.token")?.takeIf
9         { it.isNotBlank() }
10
11     fun assumePresent(message: String = "Добавьте yandex.test.oauth.token в
12     local.properties") {
13         assumeFalse(message, oauthToken().isNullOrBlank())
14     }
15 }
```


Исходный файл app/src/main/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3          xmlns:tools="http://schemas.android.com/tools">
4
5      <uses-permission android:name="android.permission.INTERNET" />
6      <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
7      <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
8      <uses-permission android:name="android.permission.FOREGROUND_SERVICE_DATA_SYNC" />
9      <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
10     <uses-permission android:name="android.permission.CAMERA" />
11     <uses-feature
12         android:name="android.hardware.camera"
13         android:required="false" />
14
15     <application
16         android:allowBackup="true"
17         android:dataExtractionRules="@xml/data_extraction_rules"
18         android:fullBackupContent="@xml/backup_rules"
19         android:icon="@mipmap/ic_launcher"
20         android:label="@string/app_name"
21         android:name=".WallencApplication"
22         android:roundIcon="@mipmap/ic_launcher_round"
23         android:supportsRtl="true"
24         android:theme="@style/Theme.Wallenc"
25         android:localeConfig="@xml/locales_config"
26         android:enableOnBackInvokedCallback="true"
27         tools:targetApi="37">
28         <activity
29             android:name=".MainActivity"
30             android:exported="true"
31             android:windowSoftInputMode="adjustNothing"
32             android:theme="@style/Theme.Wallenc">
33             <intent-filter>
34                 <action android:name="android.intent.action.MAIN" />
35                 <category android:name="android.intent.category.LAUNCHER" />
36             </intent-filter>
37             <intent-filter>
38                 <action android:name="android.intent.action.VIEW" />
39                 <category android:name="android.intent.category.DEFAULT" />
40                 <category android:name="android.intent.category.BROWSABLE" />
41                 <data android:scheme="wallenc" />
42                 <data android:host="main" />
43             </intent-filter>
```

```

44         <intent-filter>
45             <action android:name="android.intent.action.VIEW" />
46             <category android:name="android.intent.category.DEFAULT" />
47             <category android:name="android.intent.category.BROWSABLE" />
48             <data android:scheme="wallenc" />
49             <data android:host="tasks" />
50         </intent-filter>
51         <intent-filter>
52             <action android:name="android.intent.action.VIEW" />
53             <category android:name="android.intent.category.DEFAULT" />
54             <category android:name="android.intent.category.BROWSABLE" />
55             <data android:scheme="wallenc" />
56             <data android:host="settings" />
57         </intent-filter>
58     </activity>
59
60     <provider
61         android:name="androidx.startup.InitializationProvider"
62         android:authorities="${applicationId}.androidx-startup"
63         android:exported="false"
64         tools:node="merge">
65         <meta-data
66             android:name="androidx.work.WorkManagerInitializer"
67             android:value="androidx.startup"
68             tools:node="remove" />
69     </provider>
70
71     <service
72         android:name=".tasks.TaskPipelineForegroundService"
73         android:exported="false"
74         android:foregroundServiceType="dataSync"
75         android:description="@string/fgs_task_pipeline_description" />
76
77     <receiver
78         android:name=".sync.StorageSyncBootReceiver"
79         android:exported="false">
80         <intent-filter>
81             <action android:name="android.intent.action.BOOT_COMPLETED" />
82         </intent-filter>
83     </receiver>
84 </application>
85
86 </manifest>

```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/Logger.kt

```
1 package com.github.nullptroma.wallenc.app
2
3 import com.github.nullptroma.wallenc.domain.interfaces.ILogger
4 import timber.log.Timber
5
6 class Logger: ILogger {
7     override fun debug(tag: String, msg: String) {
8         Timber.tag(tag)
9         Timber.d(msg)
10    }
11 }
12
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/ MainActivity.kt

```
1 package com.github.nullptroma.wallenc.app
2
3 import android.Manifest
4 import android.content.Intent
5 import android.content.pm.PackageManager
6 import android.os.Build
7 import android.os.Bundle
8 import androidx.activity.compose.setContent
9 import androidx.activity.enableEdgeToEdge
10 import androidx.appcompat.app.AppCompatActivity
11 import androidx.compose.runtime.getValue
12 import androidx.compose.runtime.mutableIntStateOf
13 import androidx.compose.runtime.mutableStateOf
14 import androidx.compose.runtime.setValue
15 import androidx.core.app.ActivityCompat
16 import androidx.core.content.ContextCompat
17 import com.github.nullptroma.wallenc.app.auth.YandexSignInService
18 import com.github.nullptroma.wallenc.ui.WallencUi
19 import com.github.nullptroma.wallenc.ui.navigation.matchesWallencDeepLink
20 import dagger.hilt.android.AndroidEntryPoint
21 import timber.log.Timber
22 import javax.inject.Inject
23
24
25 @AndroidEntryPoint
26 class MainActivity : AppCompatActivity() {
27
28     @Inject
29     lateinit var yandexSignInService: YandexSignInService
30
31     private val deepLinkPulse = mutableIntStateOf(0)
32     private var deepLinkIntentForNav: Intent by mutableStateOf(Intent())
33
34     override fun onCreate(savedInstanceState: Bundle?) {
35         super.onCreate(savedInstanceState)
36         yandexSignInService.registerWith(this)
37         enableEdgeToEdge()
38         requestNotificationPermissionIfNeeded()
39
40         Timber.plant(Timber.DebugTree())
41
42         deepLinkIntentForNav = intent
```

```

43         if (intent.matchesWallencDeepLink()) {
44             deepLinkPulse.intValue++
45         }
46
47         setContent {
48             val pulse by deepLinkPulse
49             WallencUi(
50                 deepLinkPulse = pulse,
51                 deepLinkIntent = deepLinkIntentForNav,
52             )
53         }
54     }
55
56     override fun onNewIntent(intent: Intent) {
57         super.onNewIntent(intent)
58         setIntent(intent)
59         deepLinkIntentForNav = intent
60         if (intent.matchesWallencDeepLink()) {
61             deepLinkPulse.intValue++
62         }
63     }
64
65     override fun onDestroy() {
66         yandexSignInService.unregister(this)
67         super.onDestroy()
68     }
69
70     private fun requestNotificationPermissionIfNeeded() {
71         if (Build.VERSION.SDK_INT < Build.VERSION_CODES.TIRAMISU) return
72         val granted = ContextCompat.checkSelfPermission(
73             this,
74             Manifest.permission.POST_NOTIFICATIONS,
75         ) == PackageManager.PERMISSION_GRANTED
76         if (granted) return
77         ActivityCompat.requestPermissions(
78             this,
79             arrayOf(Manifest.permission.POST_NOTIFICATIONS),
80             NOTIFICATION_PERMISSION_REQUEST_CODE,
81         )
82     }
83
84     companion object {
85         private const val NOTIFICATION_PERMISSION_REQUEST_CODE = 100
86     }
87 }

```


Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/
WallencApplication.kt

```
1 package com.github.nullptroma.wallenc.app
2
3 import android.app.Application
4 import android.content.Context
5 import androidx.work.Configuration
6 import com.github.nullptroma.wallenc.app.di.HiltWorkerFactoryEntryPoint
7 import com.github.nullptroma.wallenc.app.locale.AppLocaleStorage
8 import com.github.nullptroma.wallenc.app.sync.StorageSyncBootstrap
9 import com.github.nullptroma.wallenc.app.tasks.TaskPipelineForegroundBootstrap
10 import dagger.hilt.android.EntryPointAccessors
11 import dagger.hilt.android.HiltAndroidApp
12 import javax.inject.Inject
13
14 @HiltAndroidApp
15 class WallencApplication : Application(), Configuration.Provider {
16
17     @Inject
18     lateinit var taskPipelineForegroundBootstrap: TaskPipelineForegroundBootstrap
19
20     @Inject
21     lateinit var storageSyncBootstrap: StorageSyncBootstrap
22
23     override fun attachBaseContext(base: Context) {
24         AppLocaleStorage.applyStored(base)
25         super.attachBaseContext(base)
26     }
27
28     override fun onCreate() {
29         super.onCreate()
30         AppLocaleStorage.migrateLegacyDataStoreIfNeeded(this)
31         AppLocaleStorage.applyStored(this)
32         taskPipelineForegroundBootstrap.start()
33         storageSyncBootstrap.start()
34     }
35
36     override val workManagerConfiguration: Configuration
37         get() {
38             val factory = EntryPointAccessors.fromApplication(
39                 applicationContext,
40                 HiltWorkerFactoryEntryPoint::class.java,
41             ).hiltWorkerFactory()
42             return Configuration.Builder()
```

```
43         .setWorkerFactory(factory)
44         .build()
45     }
46 }
47
```


Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/auth/ YandexSignInService.kt

```
1 package com.github.nullptroma.wallenc.app.auth
2
3 import android.content.Context
4 import androidx.activity.ComponentActivity
5 import androidx.activity.result.ActivityResultLauncher
6 import com.github.nullptroma.wallenc.domain.vault.vaults.yandex.YandexRegistration
7 import com.github.nullptroma.wallenc.vault.contract.CloudBrand
8 import com.github.nullptroma.wallenc.vault.contract.RemoteVaultAuthenticator
9 import com.github.nullptroma.wallenc.vault.contract.VaultLinkFailure
10 import com.github.nullptroma.wallenc.vault.contract.VaultLinkOutcome
11 import com.yandex.authsdk.YandexAuthLoginOptions
12 import com.yandex.authsdk.YandexAuthOptions
13 import com.yandex.authsdk.YandexAuthResult
14 import com.yandex.authsdk.YandexAuthSdk
15 import com.yandex.authsdk.internal.strategy.LoginType
16 import dagger.hilt.android.qualifiers.ApplicationContext
17 import javax.inject.Inject
18 import javax.inject.Singleton
19
20 /**
21  * Императивный вход через Yandex Auth SDK: [registerWith] в [ComponentActivity.onCreate],
22  * [unregister] в [ComponentActivity.onDestroy], чтобы синглтон не держал
23  * [ActivityResultLauncher]
24  *
25  * дольше жизни Activity.
26  *
27  * При смене конфигурации новая Activity может вызвать [registerWith] до [unregister] старой
28  * —
29  * тогда владелец и launcher обновляются; [unregister] для уже неактуального экземпляра —
30  * по-ор.
31  *
32  * Реализует [RemoteVaultAuthenticator] из `:vault-api`: presentation про Yandex SDK
33  * ничего не знает, а получает обобщённый [VaultLinkOutcome] с готовой
34  * `VaultRegistration` внутри.
35  */
36 @Singleton
37 class YandexSignInService @Inject constructor(
38     @ApplicationContext appContext: Context,
39 ) : RemoteVaultAuthenticator {
40
41     private val sdk = YandexAuthSdk.create(YandexAuthOptions(appContext, true))
42
43     private var launcher: ActivityResultLauncher<YandexAuthLoginOptions>? = null
```

```

41     private var registrationOwner: ComponentActivity? = null
42
43     @Volatile
44     private var pending: ((VaultLinkOutcome) -> Unit)? = null
45
46     fun registerWith(activity: ComponentActivity) {
47         if (registrationOwner === activity && launcher != null) return
48         launcher = activity.registerForActivityResult(sdk.contract) { result ->
49             val mapped = mapYandexResult(result)
50             val cb = synchronized(this) {
51                 val p = pending
52                 pending = null
53                 p
54             }
55             cb?.invoke(mapped)
56         }
57         registrationOwner = activity
58     }
59
60     fun unregister(activity: ComponentActivity) {
61         if (registrationOwner != activity) return
62         launcher = null
63         registrationOwner = null
64         val cb = synchronized(this) {
65             val p = pending
66             pending = null
67             p
68         }
69         cb?.invoke(VaultLinkOutcome.Cancelled)
70     }
71
72     override fun beginLink(brand: CloudBrand, onResult: (VaultLinkOutcome) -> Unit) {
73         if (brand != CloudBrand.YANDEX) {
74             onResult(VaultLinkOutcome.Failed(VaultLinkFailure.UnsupportedBrand))
75             return
76         }
77         val l = launcher
78         if (l == null) {
79             onResult(VaultLinkOutcome.Failed(VaultLinkFailure.NotRegistered))
80             return
81         }
82         synchronized(this) { pending = onResult }
83         l.launch(YandexAuthLoginOptions(LoginType.NATIVE))
84     }
85

```

```
86     private fun mapYandexResult(result: YandexAuthResult): VaultLinkOutcome = when (result)
87     {
88         is YandexAuthResult.Success ->
89             VaultLinkOutcome.Success(YandexRegistration(result.token.value))
90         is YandexAuthResult.Failure ->
91             VaultLinkOutcome.Failed(VaultLinkFailure.AuthError)
92         YandexAuthResult.Cancelled -> VaultLinkOutcome.Cancelled
93     }
94 }
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
HiltWorkerFactoryEntryPoint.kt

```
1  package com.github.nullptroma.wallenc.app.di
2
3  import androidx.hilt.work.HiltWorkerFactory
4  import dagger.hilt.EntryPoint
5  import dagger.hilt.InstallIn
6  import dagger.hilt.components.SingletonComponent
7
8  /**
9   * WorkManager инициализируется до [android.app.Application.onCreate], поэтому
10   * нельзя читать `@Inject lateinit var workerFactory` в [Configuration.Provider].
11   * Фабрика берётся через EntryPoint.
12   */
13  @EntryPoint
14  @InstallIn(SingletonComponent::class)
15  interface HiltWorkerFactoryEntryPoint {
16      fun hiltWorkerFactory(): HiltWorkerFactory
17  }
18
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
StorageSyncBootEntryPoint.kt

```
1 package com.github.nullptroma.wallenc.app.di
2
3 import com.github.nullptroma.wallenc.app.sync.StorageSyncScheduler
4 import dagger.hilt.EntryPoint
5 import dagger.hilt.InstallIn
6 import dagger.hilt.components.SingletonComponent
7
8 @EntryPoint
9 @InstallIn(SingletonComponent::class)
10 interface StorageSyncBootEntryPoint {
11     fun storageSyncScheduler(): StorageSyncScheduler
12 }
13
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/app/DispatchersModule.kt

```
1  package com.github.nullptroma.wallenc.app.di.modules.app
2
3  import dagger.Module
4  import dagger.Provides
5  import dagger.hilt.InstallIn
6  import dagger.hilt.components.SingletonComponent
7  import kotlinx.coroutines.CoroutineDispatcher
8  import kotlinx.coroutines.Dispatchers
9  import javax.inject.Qualifier
10 import javax.inject.Singleton
11
12 @Qualifier
13 @Retention(AnnotationRetention.BINARY)
14 annotation class MainDispatcher
15
16 @Qualifier
17 @Retention(AnnotationRetention.BINARY)
18 annotation class IoDispatcher
19
20 @Module
21 @InstallIn(SingletonComponent::class)
22 class DispatchersModule {
23     @MainDispatcher
24     @Provides
25     @Singleton
26     fun providesMainDispatcher(): CoroutineDispatcher = Dispatchers.Main
27
28     @IoDispatcher
29     @Provides
30     @Singleton
31     fun providesIoDispatcher(): CoroutineDispatcher = Dispatchers.IO
32 }
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/app/LocaleModule.kt

```
1  package com.github.nullptroma.wallenc.app.di.modules.app
2
3  import com.github.nullptroma.wallenc.app.locale.AppLocaleControllerImpl
4  import com.github.nullptroma.wallenc.ui.locale.AppLocaleController
5  import dagger.Binds
6  import dagger.Module
7  import dagger.hilt.InstallIn
8  import dagger.hilt.components.SingletonComponent
9  import javax.inject.Singleton
10
11  @Module
12  @InstallIn(SingletonComponent::class)
13  abstract class LocaleModule {
14
15      @Binds
16      @Singleton
17      abstract fun bindAppLocaleController(impl: AppLocaleControllerImpl): AppLocaleController
18  }
19
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/app/LoggerModule.kt

```
1  package com.github.nullptroma.wallenc.app.di.modules.app
2
3  import com.github.nullptroma.wallenc.app.Logger
4  import com.github.nullptroma.wallenc.domain.interfaces.ILogger
5  import dagger.Module
6  import dagger.Provides
7  import dagger.hilt.InstallIn
8  import dagger.hilt.components.SingletonComponent
9  import javax.inject.Singleton
10
11  @Module
12  @InstallIn(SingletonComponent::class)
13  object LoggerModule {
14      @Provides
15      @Singleton
16      fun provideLogger(): ILogger = Logger()
17  }
18
```


Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/app/UiStringModule.kt

```
1 package com.github.nullptroma.wallenc.app.di.modules.app
2
3 import android.content.Context
4 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
5 import dagger.Module
6 import dagger.Provides
7 import dagger.hilt.InstallIn
8 import dagger.hilt.android.qualifiers.ApplicationContext
9 import dagger.hilt.components.SingletonComponent
10 import javax.inject.Singleton
11
12 @Module
13 @InstallIn(SingletonComponent::class)
14 object UiStringModule {
15
16     @Provides
17     @Singleton
18     fun provideUiStringResolver(@ApplicationContext context: Context): UiStringResolver =
19         object : UiStringResolver {
20             override fun invoke(id: Int, vararg formatArgs: Any): String =
21                 if (formatArgs.isEmpty()) {
22                     context.getString(id)
23                 } else {
24                     context.getString(id, *formatArgs)
25                 }
26
27             override fun plurals(id: Int, quantity: Int, vararg formatArgs: Any): String =
28                 if (formatArgs.isEmpty()) {
29                     context.resources.getQuantityString(id, quantity)
30                 } else {
31                     context.resources.getQuantityString(id, quantity, *formatArgs)
32                 }
33         }
34 }
35
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/auth/YandexAuthModule.kt

```
1 package com.github.nullptroma.wallenc.app.di.modules.auth
2
3 import com.github.nullptroma.wallenc.app.auth.YandexSignInService
4 import com.github.nullptroma.wallenc.vault.contract.RemoteVaultAuthenticator
5 import dagger.Binds
6 import dagger.Module
7 import dagger.hilt.InstallIn
8 import dagger.hilt.components.SingletonComponent
9 import javax.inject.Singleton
10
11 @Module
12 @InstallIn(SingletonComponent::class)
13 abstract class YandexAuthModule {
14
15     @Binds
16     @Singleton
17     abstract fun bindRemoteVaultAuthenticator(
18         impl: YandexSignInService,
19     ): RemoteVaultAuthenticator
20 }
21
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/data/NetworkModule.kt

```
1 package com.github.nullptroma.wallenc.app.di.modules.data
2
3 import com.github.nullptroma.wallenc.app.di.modules.app.IoDispatcher
4 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskApiFactory
5 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository.YandexDiskRepository
6 import com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo.YandexUserInfoApi
7 import com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo.YandexUserInfoApiFactory
8 import com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo.repository.YandexUserInfoRepository
9 import com.github.nullptroma.wallenc.domain.vault.ports.YandexAccountStore
10 import dagger.Module
11 import dagger.Provides
12 import dagger.hilt.InstallIn
13 import dagger.hilt.components.SingletonComponent
14 import kotlinx.coroutines.CoroutineDispatcher
15 import javax.inject.Singleton
16
17 @Module
18 @InstallIn(SingletonComponent::class)
19 object NetworkModule {
20
21     @Provides
22     @Singleton
23     fun provideYandexUserInfoApi(): YandexUserInfoApi = YandexUserInfoApiFactory.create()
24
25     @Provides
26     @Singleton
27     fun provideYandexDiskApiFactory(
28         yandexAccountStore: YandexAccountStore,
29         @IoDispatcher ioDispatcher: CoroutineDispatcher,
30     ): YandexDiskApiFactory = YandexDiskApiFactory(
31         accountRepository = yandexAccountStore,
32         ioDispatcher = ioDispatcher,
33     )
34
35     @Provides
36     @Singleton
37     fun provideYandexDiskRepositoryFactory(
38         apiFactory: YandexDiskApiFactory,
39         @IoDispatcher ioDispatcher: CoroutineDispatcher,
40     ): YandexDiskRepositoryFactory = YandexDiskRepositoryFactory(
```

```
41         apiFactory = apiFactory,
42         ioDispatcher = ioDispatcher,
43     )
44
45     @Provides
46     @Singleton
47     fun provideYandexUserInfoRepository(
48         api: YandexUserInfoApi,
49         @IoDispatcher ioDispatcher: CoroutineDispatcher,
50     ): YandexUserInfoRepository = YandexUserInfoRepository(api, ioDispatcher)
51 }
52
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/data/PersistenceModule.kt

```
1 package com.github.nullptroma.wallenc.app.di.modules.data
2
3 import com.github.nullptroma.wallenc.app.di.modules.app.IoDispatcher
4 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncGroupStore
5 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageKeyMapDao
6 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageSyncGroupDao
7 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.YandexAccountDao
8 import
9 com.github.nullptroma.wallenc.infrastructure.android.db.app.repository.StorageKeyMapRepository
10 import
11 com.github.nullptroma.wallenc.infrastructure.android.db.app.repository.StorageSyncGroupRepository
12 import com.github.nullptroma.wallenc.infrastructure.android.db.app.repository.YandexAccountRepository
13 import com.github.nullptroma.wallenc.domain.vault.ports.StorageKeyMapStore
14 import com.github.nullptroma.wallenc.domain.vault.ports.YandexAccountStore
15 import dagger.Module
16 import dagger.Provides
17 import dagger.hilt.InstallIn
18 import dagger.hilt.components.SingletonComponent
19 import kotlinx.coroutines.CoroutineDispatcher
20 import javax.inject.Singleton
21
22 /**
23  * Room DAO/repository types live in `:infrastructure-android` under `domain.vault`
24  * packages.
25  */
26 @Module
27 @InstallIn(SingletonComponent::class)
28 object PersistenceModule {
29
30     @Provides
31     @Singleton
32     fun provideStorageKeyMapRepository(
33         dao: StorageKeyMapDao,
34         @IoDispatcher ioDispatcher: CoroutineDispatcher,
35     ): StorageKeyMapRepository = StorageKeyMapRepository(dao, ioDispatcher)
36
37     @Provides
38     @Singleton
39     fun provideStorageKeyMapStore(impl: StorageKeyMapRepository): StorageKeyMapStore = impl
```

```

40     fun provideStorageSyncGroupStore(
41         dao: StorageSyncGroupDao,
42         @IoDispatcher ioDispatcher: CoroutineDispatcher,
43     ): IStorageSyncGroupStore = StorageSyncGroupRepository(dao, ioDispatcher)
44
45     @Provides
46     @Singleton
47     fun provideYandexAccountRepository(
48         dao: YandexAccountDao,
49         @IoDispatcher ioDispatcher: CoroutineDispatcher,
50     ): YandexAccountRepository = YandexAccountRepository(dao, ioDispatcher)
51
52     @Provides
53     @Singleton
54     fun provideYandexAccountStore(impl: YandexAccountRepository): YandexAccountStore = impl
55 }
56

```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/data/RoomModule.kt

```
1 package com.github.nullptroma.wallenc.app.di.modules.data
2
3 import android.content.Context
4 import com.github.nullptroma.wallenc.infrastructure.android.db.RoomFactory
5 import com.github.nullptroma.wallenc.infrastructure.android.db.app.IAppDb
6 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageKeyMapDao
7 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageMetaInfoDao
8 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageSyncGroupDao
9 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.YandexAccountDao
10 import dagger.Module
11 import dagger.Provides
12 import dagger.hilt.InstallIn
13 import dagger.hilt.android.qualifiers.ApplicationContext
14 import dagger.hilt.components.SingletonComponent
15 import javax.inject.Singleton
16
17 /** Room types are implemented in `:infrastructure-android` with `domain.vault` packages. */
18 @Module
19 @InstallIn(SingletonComponent::class)
20 object RoomModule {
21     @Provides
22     @Singleton
23     fun provideRoomFactory(@ApplicationContext appContext: Context): RoomFactory {
24         return RoomFactory(appContext)
25     }
26
27     @Provides
28     @Singleton
29     fun provideStorageKeyDao(database: IAppDb): StorageKeyMapDao {
30         return database.storageKeyMapDao
31     }
32
33     @Provides
34     @Singleton
35     fun provideStorageMetaInfoDao(database: IAppDb): StorageMetaInfoDao {
36         return database.storageMetaInfoDao
37     }
38
39     @Provides
40     @Singleton
41     fun provideStorageSyncGroupDao(database: IAppDb): StorageSyncGroupDao {
42         return database.storageSyncGroupDao
43     }
44 }
```

```
43     }
44
45     @Provides
46     @Singleton
47     fun provideYandexAccountDao(database: IAppDb): YandexAccountDao {
48         return database.yandexAccountDao
49     }
50
51     @Provides
52     @Singleton
53     fun provideAppDb(factory: RoomFactory): IAppDb {
54         return factory.buildAppDb()
55     }
56 }
```


Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/data/TaskModule.kt

```
1  package com.github.nullptroma.wallenc.app.di.modules.data
2
3  import com.github.nullptroma.wallenc.app.di.modules.app.IoDispatcher
4  import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
5  import com.github.nullptroma.wallenc.task.runtime.TaskOrchestrator
6  import dagger.Module
7  import dagger.Provides
8  import dagger.hilt.InstallIn
9  import dagger.hilt.components.SingletonComponent
10 import kotlinx.coroutines.CoroutineDispatcher
11 import javax.inject.Singleton
12
13 @Module
14 @InstallIn(SingletonComponent::class)
15 object TaskModule {
16
17     @Provides
18     @Singleton
19     fun provideTaskOrchestrator(
20         @IoDispatcher ioDispatcher: CoroutineDispatcher,
21     ): ITaskOrchestrator = TaskOrchestrator(ioDispatcher)
22 }
23
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/data/VaultBindingsModule.kt

```
1 package com.github.nullptroma.wallenc.app.di.modules.data
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
4 import com.github.nullptroma.wallenc.domain.vault.vaults.VaultsManager
5 import com.github.nullptroma.wallenc.vault.contract.VaultRegistrar
6 import dagger.Binds
7 import dagger.Module
8 import dagger.hilt.InstallIn
9 import dagger.hilt.components.SingletonComponent
10 import javax.inject.Singleton
11
12 @Module
13 @InstallIn(SingletonComponent::class)
14 abstract class VaultBindingsModule {
15
16     @Binds
17     @Singleton
18     abstract fun bindVaultsManager(impl: VaultsManager): IVaultsManager
19
20     @Binds
21     @Singleton
22     abstract fun bindVaultRegistrar(impl: VaultsManager): VaultRegistrar
23 }
24
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/
modules/data/VaultModule.kt

```
1 package com.github.nullptroma.wallenc.app.di.modules.data
2
3 import android.content.Context
4 import com.github.nullptroma.wallenc.app.di.modules.app.IoDispatcher
5 import com.github.nullptroma.wallenc.domain.interfaces.IUnlockManager
6 import com.github.nullptroma.wallenc.domain.interfaces.IVault
7 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
8 import
9 com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository.YandexDiskRepository
10 import
11 com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo.repository.YandexUserInfoRepository
12 import com.github.nullptroma.wallenc.domain.vault.ports.StorageKeyMapStore
13 import com.github.nullptroma.wallenc.domain.vault.ports.YandexAccountStore
14 import com.github.nullptroma.wallenc.domain.vault.vaults.VaultsManager
15 import com.github.nullptroma.wallenc.domain.vault.vaults.local.LocalVault
16 import dagger.Module
17 import dagger.Provides
18 import dagger.hilt.InstallIn
19 import dagger.hilt.android.qualifiers.ApplicationContext
20 import dagger.hilt.components.SingletonComponent
21 import kotlinx.coroutines.CoroutineDispatcher
22 import javax.inject.Singleton
23
24 @Module
25 @InstallIn(SingletonComponent::class)
26 object VaultModule {
27
28     @Provides
29     @Singleton
30     fun provideLocalVault(
31         @IoDispatcher ioDispatcher: CoroutineDispatcher,
32         @ApplicationContext context: Context,
33     ): IVault = LocalVault(
34         ioDispatcher = ioDispatcher,
35         // getExternalFilesDir may return null before storage is ready; LocalVault handles
36         null root.
37         vaultRoot = context.getExternalFilesDir("LocalVault"),
38     )
39
40     @Provides
41     @Singleton
42     fun provideVaultsManager(
43         @IoDispatcher ioDispatcher: CoroutineDispatcher,
```

```

41         localVault: IVault,
42         keyRepo: StorageKeyMapStore,
43         yandexAccountStore: YandexAccountStore,
44         yandexUserInfoRepository: YandexUserInfoRepository,
45         yandexDiskRepositoryFactory: YandexDiskRepositoryFactory,
46     ): VaultsManager = VaultsManager(
47         ioDispatcher = ioDispatcher,
48         localVault = localVault,
49         keyRepo = keyRepo,
50         yandexAccountStore = yandexAccountStore,
51         yandexUserInfoRepository = yandexUserInfoRepository,
52         yandexDiskRepositoryFactory = yandexDiskRepositoryFactory,
53     )
54
55     @Provides
56     @Singleton
57     fun provideUnlockManager(vaultsManager: IVaultsManager): IUnlockManager =
58         vaultsManager.unlockManager
59 }
60

```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/di/modules/domain/UseCasesModule.kt

```
1 package com.github.nullptroma.wallenc.app.di.modules.domain
2
3 import com.github.nullptroma.wallenc.app.sync.StorageSyncTaskTitleFormatter
4 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncEngine
5 import com.github.nullptroma.wallenc.domain.tasks.IStorageSyncTaskTitleFormatter
6 import com.github.nullptroma.wallenc.usecases.StorageSyncEngine
7 import dagger.Binds
8 import dagger.Module
9 import dagger.hilt.InstallIn
10 import dagger.hilt.components.SingletonComponent
11 import javax.inject.Singleton
12
13 @Module
14 @InstallIn(SingletonComponent::class)
15 abstract class UseCasesModule {
16
17     @Binds
18     @Singleton
19     abstract fun bindStorageSyncEngine(impl: StorageSyncEngine): IStorageSyncEngine
20
21     @Binds
22     @Singleton
23     abstract fun bindStorageSyncTaskTitleFormatter(
24         impl: StorageSyncTaskTitleFormatter,
25     ): IStorageSyncTaskTitleFormatter
26 }
27
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/locale/
AppLocaleControllerImpl.kt

```
1 package com.github.nullptroma.wallenc.app.locale
2
3 import android.content.Context
4 import com.github.nullptroma.wallenc.ui.locale.AppLanguage
5 import com.github.nullptroma.wallenc.ui.locale.AppLocaleController
6 import dagger.hilt.android.qualifiers.ApplicationContext
7 import kotlinx.coroutines.flow.Flow
8 import javax.inject.Inject
9 import javax.inject.Singleton
10
11 @Singleton
12 class AppLocaleControllerImpl @Inject constructor(
13     @param:ApplicationContext private val context: Context,
14 ) : AppLocaleController {
15
16     override val language: Flow<AppLanguage> = AppLocaleStorage.languageFlow(context)
17
18     override suspend fun setLanguage(language: AppLanguage) {
19         AppLocaleStorage.persistLanguage(context, language)
20     }
21
22     override suspend fun applyStoredLocale() {
23         AppLocaleStorage.applyStored(context)
24     }
25 }
26
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/locale/ AppLocaleStorage.kt

```
1 package com.github.nullptroma.wallenc.app.locale
2
3 import android.content.Context
4 import androidx.appcompat.app.AppCompatActivity
5 import androidx.core.os.LocaleListCompat
6 import androidx.datastore.core.DataStore
7 import androidx.datastore.preferences.core.Preferences
8 import androidx.datastore.preferences.core.stringPreferencesKey
9 import androidx.datastore.preferences.preferencesDataStore
10 import com.github.nullptroma.wallenc.ui.locale.AppLanguage
11 import kotlinx.coroutines.channels.awaitClose
12 import kotlinx.coroutines.flow.Flow
13 import kotlinx.coroutines.flow.callbackFlow
14 import kotlinx.coroutines.flow.first
15 import kotlinx.coroutines.runBlocking
16 import androidx.core.content.edit
17
18 /**
19  * Хранение выбранного языка. SharedPreferences — для синхронного чтения в
20  * [android.app.Application.attachBaseContext]; DataStore — только для миграции со старой
21  * версии.
22  */
23 internal object AppLocaleStorage {
24     private const val PREFS_NAME = "app_locale"
25     private const val PREFS_KEY_LANGUAGE = "app_language"
26
27     private val legacyLanguageKey = stringPreferencesKey(PREFS_KEY_LANGUAGE)
28
29     private val Context.legacyLocaleDataStore: DataStore<Preferences> by
30     preferencesDataStore(
31         name = "app_locale",
32     )
33
34     /** B [android.app.Application.attachBaseContext] [Context.getApplicationContext] ещё
35     null. */
36     private fun storageContext(context: Context): Context =
37         context.applicationContext ?: context
38
39     private fun prefs(context: Context) =
40         storageContext(context).getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)
41
42     fun languageFlow(context: Context): Flow<AppLanguage> = callbackFlow {
```

```

41     val storage = storageContext(context)
42     val listener =
43         android.content.SharedPreferences.OnSharedPreferenceChangeListener { _, key ->
44             if (key == PREFS_KEY_LANGUAGE) {
45                 trySend(readLanguageSync(storage))
46             }
47         }
48     trySend(readLanguageSync(storage))
49     prefs(storage).registerOnSharedPreferenceChangeListener(listener)
50     awaitClose { prefs(storage).unregisterOnSharedPreferenceChangeListener(listener) }
51 }
52
53 fun persistLanguage(context: Context, language: AppLanguage) {
54     prefs(context).edit { putString(PREFS_KEY_LANGUAGE, language.storageValue) }
55     applyLanguage(language)
56 }
57
58 fun readLanguageSync(context: Context): AppLanguage {
59     val raw = prefs(context).getString(PREFS_KEY_LANGUAGE, null)
60     return raw?.toAppLanguage() ?: AppLanguage.System
61 }
62
63 /** Без блокировок; безопасно из [android.app.Application.attachBaseContext]. */
64 fun applyStored(context: Context) {
65     applyLanguage(readLanguageSync(context))
66 }
67
68 fun applyLanguage(language: AppLanguage) {
69     val localeList = when (language) {
70         AppLanguage.System -> LocaleListCompat.getEmptyLocaleList()
71         AppLanguage.English -> LocaleListCompat.forLanguageTags("en")
72         AppLanguage.Russian -> LocaleListCompat.forLanguageTags("ru")
73     }
74     AppCompatDelegate.setApplicationLocales(localeList)
75 }
76
77 /** Однократно переносит значение из DataStore (если SP ещё пуст). */
78 fun migrateLegacyDataStoreIfNeeded(context: Context) {
79     val storage = storageContext(context)
80     if (prefs(storage).contains(PREFS_KEY_LANGUAGE)) {
81         return
82     }
83     runBlocking {
84         runCatching {
85             val legacy = storage.legacyLocaleDataStore.data.first()[legacyLanguageKey]

```



```

86         if (legacy != null) {
87             prefs(storage).edit { putString(PREFS_KEY_LANGUAGE, legacy) }
88         }
89     }
90 }
91
92
93 private fun String.toAppLanguage(): AppLanguage = when (this) {
94     STORAGE_EN -> AppLanguage.English
95     STORAGE_RU -> AppLanguage.Russian
96     else -> AppLanguage.System
97 }
98
99 private val AppLanguage.storageValue: String
100     get() = when (this) {
101         AppLanguage.System -> STORAGE_SYSTEM
102         AppLanguage.English -> STORAGE_EN
103         AppLanguage.Russian -> STORAGE_RU
104     }
105
106 private const val STORAGE_SYSTEM = "system"
107 private const val STORAGE_EN = "en"
108 private const val STORAGE_RU = "ru"
109 }
110

```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/navigation/ WallencExternalLaunch.kt

```
1  package com.github.nullptroma.wallenc.app.navigation
2
3  import android.content.Context
4  import android.content.Intent
5  import android.net.Uri
6  import androidx.core.net.toUri
7  import com.github.nullptroma.wallenc.app.MainActivity
8  import com.github.nullptroma.wallenc.ui.navigation.WallencDeepLinks
9
10 /**
11  * Запуск [MainActivity] и взаимодействие с foreground-сервисом из **вне** Compose:
12  * [android.app.PendingIntent],
13  * стабильные action, id канала/нотификации.
14  *
15  * URI см. [WallencDeepLinks] в `:ui` — тот же контракт, что у
16  * [androidx.navigation.navDeepLink] и манифеста.
17  */
18 object WallencExternalLaunch {
19
20     /** Коды [android.app.PendingIntent]: у каждого сценария свой requestCode. */
21     object PendingIntentRequestCodes {
22         const val NOTIFICATION_TASK_PIPELINE_CANCEL_SERVICE = 0
23         const val NOTIFICATION_TASK_PIPELINE_OPEN_TASKS = 2
24     }
25
26     /**
27      * [Intent.getAction] для
28      * [com.github.nullptroma.wallenc.app.tasks.TaskPipelineForegroundService]
29      * из кнопки уведомления ([PendingIntent.getService]).
30      */
31     const val FOREGROUND_SERVICE_ACTION_CANCEL_ALL_TASKS =
32         "com.github.nullptroma.wallenc.action.CANCEL_ALL_TASKS"
33
34     object ForegroundTaskPipelineNotification {
35         const val CHANNEL_ID = "wallenc_task_pipeline"
36         const val NOTIFICATION_ID = 1001
37     }
38
39     fun mainActivityViewIntent(context: Context, uri: Uri): Intent =
40         Intent(Intent.ACTION_VIEW, uri).apply {
41             setClass(context, MainActivity::class.java)
42             flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_SINGLE_TOP
43         }
```

```
41
42     fun mainActivityViewIntentForTasks(context: Context): Intent =
43         mainActivityViewIntent(context, WallencDeepLinks.TASKS_URI_PATTERN.toUri())
44     }
45
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/sync/
StorageSyncBootReceiver.kt

```
1 package com.github.nullptroma.wallenc.app.sync
2
3 import android.content.BroadcastReceiver
4 import android.content.Context
5 import android.content.Intent
6 import com.github.nullptroma.wallenc.app.di.StorageSyncBootEntryPoint
7 import dagger.hilt.android.EntryPointAccessors
8 import timber.log.Timber
9
10 class StorageSyncBootReceiver : BroadcastReceiver() {
11     override fun onReceive(context: Context, intent: Intent?) {
12         if (intent?.action != Intent.ACTION_BOOT_COMPLETED) {
13             return
14         }
15         val scheduler = EntryPointAccessors.fromApplication(
16             context.applicationContext,
17             StorageSyncBootEntryPoint::class.java,
18         ).storageSyncScheduler()
19         scheduler.ensureScheduled()
20         Timber.d("Rescheduled periodic storage sync after boot")
21     }
22 }
23
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/sync/ StorageSyncBootstrap.kt

```
1 package com.github.nullptroma.wallenc.app.sync
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths
4 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
5 import com.github.nullptroma.wallenc.domain.tasks.StorageSyncTriggerReason
6 import com.github.nullptroma.wallenc.usecases.RunStorageSyncUseCase
7 import kotlinx.coroutines.CoroutineScope
8 import kotlinx.coroutines.Dispatchers
9 import kotlinx.coroutines.FlowPreview
10 import kotlinx.coroutines.SupervisorJob
11 import kotlinx.coroutines.flow.collectLatest
12 import kotlinx.coroutines.flow.combine
13 import kotlinx.coroutines.flow.debounce
14 import kotlinx.coroutines.flow.filter
15 import kotlinx.coroutines.flow.map
16 import kotlinx.coroutines.flow.merge
17 import kotlinx.coroutines.launch
18 import javax.inject.Inject
19 import javax.inject.Singleton
20
21 @Singleton
22 @OptIn(FlowPreview::class)
23 class StorageSyncBootstrap @Inject constructor(
24     private val scheduler: StorageSyncScheduler,
25     private val vaultsManager: IVaultsManager,
26     private val syncRunner: RunStorageSyncUseCase,
27 ) {
28     private val scope = CoroutineScope(SupervisorJob() + Dispatchers.IO)
29
30     fun start() {
31         scheduler.ensureScheduled()
32         scope.launch {
33             combine(
34                 vaultsManager.allStorages,
35                 vaultsManager.unlockManager.openedStorages,
36             ) { rootStorages, opened ->
37                 (rootStorages + opened.values).distinctBy { it.uuid }
38             }.collectLatest { storages ->
39                 if (storages.isEmpty()) {
40                     return@collectLatest
41                 }
42                 val triggers = storages.flatMap { storage ->
```

```

43         listOf(
44             storage.accessor.filesUpdates
45                 .filter { page ->
46                     page.data.any { file ->
47                         StorageSyncPaths.isSyncableUserPath(file.metaInfo.path)
48                     }
49                 }
50             .map {},
51             storage.accessor.dirsUpdates
52                 .filter { page ->
53                     page.data.any { dir ->
54                         StorageSyncPaths.isSyncableUserPath(dir.metaInfo.path)
55                     }
56                 }
57             .map {},
58         )
59     }
60     merge(*triggers.toTypedArray())
61         .filter { shouldScheduleDebounceSync() }
62         .debounce(RunStorageSyncUseCase.DEBOUNCE_AFTER_CHANGE_MS)
63         .collect {
64             syncRunner.enqueue(StorageSyncTriggerReason.Debounce)
65         }
66     }
67 }
68 }
69
70 /**
71  * Игнорировать события во время sync и сразу после него: запись файлов при sync
72  * (в т.ч. через [EncryptedStorageAccessor]) иначе снова запускает debounce через 60 с.
73  */
74 private fun shouldScheduleDebounceSync(): Boolean {
75     if (syncRunner.syncRunning.value) {
76         return false
77     }
78     return System.currentTimeMillis() >= syncRunner.debounceSuppressUntilMs.value
79 }
80 }
81

```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/sync/
StorageSyncScheduler.kt

```
1  package com.github.nullptroma.wallenc.app.sync
2
3  import android.content.Context
4  import androidx.work.Constraints
5  import androidx.work.ExistingPeriodicWorkPolicy
6  import androidx.work.NetworkType
7  import androidx.work.PeriodicWorkRequestBuilder
8  import androidx.work.WorkManager
9  import dagger.hilt.android.qualifiers.ApplicationContext
10 import java.util.concurrent.TimeUnit
11 import javax.inject.Inject
12 import javax.inject.Singleton
13
14 @Singleton
15 class StorageSyncScheduler @Inject constructor(
16     @param:ApplicationContext private val app: Context,
17 ) {
18     fun ensureScheduled() {
19         val request = PeriodicWorkRequestBuilder<StorageSyncWorker>(
20             repeatInterval = 30,
21             repeatIntervalTimeUnit = TimeUnit.MINUTES,
22         )
23             .setConstraints(
24                 Constraints.Builder()
25                     .setRequiredNetworkType(NetworkType.CONNECTED)
26                     .build(),
27             )
28             .build()
29
30         // KEEP: UPDATE сбрасывает таймер periodic work при каждом onCreate процесса.
31         WorkManager.getInstance(app).enqueueUniquePeriodicWork(
32             StorageSyncWorker.UNIQUE_WORK_NAME,
33             ExistingPeriodicWorkPolicy.KEEP,
34             request,
35         )
36     }
37 }
38
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/sync/
StorageSyncTaskTitleFormatter.kt

```
1 package com.github.nullptroma.wallenc.app.sync
2
3 import com.github.nullptroma.wallenc.domain.tasks.IStorageSyncTaskTitleFormatter
4 import com.github.nullptroma.wallenc.domain.tasks.StorageSyncTriggerReason
5 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
6 import com.github.nullptroma.wallenc.ui.resources.storageSyncTaskTitle
7 import javax.inject.Inject
8 import javax.inject.Singleton
9
10 @Singleton
11 class StorageSyncTaskTitleFormatter @Inject constructor(
12     private val uiStrings: UiStringResolver,
13 ) : IStorageSyncTaskTitleFormatter {
14     override fun format(reason: StorageSyncTriggerReason): String =
15         uiStrings.storageSyncTaskTitle(reason)
16 }
17
```


Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/sync/
StorageSyncWorker.kt

```
1 package com.github.nullptroma.wallenc.app.sync
2
3 import android.content.Context
4 import androidx.hilt.work.HiltWorker
5 import androidx.work.CoroutineWorker
6 import androidx.work.WorkerParameters
7 import com.github.nullptroma.wallenc.domain.tasks.StorageSyncTriggerReason
8 import com.github.nullptroma.wallenc.usecases.RunStorageSyncUseCase
9 import com.github.nullptroma.wallenc.usecases.StorageSyncRunOutcome
10 import dagger.assisted.Assisted
11 import dagger.assisted.AssistedInject
12 import timber.log.Timber
13
14 @HiltWorker
15 class StorageSyncWorker @AssistedInject constructor(
16     @Assisted appContext: Context,
17     @Assisted params: WorkerParameters,
18     private val syncRunner: RunStorageSyncUseCase,
19 ) : CoroutineWorker(appContext, params) {
20
21     override suspend fun doWork(): Result {
22         Timber.d("Periodic storage sync started (attempt %d)", runAttemptCount)
23         return when (val outcome =
24             syncRunner.enqueueAndAwait(StorageSyncTriggerReason.Background)) {
25             StorageSyncRunOutcome.SkippedAlreadyRunning -> {
26                 Timber.d("Periodic storage sync skipped – already running")
27                 Result.success()
28             }
29             StorageSyncRunOutcome.Completed -> {
30                 Timber.d("Periodic storage sync finished")
31                 Result.success()
32             }
33             StorageSyncRunOutcome.Cancelled -> {
34                 Timber.d("Periodic storage sync cancelled")
35                 Result.success()
36             }
37             is StorageSyncRunOutcome.Failed -> {
38                 Timber.w(outcome.error, "Periodic storage sync failed")
39                 Result.retry()
40             }
41         }
42     }
43 }
```

```
42
43     companion object {
44         const val UNIQUE_WORK_NAME = "wallenc-storage-sync-periodic"
45     }
46 }
47
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/tasks/ TaskPipelineForegroundBootstrap.kt

```
1  package com.github.nullptroma.wallenc.app.tasks
2
3  import android.content.Context
4  import android.content.Intent
5  import androidx.core.content.ContextCompat
6  import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
7  import com.github.nullptroma.wallenc.domain.tasks.TaskForegroundUiState
8  import dagger.hilt.android.qualifiers.ApplicationContext
9  import kotlinx.coroutines.CoroutineScope
10 import kotlinx.coroutines.Dispatchers
11 import kotlinx.coroutines.SupervisorJob
12 import kotlinx.coroutines.flow.distinctUntilChanged
13 import kotlinx.coroutines.flow.filter
14 import kotlinx.coroutines.flow.map
15 import kotlinx.coroutines.launch
16 import javax.inject.Inject
17 import javax.inject.Singleton
18
19 @Singleton
20 class TaskPipelineForegroundBootstrap @Inject constructor(
21     @param:ApplicationContext private val app: Context,
22     private val orchestrator: ITaskOrchestrator,
23 ) {
24     private val scope = CoroutineScope(SupervisorJob() + Dispatchers.Default)
25
26     fun start() {
27         scope.launch {
28             orchestrator.foregroundUi
29                 .map { it is TaskForegroundUiState.Visible }
30                 .distinctUntilChanged()
31                 .filter { visible -> visible }
32                 .collect {
33                     ContextCompat.startForegroundService(
34                         app,
35                         Intent(app, TaskPipelineForegroundService::class.java),
36                     )
37                 }
38         }
39     }
40 }
41
```

Исходный файл app/src/main/java/com/github/nullptroma/wallenc/app/tasks/ TaskPipelineForegroundService.kt

```
1 package com.github.nullptroma.wallenc.app.tasks
2
3 import android.app.Notification
4 import android.app.NotificationChannel
5 import android.app.NotificationManager
6 import android.app.PendingIntent
7 import android.app.Service
8 import android.content.Intent
9 import android.graphics.Color
10 import android.os.IBinder
11 import android.view.View
12 import android.widget.RemoteViews
13 import androidx.core.app.NotificationCompat
14 import com.github.nullptroma.wallenc.app.R
15 import com.github.nullptroma.wallenc.app.navigation.WallencExternalLaunch
16 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
17 import com.github.nullptroma.wallenc.domain.tasks.TaskForegroundItem
18 import com.github.nullptroma.wallenc.domain.tasks.TaskForegroundUiState
19 import com.github.nullptroma.wallenc.domain.tasks.TaskId
20 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
21 import com.github.nullptroma.wallenc.ui.resources.resolve
22 import dagger.hilt.android.AndroidEntryPoint
23 import kotlinx.coroutines.CoroutineScope
24 import kotlinx.coroutines.Dispatchers
25 import kotlinx.coroutines.ExperimentalCoroutinesApi
26 import kotlinx.coroutines.SupervisorJob
27 import kotlinx.coroutines.cancel
28 import kotlinx.coroutines.channels.Channel
29 import kotlinx.coroutines.delay
30 import kotlinx.coroutines.launch
31 import java.util.concurrent.ConcurrentHashMap
32 import javax.inject.Inject
33 import kotlin.math.roundToInt
34
35 /**
36  * Single foreground notification (one [NotificationManager.notify] id) that accumulates all
37  * task
38  * states. [orchestrator.foregroundUi] enqueues into [foregroundUiQueue] while [canPush]
39  * allows
40  * (at most one pending frame until the worker finishes an iteration). Indeterminate tasks:
41  * the
42  * worker reuses [lastUiState] when the queue is empty so dots advance without extra queue
43  * entries.
```

```

40     * Each handled state is followed by [NOTIFICATION_QUEUE_STEP_MS] before the next step.
41     */
42     @AndroidEntryPoint
43     class TaskPipelineForegroundService : Service() {
44
45         @Inject
46         lateinit var orchestrator: ITaskOrchestrator
47
48         @Inject
49         lateinit var uiStrings: UiStringResolver
50
51         private var repeat = false
52         private var canPush = true
53         private var lastUiState: TaskForegroundUiState? = null
54
55         /** Dot-animation phase per task (0...3); absent means 0 for first paint. */
56         private val indeterminateDotsPhaseByTaskId = ConcurrentHashMap<TaskId, Int>()
57
58         private val foregroundUiQueue = Channel<TaskForegroundUiState>(Channel.UNLIMITED)
59
60         private val serviceJob = SupervisorJob()
61         private val serviceScope = CoroutineScope(serviceJob + Dispatchers.Main.immediate)
62
63         override fun onBind(intent: Intent?): IBinder? = null
64
65         override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
66             if (intent?.action ==
WallencExternalLaunch.FOREGROUND_SERVICE_ACTION_CANCEL_ALL_TASKS &&
67                 ::orchestrator.isInitialized
68             ) {
69                 orchestrator.cancelAll()
70             }
71             return START_STICKY
72         }
73
74         @OptIn(ExperimentalCoroutinesApi::class)
75         override fun onCreate() {
76             super.onCreate()
77             ensureChannel()
78             startForeground(
79                 WallencExternalLaunch.ForegroundTaskPipelineNotification.NOTIFICATION_ID,
80                 buildPlaceholderNotification(),
81             )
82
83             val nm = getSystemService(NotificationManager::class.java)
84

```

```

85         serviceScope.launch {
86             var sawVisible = false
87             while (true) {
88                 if(repeat && !foregroundUiQueue.isEmpty || !repeat) {
89                     lastUiState = foregroundUiQueue.receive()
90                 }
91                 canPush = true
92
93                 val ui = lastUiState ?: continue
94                 when (ui) {
95                     TaskForegroundUiState.Hidden -> {
96                         repeat = false
97                         if (sawVisible) {
98                             nm.cancel(WallencExternalLaunch.ForegroundTaskPipelineNotification.NOTIFICATION_ID)
99                             indeterminateDotsPhaseByTaskId.clear()
100                             stopForeground(STOP_FOREGROUND_REMOVE)
101                             stopSelf()
102                             return@launch
103                         }
104                     }
105
106                     is TaskForegroundUiState.Visible -> {
107                         if (ui.tasks.isNotEmpty()) {
108                             sawVisible = true
109                             nm.notify(
110                                 WallencExternalLaunch.ForegroundTaskPipelineNotification.NOTIFICATION_ID,
111                                 buildAccumulatedNotification(ui.tasks),
112                             )
113                         }
114                         // repeatedly receive Visible with same tasks while indeterminate
115                         dots animate
116                         repeat = ui.tasks.any { it.progress?.fraction == null }
117                     }
118                 }
119                 delay(NOTIFICATION_QUEUE_STEP_MS)
120             }
121
122         serviceScope.launch {
123             orchestrator.foregroundUi.collect { ui ->
124                 if(canPush || ui is TaskForegroundUiState.Hidden) {
125                     foregroundUiQueue.send(ui)
126                     canPush = false
127                 }

```

```

128         }
129     }
130 }
131
132 override fun onDestroy() {
133     foregroundUiQueue.close()
134     indeterminateDotsPhaseByTaskId.clear()
135     serviceScope.cancel()
136     super.onDestroy()
137 }
138
139 private fun ensureChannel() {
140     val nm = getSystemService(NotificationManager::class.java)
141     val channel = NotificationChannel(
142         WallencExternalLaunch.ForegroundTaskPipelineNotification.CHANNEL_ID,
143         getString(R.string.task_notification_channel_name),
144         NotificationManager.IMPORTANCE_LOW,
145     )
146     nm.createNotificationChannel(channel)
147 }
148
149 private fun buildPlaceholderNotification(): Notification =
150     NotificationCompat.Builder(this,
151         WallencExternalLaunch.ForegroundTaskPipelineNotification.CHANNEL_ID)
152         .setContentTitle(getString(R.string.task_notification_title))
153         .setContentText(getString(R.string.task_notification_preparing))
154         .setSmallIcon(R.drawable.ic_stat_wallenc)
155         .setOngoing(true)
156         .setOnlyAlertOnce(true)
157         .setContentIntent(openTaskPipelinePendingIntent())
158         .addAction(
159             0,
160             getString(R.string.task_notification_cancel),
161             cancelAllTasksPendingIntent(),
162         )
163         .build()
164
165 private fun buildAccumulatedNotification(tasks: List<TaskForegroundItem>): Notification
166 {
167     val sorted = tasks.sortedBy { it.taskId.uuid }
168     val big = RemoteViews(packageName, R.layout.notification_wallenc_tasks_big)
169     applyNotificationTemplateTextColor(big)
170     bindTaskRows(big, sorted)
171
172     val collapsedSubtext = when {
173         sorted.isEmpty() ->

```

```

172         getString(R.string.task_notification_preparing)
173
174         sorted.all { it.progress?.fraction == null } ->
175             getString(R.string.task_notification_indeterminate)
176
177         else -> resources.getQuantityString(
178             R.plurals.task_notification_group_subtext,
179             sorted.size,
180             sorted.size,
181         )
182     }
183
184     return NotificationCompat.Builder(
185         this,
186         WallencExternalLaunch.ForegroundTaskPipelineNotification.CHANNEL_ID,
187     )
188         .setContentTitle(getString(R.string.task_notification_title))
189         .setContentText(collapsedSubtext)
190         .setSmallIcon(R.drawable.ic_stat_wallenc)
191         .setOngoing(true)
192         .setOnlyAlertOnce(true)
193         .setContentIntent(openTaskPipelinePendingIntent())
194         .setStyle(NotificationCompat.DecoratedCustomViewStyle())
195         .setCustomBigContentView(big)
196         .addAction(
197             0,
198             getString(R.string.task_notification_cancel),
199             cancelAllTasksPendingIntent(),
200         )
201         .build()
202     }
203
204     private fun openTaskPipelinePendingIntent(): PendingIntent =
205         PendingIntent.getActivity(
206             this,
207             WallencExternalLaunch.PendingIntentRequestCodes.NOTIFICATION_TASK_PIPELINE_OPEN_TASKS,
208             WallencExternalLaunch.mainActivityViewIntentForTasks(this),
209             PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE,
210         )
211
212     private fun cancelAllTasksPendingIntent(): PendingIntent =
213         PendingIntent.getService(
214             this,
215             WallencExternalLaunch.PendingIntentRequestCodes.NOTIFICATION_TASK_PIPELINE_CANCEL_SERVICE,

```



```

216         Intent(this, TaskPipelineForegroundService::class.java).setAction(
217             WallencExternalLaunch.FOREGROUND_SERVICE_ACTION_CANCEL_ALL_TASKS,
218         ),
219         PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE,
220     )
221
222     /**
223      * Uses only [android.R.attr.textColor] from
224      * [android.R.style.TextAppearance_Material_Notification_Title]
225      * so line size stays from layout (13sp) while color matches system notification title
226      * text.
227      */
228     private fun applyNotificationTemplateTextColor(remoteViews: RemoteViews) {
229         val color = notificationTemplateTextColor()
230         for (i in 0 until MAX_TASK_ROWS) {
231             remoteViews.setTextColor(TASK_TITLE_IDS[i], color)
232             remoteViews.setTextColor(TASK_SUBTITLE_IDS[i], color)
233         }
234     }
235
236     private fun notificationTemplateTextColor(): Int {
237         val attrs = intArrayOf(android.R.attr.textColor)
238         val a = theme.obtainStyledAttributes(
239             null,
240             attrs,
241             0,
242             android.R.style.TextAppearance_Material_Notification
243         )
244         val c = a.getColor(0, Color.WHITE)
245         a.recycle()
246         return c
247     }
248
249     private fun bindTaskRows(remoteViews: RemoteViews, sorted: List<TaskForegroundItem>) {
250         val n = sorted.size
251         if (n == 0) {
252             for (i in 0 until MAX_TASK_ROWS) {
253                 hideRow(remoteViews, i)
254             }
255             return
256         }
257         if (n <= MAX_TASK_ROWS) {
258             for (i in 0 until n) {
259                 showTaskRow(remoteViews, i, sorted[i])
260             }
261             for (i in n until MAX_TASK_ROWS) {

```

```

260         hideRow(remoteViews, i)
261     }
262 } else {
263     for (i in 0 until MAX_TASK_ROWS - 1) {
264         showTaskRow(remoteViews, i, sorted[i])
265     }
266     val remaining = n - (MAX_TASK_ROWS - 1)
267     showOverflowRow(remoteViews, remaining)
268 }
269 advanceIndeterminateDotPhasesAfterBind(sorted, n)
270 }
271
272 private fun advanceIndeterminateDotPhasesAfterBind(sorted: List<TaskForegroundItem>, n:
Int) {
273     if (n == 0) return
274     val displayed =
275         if (n <= MAX_TASK_ROWS) sorted else sorted.take(MAX_TASK_ROWS - 1)
276     val presentIds = sorted.map { it.taskId }.toSet()
277     indeterminateDotsPhaseByTaskId.keys.retainAll { it in presentIds }
278     for (t in displayed) {
279         if (t.progress?.fraction == null) {
280             val id = t.taskId
281             indeterminateDotsPhaseByTaskId[id] =
282                 ((indeterminateDotsPhaseByTaskId[id] ?: 0) + 1) % 4
283         } else {
284             indeterminateDotsPhaseByTaskId.remove(t.taskId)
285         }
286     }
287 }
288
289 private fun hideRow(remoteViews: RemoteViews, index: Int) {
290     remoteViews.setViewVisibility(TASK_ROW_IDS[index], View.GONE)
291     remoteViews.setViewVisibility(TASK_LABEL_BAR_ROW_IDS[index], View.GONE)
292     remoteViews.setTextViewText(TASK_SUBTITLE_IDS[index], "")
293 }
294
295 private fun showTaskRow(remoteViews: RemoteViews, index: Int, task: TaskForegroundItem)
{
296     remoteViews.setViewVisibility(TASK_ROW_IDS[index], View.VISIBLE)
297     remoteViews.setViewVisibility(TASK_LABEL_BAR_ROW_IDS[index], View.VISIBLE)
298     remoteViews.setTextViewText(TASK_TITLE_IDS[index], task.title)
299     val label = task.progress?.label?.resolve(uiStrings).orEmpty()
300     val fraction = task.progress?.fraction
301     if (fraction != null) {
302         if (label.isNotEmpty()) {

```

```

303         remoteViews.setViewVisibility(TASK_SUBTITLE_IDS[index], View.VISIBLE)
304         remoteViews.setTextViewText(TASK_SUBTITLE_IDS[index], label)
305     } else {
306         remoteViews.setViewVisibility(TASK_SUBTITLE_IDS[index], View.GONE)
307         remoteViews.setTextViewText(TASK_SUBTITLE_IDS[index], "")
308     }
309     remoteViews.setViewVisibility(TASK_PROGRESS_IDS[index], View.VISIBLE)
310     val pct = (fraction.coerceIn(0f, 1f) * 100).roundToInt()
311     remoteViews.setProgressBar(TASK_PROGRESS_IDS[index], 100, pct, false)
312 } else {
313     remoteViews.setViewVisibility(TASK_SUBTITLE_IDS[index], View.VISIBLE)
314     remoteViews.setTextViewText(
315         TASK_SUBTITLE_IDS[index],
316         indeterminateSubtitleWithDots(task.taskId, label),
317     )
318     remoteViews.setViewVisibility(TASK_PROGRESS_IDS[index], View.GONE)
319 }
320 }
321
322 private fun showOverflowRow(remoteViews: RemoteViews, remainingCount: Int) {
323     val overflowIndex = MAX_TASK_ROWS - 1
324     remoteViews.setViewVisibility(TASK_ROW_IDS[overflowIndex], View.VISIBLE)
325     remoteViews.setTextViewText(
326         TASK_TITLE_IDS[overflowIndex],
327         resources.getQuantityString(
328             R.plurals.task_notification_more_tasks,
329             remainingCount,
330             remainingCount,
331         ),
332     )
333     remoteViews.setViewVisibility(TASK_LABEL_BAR_ROW_IDS[overflowIndex], View.GONE)
334     remoteViews.setTextViewText(TASK_SUBTITLE_IDS[overflowIndex], "")
335 }
336
337 private fun indeterminateSubtitleWithDots(taskId: TaskId, baseLabel: String): String {
338     val phase = (indeterminateDotsPhaseByTaskId[taskId] ?: 0) % 4
339     val dots = when (phase) {
340         0 -> ""
341         1 -> "."
342         2 -> ".."
343         else -> "..."
344     }
345     return if (baseLabel.isEmpty()) dots else baseLabel + dots
346 }
347

```

```

348     companion object {
349         private const val NOTIFICATION_QUEUE_STEP_MS = 500L
350
351         /** Must match [R.layout.notification_wallenc_tasks_big] row count. */
352         private const val MAX_TASK_ROWS = 8
353
354         private val TASK_ROW_IDS = intArrayOf(
355             R.id.task_row_0,
356             R.id.task_row_1,
357             R.id.task_row_2,
358             R.id.task_row_3,
359             R.id.task_row_4,
360             R.id.task_row_5,
361             R.id.task_row_6,
362             R.id.task_row_7,
363         )
364
365         private val TASK_TITLE_IDS = intArrayOf(
366             R.id.task_title_0,
367             R.id.task_title_1,
368             R.id.task_title_2,
369             R.id.task_title_3,
370             R.id.task_title_4,
371             R.id.task_title_5,
372             R.id.task_title_6,
373             R.id.task_title_7,
374         )
375
376         private val TASK_PROGRESS_IDS = intArrayOf(
377             R.id.task_progress_0,
378             R.id.task_progress_1,
379             R.id.task_progress_2,
380             R.id.task_progress_3,
381             R.id.task_progress_4,
382             R.id.task_progress_5,
383             R.id.task_progress_6,
384             R.id.task_progress_7,
385         )
386
387         private val TASK_SUBTITLE_IDS = intArrayOf(
388             R.id.task_subtitle_0,
389             R.id.task_subtitle_1,
390             R.id.task_subtitle_2,
391             R.id.task_subtitle_3,
392             R.id.task_subtitle_4,

```

```
393         R.id.task_subtitle_5,
394         R.id.task_subtitle_6,
395         R.id.task_subtitle_7,
396     )
397
398     private val TASK_LABEL_BAR_ROW_IDS = intArrayOf(
399         R.id.task_label_bar_row_0,
400         R.id.task_label_bar_row_1,
401         R.id.task_label_bar_row_2,
402         R.id.task_label_bar_row_3,
403         R.id.task_label_bar_row_4,
404         R.id.task_label_bar_row_5,
405         R.id.task_label_bar_row_6,
406         R.id.task_label_bar_row_7,
407     )
408 }
409 }
410
```

Исходный файл app/src/main/res/drawable/ic_launcher_background.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <vector xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:aapt="http://schemas.android.com/aapt"
4     android:width="108dp"
5     android:height="108dp"
6     android:viewportWidth="108"
7     android:viewportHeight="108">
8     <path android:pathData="M0,0h108v108h-108z">
9         <aapt:attr name="android:fillColor">
10             <gradient
11                 android:endX="92"
12                 android:endY="100"
13                 android:startX="16"
14                 android:startY="8"
15                 android:type="linear">
16                 <item
17                     android:color="@color/launcher_background"
18                     android:offset="0" />
19                 <item
20                     android:color="@color/launcher_background_dark"
21                     android:offset="1" />
22             </gradient>
23         </aapt:attr>
24     </path>
25 </vector>
26
```

Исходный файл app/src/main/res/drawable/ic_launcher_foreground.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <vector xmlns:android="http://schemas.android.com/apk/res/android"
3      android:width="108dp"
4      android:height="108dp"
5      android:viewportWidth="108"
6      android:viewportHeight="108">
7      <!-- Shackle -->
8      <path
9          android:fillColor="#FFFFFF"
10         android:pathData="M44,50 C44,40 54,35 54,35 C64,35 64,50 64,50 L60,50 C60,43 54,39
11 54,39 C48,39 44,43 44,50 Z" />
12     <!-- Lock body: three bricks -->
13     <path
14         android:fillColor="#FFFFFF"
15         android:pathData="M42,52 H66 V59 H42 Z" />
16     <path
17         android:fillColor="#FFFFFF"
18         android:pathData="M42,61 H66 V68 H42 Z" />
19     <path
20         android:fillColor="#FFFFFF"
21         android:pathData="M42,70 H66 V78 C66,80 64,82 54,82 C44,82 42,80 42,78 Z" />
22     <!-- Side wall bricks -->
23     <path
24         android:fillColor="#FFFFFF"
25         android:pathData="M30,56 H38 V63 H30 Z" />
26     <path
27         android:fillColor="#FFFFFF"
28         android:pathData="M30,66 H38 V73 H30 Z" />
29     <path
30         android:fillColor="#FFFFFF"
31         android:pathData="M70,56 H78 V63 H70 Z" />
32     <path
33         android:fillColor="#FFFFFF"
34         android:pathData="M70,66 H78 V73 H70 Z" />
35 </vector>
```

Исходный файл app/src/main/res/drawable/ic_launcher_monochrome.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <vector xmlns:android="http://schemas.android.com/apk/res/android"
3     android:width="108dp"
4     android:height="108dp"
5     android:viewportWidth="108"
6     android:viewportHeight="108">
7     <path
8         android:fillColor="#FFFFFF"
9         android:pathData="M44,50 C44,40 54,35 54,35 C64,35 64,50 64,50 L60,50 C60,43 54,39
10        54,39 C48,39 44,43 44,50 Z" />
11     <path
12         android:fillColor="#FFFFFF"
13         android:pathData="M42,52 H66 V59 H42 Z" />
14     <path
15         android:fillColor="#FFFFFF"
16         android:pathData="M42,61 H66 V68 H42 Z" />
17     <path
18         android:fillColor="#FFFFFF"
19         android:pathData="M42,70 H66 V78 C66,80 64,82 54,82 C44,82 42,80 42,78 Z" />
20     <path
21         android:fillColor="#FFFFFF"
22         android:pathData="M30,56 H38 V63 H30 Z" />
23     <path
24         android:fillColor="#FFFFFF"
25         android:pathData="M30,66 H38 V73 H30 Z" />
26     <path
27         android:fillColor="#FFFFFF"
28         android:pathData="M70,56 H78 V63 H70 Z" />
29     <path
30         android:fillColor="#FFFFFF"
31         android:pathData="M70,66 H78 V73 H70 Z" />
32 </vector>
```


Исходный файл app/src/main/res/drawable/ic_stat_wallenc.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <vector xmlns:android="http://schemas.android.com/apk/res/android"
3      android:width="24dp"
4      android:height="24dp"
5      android:viewportWidth="24"
6      android:viewportHeight="24">
7      <path
8          android:fillColor="#FFFFFFFF"
9          android:pathData="M9,10 C9,8 10.5,7 12,7 C13.5,7 15,8 15,10 L14,10 C14,8.9 13.1,8
10         12,8 C10.9,8 10,8.9 10,10 Z" />
11      <path
12          android:fillColor="#FFFFFFFF"
13          android:pathData="M9,11 H15 V13 H9 Z" />
14      <path
15          android:fillColor="#FFFFFFFF"
16          android:pathData="M9,14 H15 V16 H9 Z" />
17      <path
18          android:fillColor="#FFFFFFFF"
19          android:pathData="M9,17 H15 V19 C15,19.6 14.6,20 12,20 C9.4,20 9,19.6 9,19 Z" />
20  </vector>
```

Исходный файл app/src/main/res/layout/notification_wallenc_tasks_big.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="wrap_content"
5      android:orientation="vertical"
6      android:paddingStart="16dp"
7      android:paddingEnd="16dp"
8      android:paddingBottom="8dp">
9
10     <LinearLayout
11         android:id="@+id/task_row_0"
12         android:layout_width="match_parent"
13         android:layout_height="wrap_content"
14         android:orientation="vertical"
15         android:paddingTop="4dp"
16         android:visibility="gone">
17
18         <TextView
19             android:id="@+id/task_title_0"
20             android:layout_width="match_parent"
21             android:layout_height="wrap_content"
22             android:ellipsize="end"
23             android:maxLines="2"
24             android:textSize="13sp" />
25
26         <LinearLayout
27             android:id="@+id/task_label_bar_row_0"
28             android:layout_width="match_parent"
29             android:layout_height="wrap_content"
30             android:layout_marginTop="2dp"
31             android:gravity="center_vertical"
32             android:orientation="horizontal">
33
34             <TextView
35                 android:id="@+id/task_subtitle_0"
36                 android:layout_width="0dp"
37                 android:layout_height="wrap_content"
38                 android:layout_weight="1"
39                 android:ellipsize="end"
40                 android:maxLines="2"
41                 android:textSize="11sp" />
42
43             <ProgressBar
```

```

44         android:id="@+id/task_progress_0"
45         style="?android:attr/progressBarStyleHorizontal"
46         android:layout_width="112dp"
47         android:layout_height="8dp"
48         android:layout_marginStart="2dp" />
49     </LinearLayout>
50 </LinearLayout>
51
52 <LinearLayout
53     android:id="@+id/task_row_1"
54     android:layout_width="match_parent"
55     android:layout_height="wrap_content"
56     android:orientation="vertical"
57     android:paddingTop="4dp"
58     android:visibility="gone">
59
60     <TextView
61         android:id="@+id/task_title_1"
62         android:layout_width="match_parent"
63         android:layout_height="wrap_content"
64         android:ellipsize="end"
65         android:maxLines="2"
66         android:textSize="13sp" />
67
68     <LinearLayout
69         android:id="@+id/task_label_bar_row_1"
70         android:layout_width="match_parent"
71         android:layout_height="wrap_content"
72         android:layout_marginTop="2dp"
73         android:gravity="center_vertical"
74         android:orientation="horizontal">
75
76         <TextView
77             android:id="@+id/task_subtitle_1"
78             android:layout_width="0dp"
79             android:layout_height="wrap_content"
80             android:layout_weight="1"
81             android:ellipsize="end"
82             android:maxLines="2"
83             android:textSize="11sp" />
84
85         <ProgressBar
86             android:id="@+id/task_progress_1"
87             style="?android:attr/progressBarStyleHorizontal"
88             android:layout_width="112dp"

```

```

89         android:layout_height="8dp"
90         android:layout_marginStart="2dp" />
91     </LinearLayout>
92 </LinearLayout>
93
94 <LinearLayout
95     android:id="@+id/task_row_2"
96     android:layout_width="match_parent"
97     android:layout_height="wrap_content"
98     android:orientation="vertical"
99     android:paddingTop="4dp"
100     android:visibility="gone">
101
102     <TextView
103         android:id="@+id/task_title_2"
104         android:layout_width="match_parent"
105         android:layout_height="wrap_content"
106         android:ellipsize="end"
107         android:maxLines="2"
108         android:textSize="13sp" />
109
110     <LinearLayout
111         android:id="@+id/task_label_bar_row_2"
112         android:layout_width="match_parent"
113         android:layout_height="wrap_content"
114         android:layout_marginTop="2dp"
115         android:gravity="center_vertical"
116         android:orientation="horizontal">
117
118         <TextView
119             android:id="@+id/task_subtitle_2"
120             android:layout_width="0dp"
121             android:layout_height="wrap_content"
122             android:layout_weight="1"
123             android:ellipsize="end"
124             android:maxLines="2"
125             android:textSize="11sp" />
126
127         <ProgressBar
128             android:id="@+id/task_progress_2"
129             style="?android:attr/progressBarStyleHorizontal"
130             android:layout_width="112dp"
131             android:layout_height="8dp"
132             android:layout_marginStart="2dp" />
133     </LinearLayout>

```

```

134     </LinearLayout>
135
136     <LinearLayout
137         android:id="@+id/task_row_3"
138         android:layout_width="match_parent"
139         android:layout_height="wrap_content"
140         android:orientation="vertical"
141         android:paddingTop="4dp"
142         android:visibility="gone">
143
144         <TextView
145             android:id="@+id/task_title_3"
146             android:layout_width="match_parent"
147             android:layout_height="wrap_content"
148             android:ellipsize="end"
149             android:maxLines="2"
150             android:textSize="13sp" />
151
152         <LinearLayout
153             android:id="@+id/task_label_bar_row_3"
154             android:layout_width="match_parent"
155             android:layout_height="wrap_content"
156             android:layout_marginTop="2dp"
157             android:gravity="center_vertical"
158             android:orientation="horizontal">
159
160             <TextView
161                 android:id="@+id/task_subtitle_3"
162                 android:layout_width="0dp"
163                 android:layout_height="wrap_content"
164                 android:layout_weight="1"
165                 android:ellipsize="end"
166                 android:maxLines="2"
167                 android:textSize="11sp" />
168
169             <ProgressBar
170                 android:id="@+id/task_progress_3"
171                 style="?android:attr/progressBarStyleHorizontal"
172                 android:layout_width="112dp"
173                 android:layout_height="8dp"
174                 android:layout_marginStart="2dp" />
175         </LinearLayout>
176     </LinearLayout>
177
178     <LinearLayout

```

```

179         android:id="@+id/task_row_4"
180         android:layout_width="match_parent"
181         android:layout_height="wrap_content"
182         android:orientation="vertical"
183         android:paddingTop="4dp"
184         android:visibility="gone">
185
186         <TextView
187             android:id="@+id/task_title_4"
188             android:layout_width="match_parent"
189             android:layout_height="wrap_content"
190             android:ellipsize="end"
191             android:maxLines="2"
192             android:textSize="13sp" />
193
194         <LinearLayout
195             android:id="@+id/task_label_bar_row_4"
196             android:layout_width="match_parent"
197             android:layout_height="wrap_content"
198             android:layout_marginTop="2dp"
199             android:gravity="center_vertical"
200             android:orientation="horizontal">
201
202             <TextView
203                 android:id="@+id/task_subtitle_4"
204                 android:layout_width="0dp"
205                 android:layout_height="wrap_content"
206                 android:layout_weight="1"
207                 android:ellipsize="end"
208                 android:maxLines="2"
209                 android:textSize="11sp" />
210
211             <ProgressBar
212                 android:id="@+id/task_progress_4"
213                 style="?android:attr/progressBarStyleHorizontal"
214                 android:layout_width="112dp"
215                 android:layout_height="8dp"
216                 android:layout_marginStart="2dp" />
217         </LinearLayout>
218     </LinearLayout>
219
220     <LinearLayout
221         android:id="@+id/task_row_5"
222         android:layout_width="match_parent"
223         android:layout_height="wrap_content"

```

```

224         android:orientation="vertical"
225         android:paddingTop="4dp"
226         android:visibility="gone">
227
228         <TextView
229             android:id="@+id/task_title_5"
230             android:layout_width="match_parent"
231             android:layout_height="wrap_content"
232             android:ellipsize="end"
233             android:maxLines="2"
234             android:textSize="13sp" />
235
236         <LinearLayout
237             android:id="@+id/task_label_bar_row_5"
238             android:layout_width="match_parent"
239             android:layout_height="wrap_content"
240             android:layout_marginTop="2dp"
241             android:gravity="center_vertical"
242             android:orientation="horizontal">
243
244             <TextView
245                 android:id="@+id/task_subtitle_5"
246                 android:layout_width="0dp"
247                 android:layout_height="wrap_content"
248                 android:layout_weight="1"
249                 android:ellipsize="end"
250                 android:maxLines="2"
251                 android:textSize="11sp" />
252
253             <ProgressBar
254                 android:id="@+id/task_progress_5"
255                 style="?android:attr/progressBarStyleHorizontal"
256                 android:layout_width="112dp"
257                 android:layout_height="8dp"
258                 android:layout_marginStart="2dp" />
259         </LinearLayout>
260     </LinearLayout>
261
262     <LinearLayout
263         android:id="@+id/task_row_6"
264         android:layout_width="match_parent"
265         android:layout_height="wrap_content"
266         android:orientation="vertical"
267         android:paddingTop="4dp"
268         android:visibility="gone">

```

```

269
270     <TextView
271         android:id="@+id/task_title_6"
272         android:layout_width="match_parent"
273         android:layout_height="wrap_content"
274         android:ellipsize="end"
275         android:maxLines="2"
276         android:textSize="13sp" />
277
278     <LinearLayout
279         android:id="@+id/task_label_bar_row_6"
280         android:layout_width="match_parent"
281         android:layout_height="wrap_content"
282         android:layout_marginTop="2dp"
283         android:gravity="center_vertical"
284         android:orientation="horizontal">
285
286         <TextView
287             android:id="@+id/task_subtitle_6"
288             android:layout_width="0dp"
289             android:layout_height="wrap_content"
290             android:layout_weight="1"
291             android:ellipsize="end"
292             android:maxLines="2"
293             android:textSize="11sp" />
294
295         <ProgressBar
296             android:id="@+id/task_progress_6"
297             style="?android:attr/progressBarStyleHorizontal"
298             android:layout_width="112dp"
299             android:layout_height="8dp"
300             android:layout_marginStart="2dp" />
301     </LinearLayout>
302 </LinearLayout>
303
304 <LinearLayout
305     android:id="@+id/task_row_7"
306     android:layout_width="match_parent"
307     android:layout_height="wrap_content"
308     android:orientation="vertical"
309     android:paddingTop="4dp"
310     android:visibility="gone">
311
312     <TextView
313         android:id="@+id/task_title_7"

```



```

314         android:layout_width="match_parent"
315         android:layout_height="wrap_content"
316         android:ellipsize="end"
317         android:maxLines="2"
318         android:textSize="13sp" />
319
320     <LinearLayout
321         android:id="@+id/task_label_bar_row_7"
322         android:layout_width="match_parent"
323         android:layout_height="wrap_content"
324         android:layout_marginTop="2dp"
325         android:gravity="center_vertical"
326         android:orientation="horizontal">
327
328         <TextView
329             android:id="@+id/task_subtitle_7"
330             android:layout_width="0dp"
331             android:layout_height="wrap_content"
332             android:layout_weight="1"
333             android:ellipsize="end"
334             android:maxLines="2"
335             android:textSize="11sp" />
336
337         <ProgressBar
338             android:id="@+id/task_progress_7"
339             style="?android:attr/progressBarStyleHorizontal"
340             android:layout_width="112dp"
341             android:layout_height="8dp"
342             android:layout_marginStart="2dp" />
343     </LinearLayout>
344 </LinearLayout>
345 </LinearLayout>
346

```

Исходный файл app/src/main/res/mipmap-anydpi/ic_launcher.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
3     <background android:drawable="@drawable/ic_launcher_background" />
4     <foreground android:drawable="@drawable/ic_launcher_foreground" />
5     <monochrome android:drawable="@drawable/ic_launcher_monochrome" />
6 </adaptive-icon>
```

Исходный файл app/src/main/res/mipmap-anydpi/ic_launcher_round.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
3     <background android:drawable="@drawable/ic_launcher_background" />
4     <foreground android:drawable="@drawable/ic_launcher_foreground" />
5     <monochrome android:drawable="@drawable/ic_launcher_monochrome" />
6 </adaptive-icon>
```

Исходный файл app/src/main/res/values-night/colors.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <!-- Тёмная тема: фон splash и окна до отрисовки Compose -->
4      <color name="splash_screen_background">#FF1C1B1F</color>
5  </resources>
6
```

Исходный файл app/src/main/res/values-ru/plurals.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <plurals name="task_notification_group_subtext">
4          <item quantity="one">Выполняется %d задача</item>
5          <item quantity="few">Выполняется %d задачи</item>
6          <item quantity="many">Выполняется %d задач</item>
7          <item quantity="other">Выполняется %d задач</item>
8      </plurals>
9      <plurals name="task_notification_more_tasks">
10         <item quantity="one">ещё %d</item>
11         <item quantity="few">ещё %d</item>
12         <item quantity="many">ещё %d</item>
13         <item quantity="other">ещё %d</item>
14     </plurals>
15 </resources>
16
```

Исходный файл app/src/main/res/values-ru/strings.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <string name="app_name">Wallenc</string>
4      <string name="task_notification_channel_name">Фоновые задачи</string>
5      <string name="task_notification_title">Задачи Wallenc</string>
6      <string name="task_notification_preparing">Подготовка...</string>
7      <string name="task_notification_indeterminate">Выполняется...</string>
8      <string name="task_notification_cancel">Отмена</string>
9      <string name="fgs_task_pipeline_description">Показывает прогресс фоновых задач хранилищ
и vault.</string>
10 </resources>
11
```

Исходный файл app/src/main/res/values/colors.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <color name="splash_screen_background">#FFF7F2FF</color>
4      <color name="launcher_background">#FF6650A4</color>
5      <color name="launcher_background_dark">#FF4A3F7A</color>
6  </resources>
7
```

Исходный файл app/src/main/res/values/plurals.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <plurals name="task_notification_group_subtext">
4          <item quantity="one">%d task running</item>
5          <item quantity="other">%d tasks running</item>
6      </plurals>
7      <plurals name="task_notification_more_tasks">
8          <item quantity="one">+%d more</item>
9          <item quantity="other">+%d more</item>
10     </plurals>
11 </resources>
12
```


Исходный файл app/src/main/res/values/strings.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <string name="app_name">Wallenc</string>
4      <string name="task_notification_channel_name">Background tasks</string>
5      <string name="task_notification_title">Wallenc tasks</string>
6      <string name="task_notification_preparing">Preparing...</string>
7      <string name="task_notification_indeterminate">Running...</string>
8      <string name="task_notification_cancel">Cancel</string>
9      <string name="fgs_task_pipeline_description">Shows progress while storage and vault
10 tasks run in the background.</string>
11 </resources>
```

Исходный файл app/src/main/res/values/themes.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources xmlns:tools="http://schemas.android.com/tools">
3
4      <style name="Theme.Wallenc" parent="Theme.AppCompat.Light.NoActionBar">
5          <!-- До первого кадра Compose и системный splash (12+) -->
6          <item name="android:windowBackground">@color/splash_screen_background</item>
7          <item name="android:windowSplashScreenBackground" tools:targetApi="31">
8              @color/splash_screen_background
9          </item>
10         <item name="android:windowSplashScreenAnimatedIcon" tools:targetApi="31">
11             @drawable/ic_launcher_foreground
12         </item>
13         <item name="android:windowSplashScreenIconBackgroundColor" tools:targetApi="31">
14             @color/launcher_background
15         </item>
16     </style>
17 </resources>
18
```

Исходный файл app/src/main/res/xml/backup_rules.xml

```
1  <?xml version="1.0" encoding="utf-8"?><!--
2      Sample backup rules file; uncomment and customize as necessary.
3      See https://developer.android.com/guide/topics/data/autobackup
4      for details.
5      Note: This file is ignored for devices older than API 31
6      See https://developer.android.com/about/versions/12/backup-restore
7  -->
8  <full-backup-content>
9      <!--
10         <include domain="sharedpref" path="."/>
11         <exclude domain="sharedpref" path="device.xml"/>
12     -->
13 </full-backup-content>
```

Исходный файл app/src/main/res/xml/data_extraction_rules.xml

```
1  <?xml version="1.0" encoding="utf-8"?><!--
2      Sample data extraction rules file; uncomment and customize as necessary.
3      See https://developer.android.com/about/versions/12/backup-restore#xml-changes
4      for details.
5  -->
6  <data-extraction-rules>
7      <cloud-backup>
8          <!-- TODO: Use <include> and <exclude> to control what is backed up.
9              <include .../>
10             <exclude .../>
11         -->
12     </cloud-backup>
13     <!--
14     <device-transfer>
15         <include .../>
16         <exclude .../>
17     </device-transfer>
18     -->
19 </data-extraction-rules>
```

Исходный файл app/src/main/res/xml/locales_config.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <locale-config xmlns:android="http://schemas.android.com/apk/res/android">
3      <locale android:name="en" />
4      <locale android:name="ru" />
5  </locale-config>
6
```

ПРИЛОЖЕНИЕ А.3

Модуль :domain

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
common/impl/CommonDirectory.kt

```
1 package com.github.nullptroma.wallenc.domain.common.impl
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IDirectory
4
5 data class CommonDirectory(
6     override val metaInfo: CommonMetaInfo,
7     override val elementsCount: Int?
8 ) : IDirectory
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
common/impl/CommonFile.kt

```
1 package com.github.nullptroma.wallenc.domain.common.impl
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IFile
4 import com.github.nullptroma.wallenc.domain.interfaces.IMetaInfo
5
6 data class CommonFile(override val metaInfo: IMetaInfo) : IFile
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
common/impl/CommonMetaInfo.kt

```
1 package com.github.nullptroma.wallenc.domain.common.impl
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IMetaInfo
4 import java.time.Clock
5 import java.time.Instant
6
7
8 data class CommonMetaInfo(
9     override val size: Long,
10    override val isDeleted: Boolean = false,
11    override val isHidden: Boolean = false,
12    override val lastModified: Instant = Clock.systemUTC().instant(),
13    override val path: String
14 ) : IMetaInfo
```


Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
common/impl/CommonStorageMetaInfo.kt

```
1 package com.github.nullptroma.wallenc.domain.common.impl
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
4 import com.github.nullptroma.wallenc.domain.interfaces.IStorageMetaInfo
5 import java.time.Clock
6 import java.time.Instant
7
8
9 data class CommonStorageMetaInfo(
10     override val encInfo: StorageEncryptionInfo? = null,
11     override val name: String? = null,
12     override val lastModified: Instant = Clock.systemUTC().instant()
13 ) : IStorageMetaInfo
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/DataPackage.kt

```
1 package com.github.nullptroma.wallenc.domain.datatypes
2
3 open class DataPackage<T>{
4     val data: T,
5     val isLoading: Boolean? = false,
6     val isError: Boolean? = false
7 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/DataPage.kt

```
1 package com.github.nullptroma.wallenc.domain.datatypes
2
3 class DataPage<T>{
4     list: List<T>,
5     isLoading: Boolean? = null,
6     isError: Boolean? = null,
7     val hasNext: Boolean? = null,
8     val pageLength: Int,
9     val pageIndex: Int
10 ) : DataPackage<List<T>>(data = list, isLoading = isLoading, isError = isError)
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/EncryptKey.kt

```
1  package com.github.nullptroma.wallenc.domain.datatypes
2
3  import java.security.MessageDigest
4  import javax.crypto.spec.SecretKeySpec
5
6  class EncryptKey {
7      val bytes: ByteArray
8
9      constructor(password: String) {
10         val digest = MessageDigest.getInstance("SHA-256")
11         bytes = digest.digest(password.toByteArray(Charsets.UTF_8))
12     }
13
14     constructor(key: ByteArray) {
15         this.bytes = key.clone()
16     }
17
18     fun toAesKey() : SecretKeySpec {
19         return SecretKeySpec(bytes, "AES")
20     }
21 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/StorageDomainRecords.kt

```
1 package com.github.nullptroma.wallenc.domain.datatypes
2
3 data class TwoFaTokenRecord(
4     val id: String,
5     val issuer: String,
6     val account: String,
7     val secret: String,
8     val notes: String? = null,
9     val digits: Int = 6,
10    val periodSeconds: Int = 30,
11    val algorithm: String = "SHA1",
12 )
13
14 data class TextSecretEntryRecord(
15     val label: String?,
16     val value: String,
17 )
18
19 data class TextSecretRecord(
20     val id: String,
21     val title: String,
22     val items: List<TextSecretEntryRecord>,
23 )
24
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/StorageEncryptionInfo.kt

```
1 package com.github.nullptroma.wallenc.domain.datatypes
2
3 data class StorageEncryptionInfo(
4     val encryptedTestData: String,
5     val pathIv: ByteArray?
6 ) {
7     override fun equals(other: Any?): Boolean {
8         if (this === other) return true
9         if (javaClass != other?.javaClass) return false
10
11         other as StorageEncryptionInfo
12
13         if (encryptedTestData != other.encryptedTestData) return false
14         if (!pathIv.contentEquals(other.pathIv)) return false
15
16         return true
17     }
18
19     override fun hashCode(): Int {
20         var result = encryptedTestData.hashCode()
21         result = 31 * result + pathIv.contentHashCode()
22         return result
23     }
24 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/StorageMetaLoadState.kt

```
1 package com.github.nullptroma.wallenc.domain.datatypes
2
3 enum class StorageMetaLoadState {
4     Loading,
5     Ready,
6     Unavailable,
7 }
8
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/StorageSyncJournal.kt

```
1 package com.github.nullptroma.wallenc.domain.datatypes
2
3 /** Журнал синхронизации: один актуальный [StorageSyncJournalEntry] на нормализованный путь
4 */
5 typealias StorageSyncJournal = Map<String, StorageSyncJournalEntry>
6
7 object StorageSyncPaths {
8     fun normalize(path: String): String = if (path.startsWith("/")) path else "/$path"
9
10    /**
11     * Пути, которые участвуют в sync и попадают в журнал при пользовательских операциях.
12     * Служебные каталоги, lock/journal/мета и файлы внутри *-enc-dir исключены.
13     */
14    fun isSyncableUserPath(path: String): Boolean {
15        val p = normalize(path)
16        if (p == "/" || p.isBlank()) return false
17        if (p == "/wallenc-yandex-system" || p.startsWith("/wallenc-yandex-system/")) return
18        false
19        if (p.contains("-enc-dir/") || p.endsWith("-enc-dir")) return false
20        val name = p.substringAfterLast('/')
21        if (name == "sync-journal.json" || name == "sync-lock.json") return false
22        if (name.endsWith(".enc-meta") || name.endsWith(".storage-info")) return false
23        return true
24    }
25 }
26
27 object StorageSyncJournalMerge {
28     fun merge(into: StorageSyncJournal, entries: Map<String, StorageSyncJournalEntry>):
29     StorageSyncJournal {
30         if (entries.isEmpty()) return into
31         val result = into.toMutableMap()
32         for ((rawPath, entry) in entries) {
33             val path = StorageSyncPaths.normalize(rawPath)
34             if (!StorageSyncPaths.isSyncableUserPath(path)) continue
35             val normalizedEntry = entry.copy(path = path)
36             val current = result[path]
37             if (current == null || compareEntries(normalizedEntry, current) > 0) {
38                 result[path] = normalizedEntry
39             }
40         }
41         return result
42     }
43 }
```



```

41     fun merge(into: StorageSyncJournal, entry: StorageSyncJournalEntry): StorageSyncJournal
42     =
43         merge(into, mapOf(entry.path to entry))
44
45     fun mergeAll(journals: Collection<StorageSyncJournal>): StorageSyncJournal {
46         var acc: StorageSyncJournal = emptyMap()
47         for (journal in journals) {
48             acc = merge(acc, journal)
49         }
50         return acc
51     }
52
53     private fun compareEntries(a: StorageSyncJournalEntry, b: StorageSyncJournalEntry): Int
54     {
55         val seqCmp = a.revision.sequence.compareTo(b.revision.sequence)
56         if (seqCmp != 0) return seqCmp
57         val actorCmp = a.revision.actorId.compareTo(b.revision.actorId)
58         if (actorCmp != 0) return actorCmp
59         return a.revision.createdAt.compareTo(b.revision.createdAt)
60     }

```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/StorageSyncModels.kt

```
1 package com.github.nullptroma.wallenc.domain.datatypes
2
3 import java.time.Instant
4 import java.util.UUID
5
6 enum class StorageSyncOperation {
7     UPSERT,
8     /** Soft-delete (корзина): на peer вызывается [IStorageAccessor.moveToTrash]. */
9     TRASH,
10    /** Жёсткое удаление файла с носителя. */
11    DELETE,
12 }
13
14 data class StorageSyncRevision(
15     val sequence: Long,
16     val actorId: String,
17     val createdAt: Instant,
18 )
19
20 data class StorageSyncJournalEntry(
21     val path: String,
22     val operation: StorageSyncOperation,
23     val revision: StorageSyncRevision,
24     val size: Long? = null,
25     val originStorageUuid: UUID? = null,
26 )
27
28 data class StorageSyncLock(
29     val holderId: String,
30     val leaseUntil: Instant,
31     val updatedAt: Instant,
32 )
33
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
datatypes/Tree.kt

```
1 package com.github.nullptroma.wallenc.domain.datatypes
2
3 class Tree<T> (val value: T, var children: List<Tree<T>>? = null)
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/encrypt/Encryptor.kt

```
1  package com.github.nullptroma.wallenc.domain.encrypt
2
3  import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
4  import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
5  import kotlinx.coroutines.DisposableHandle
6  import java.io.InputStream
7  import java.io.OutputStream
8  import javax.crypto.Cipher
9  import javax.crypto.CipherInputStream
10 import javax.crypto.CipherOutputStream
11 import javax.crypto.SecretKey
12 import javax.crypto.spec.IvParameterSpec
13 import kotlin.io.encoding.Base64
14 import kotlin.io.encoding.ExperimentalEncodingApi
15 import kotlin.random.Random
16
17 class Encryptor(private var secretKey: SecretKey) : DisposableHandle {
18     @OptIn(ExperimentalEncodingApi::class)
19     fun encryptString(str: String): String {
20         val bytesToEncrypt = str.toByteArray(Charsets.UTF_8)
21         val encryptedBytes = encryptBytes(bytesToEncrypt)
22         return Base64.encode(encryptedBytes).replace("/", ".")
23     }
24
25     @OptIn(ExperimentalEncodingApi::class)
26     fun decryptString(str: String): String {
27         val bytesToDecrypt = Base64.decode(str.replace(".", "/"))
28         val decryptedBytes = decryptBytes(bytesToDecrypt)
29         return String(decryptedBytes, Charsets.UTF_8)
30     }
31
32     fun encryptBytes(bytes: ByteArray): ByteArray {
33         if(secretKey.isDestroyed)
34             throw Exception("Object was destroyed")
35         val cipher = Cipher.getInstance(AES_SETTINGS)
36         val ivSpec = IvParameterSpec(Random.nextBytes(IV_LEN))
37         cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivSpec)
38         val encryptedBytes = ivSpec.iv + cipher.doFinal(bytes) // iv + зашифрованные байты
39         return encryptedBytes
40     }
41
42     fun decryptBytes(bytes: ByteArray): ByteArray {
```

```

43         if(secretKey.isDestroyed)
44             throw Exception("Object was destroyed")
45         val cipher = Cipher.getInstance(AES_SETTINGS)
46         val ivSpec = IvParameterSpec(bytes.take(IV_LEN).toByteArray())
47         cipher.init(Cipher.DECRYPT_MODE, secretKey, ivSpec)
48         val decryptedBytes = cipher.doFinal(bytes.drop(IV_LEN).toByteArray())
49         return decryptedBytes
50     }
51
52     fun encryptStream(stream: OutputStream): OutputStream {
53         if(secretKey.isDestroyed)
54             throw Exception("Object was destroyed")
55         val ivSpec = IvParameterSpec(Random.nextBytes(IV_LEN))
56         stream.write(ivSpec.iv) // Запись инициализационного вектора сырой файл
57         val cipher = Cipher.getInstance(AES_SETTINGS)
58         cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivSpec) // инициализация шифратора
59         return CipherOutputStream(stream, cipher)
60     }
61
62     fun decryptStream(stream: InputStream): InputStream {
63         if(secretKey.isDestroyed)
64             throw Exception("Object was destroyed")
65         val ivBytes = ByteArray(IV_LEN) // Буфер для 16 байт IV
66         val bytesRead = stream.read(ivBytes) // Чтение IV вектора
67         if(bytesRead != IV_LEN)
68             throw Exception("TODO iv не прочитан")
69         val ivSpec = IvParameterSpec(ivBytes)
70
71         val cipher = Cipher.getInstance(AES_SETTINGS)
72         cipher.init(Cipher.DECRYPT_MODE, secretKey, ivSpec)
73         return CipherInputStream(stream, cipher)
74     }
75
76     override fun dispose() {
77         //secretKey.destroy()
78     }
79
80     companion object {
81         const val IV_LEN = 16
82         const val AES_SETTINGS = "AES/CBC/PKCS5Padding"
83         private const val TEST_DATA_LEN = 512
84
85         @OptIn(ExperimentalEncodingApi::class)
86         fun generateEncryptionInfo(key: EncryptKey, encryptPath: Boolean = true) :
StorageEncryptionInfo {

```

```

87         val encryptor = Encryptor(key.toAesKey())
88         val testData = ByteArray(TEST_DATA_LEN)
89         val encryptedData = encryptor.encryptBytes(testData)
90         return StorageEncryptionInfo(
91             encryptedTestData = Base64.encode(encryptedData),
92             pathIv = if(encryptPath) Random.nextBytes(IV_LEN) else null
93         )
94     }
95
96     @OptIn(ExperimentalEncodingApi::class)
97     fun checkKey(key: EncryptKey, encInfo: StorageEncryptionInfo): Boolean {
98         val encryptor = Encryptor(key.toAesKey())
99         try {
100             val encData = Base64.decode(encInfo.encryptedTestData)
101             val testData = encryptor.decryptBytes(encData)
102             return testData.all { it == 0.toByte() } && testData.size == TEST_DATA_LEN
103         }
104         catch (e: Exception) {
105             return false
106         }
107     }
108 }
109 }

```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
encrypt/EncryptorWithStaticIv.kt

```
1 package com.github.nullptroma.wallenc.domain.encrypt
2
3 import com.github.nullptroma.wallenc.domain.encrypt.Encryptor.Companion.AES_SETTINGS
4 import kotlinx.coroutines.DisposableHandle
5 import javax.crypto.Cipher
6 import javax.crypto.SecretKey
7 import javax.crypto.spec.IvParameterSpec
8 import kotlin.io.encoding.Base64
9 import kotlin.io.encoding.ExperimentalEncodingApi
10
11 class EncryptorWithStaticIv(private var secretKey: SecretKey, iv: ByteArray) :
12     DisposableHandle {
13
14     private val ivSpec = IvParameterSpec(iv)
15
16     @OptIn(ExperimentalEncodingApi::class)
17     fun encryptString(str: String): String {
18         val bytesToEncrypt = str.toByteArray(Charsets.UTF_8)
19         val encryptedBytes = encryptBytes(bytesToEncrypt)
20         return Base64.encode(encryptedBytes).replace("/", ".")
21     }
22
23     @OptIn(ExperimentalEncodingApi::class)
24     fun decryptString(str: String): String {
25         val bytesToDecrypt = Base64.decode(str.replace(".", "/"))
26         val decryptedBytes = decryptBytes(bytesToDecrypt)
27         return String(decryptedBytes, Charsets.UTF_8)
28     }
29
30     fun encryptBytes(bytes: ByteArray): ByteArray {
31         if(secretKey.isDestroyed)
32             throw Exception("Object was destroyed")
33         val cipher = Cipher.getInstance(AES_SETTINGS)
34         cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivSpec)
35         val encryptedBytes = cipher.doFinal(bytes) // зашифрованные байты
36         return encryptedBytes
37     }
38
39     fun decryptBytes(bytes: ByteArray): ByteArray {
40         if(secretKey.isDestroyed)
41             throw Exception("Object was destroyed")
42         val cipher = Cipher.getInstance(AES_SETTINGS)
43         cipher.init(Cipher.DECRYPT_MODE, secretKey, ivSpec)
```

```
42         val decryptedBytes = cipher.doFinal(bytes)
43         return decryptedBytes
44     }
45
46     override fun dispose() {
47         secretKey.destroy()
48     }
49 }
```


Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
errors/WallencException.kt

```
1 package com.github.nullptroma.wallenc.domain.errors
2
3 /**
4  * Единая иерархия сбоев Wallenc. Ожидаемые бизнес-отказы (CanEncryptResult и т.п.)
5  * остаются отдельными sealed-типами; сюда попадают только операционные ошибки.
6  */
7 sealed class WallencException(
8     message: String? = null,
9     cause: Throwable? = null,
10 ) : Exception(message, cause) {
11
12     sealed class Feature : WallencException() {
13         class StorageNotFound : Feature()
14         class NeedsDecryptedView : Feature()
15         class SecretNotFound : Feature()
16         class StorageNotWritable : Feature()
17     }
18
19     sealed class Storage(cause: Throwable? = null) : WallencException(cause = cause) {
20         class NotAvailable : Storage()
21         class FileNotFound : Storage()
22         class IncorrectKey : Storage()
23         class EncInfoMissing : Storage()
24         class NotEncrypted : Storage()
25         class NotWritable : Storage()
26         class NotAFile : Storage()
27         class NotADirectory : Storage()
28         class PathIsFile : Storage()
29         class CannotWriteOverDirectory : Storage()
30         class DeleteRootForbidden : Storage()
31         class UnexpectedState : Storage()
32         data class IoFailed(override val cause: Throwable) : Storage(cause)
33     }
34
35     sealed class Auth : WallencException() {
36         class Failed : Auth()
37         class TokenMissing : Auth()
38     }
39
40     sealed class Network(cause: Throwable? = null) : WallencException(cause = cause) {
41         data class HttpFailed(
42             val operation: String,
```

```
43         val statusCode: Int,
44         override val cause: Throwable? = null,
45     ) : Network(cause)
46
47     data class IoFailed(override val cause: Throwable) : Network(cause)
48     class ResourceLocked : Network()
49     class OperationFailed : Network()
50     class OperationTimedOut : Network()
51 }
52
53 data class Unknown(override val cause: Throwable?) : WallencException(cause = cause)
54 }
55
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
errors/WallencExceptionMapping.kt

```
1 package com.github.nullptroma.wallenc.domain.errors
2
3 import java.io.FileNotFoundException
4 import java.io.IOException
5
6 fun Throwable.toWallencException(): WallencException = when (this) {
7     is WallencException -> this
8     is FileNotFoundException -> WallencException.Storage.FileNotFound()
9     is IOException -> WallencException.Network.IoFailed(this)
10    else -> WallencException.Unknown(this)
11 }
12
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/IDirectory.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 interface IDirectory {
4     val metaInfo: IMetaInfo
5     val elementsCount: Int?
6 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/IFile.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 interface IFile {
4     val metaInfo: IMetaInfo
5 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/ILogger.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 interface ILogger {
4     fun debug(tag: String, msg: String)
5 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/IMetaInfo.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 import java.time.Instant
4
5
6 interface IMetaInfo {
7     val size: Long
8     val isDeleted: Boolean
9     val isHidden: Boolean
10    val lastModified: Instant
11    val path: String
12 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/IStorage.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageMetaLoadState
5 import com.github.nullptroma.wallenc.domain.tasks.TaskProgress
6 import kotlinx.coroutines.flow.Flow
7 import kotlinx.coroutines.flow.StateFlow
8 import java.time.Instant
9 import java.util.UUID
10
11 sealed interface IStorageInfo {
12     val uuid: UUID
13     val isAvailable: StateFlow<Boolean>
14     val size: StateFlow<Long?>
15     val numberOfFiles: StateFlow<Int?>
16     val isEmpty: Flow<Boolean?>
17     val metaInfo: StateFlow<IStorageMetaInfo>
18     val metaLoadState: StateFlow<StorageMetaLoadState>
19     val isVirtualStorage: Boolean
20 }
21
22 interface IStorage: IStorageInfo {
23     val accessor: IStorageAccessor
24
25     suspend fun rename(newName: String)
26     suspend fun setEncInfo(encInfo: StorageEncryptionInfo?)
27     suspend fun clearAllContent(onProgress: suspend (TaskProgress) -> Unit = {})
28 }
29
30 interface IStorageMetaInfo {
31     val encInfo: StorageEncryptionInfo?
32     val name: String?
33     val lastModified: Instant
34 }
35
```


Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/IStorageAccessor.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 import com.github.nullptroma.wallenc.domain.datatypes.DataPage
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournal
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncLock
6 import kotlinx.coroutines.flow.Flow
7 import kotlinx.coroutines.flow.SharedFlow
8 import kotlinx.coroutines.flow.StateFlow
9 import java.io.InputStream
10 import java.io.OutputStream
11 import java.time.Instant
12
13 interface IStorageAccessor {
14     val size: StateFlow<Long?>
15     val numberOfFiles: StateFlow<Int?>
16     val isAvailable: StateFlow<Boolean>
17     val filesUpdates: SharedFlow<DataPage<IFile>>
18     val dirsUpdates: SharedFlow<DataPage<IDirectory>>
19
20     suspend fun getAllFiles(): List<IFile>
21     suspend fun getFiles(path: String): List<IFile>
22     /**
23      * Получение списка файлов в директории
24      * @param path Путь к директории
25      * @return Поток файлов
26      */
27     fun getFilesFlow(path: String): Flow<DataPage<IFile>>
28
29     suspend fun getAllDirs(): List<IDirectory>
30     suspend fun getDirs(path: String): List<IDirectory>
31     /**
32      * Получение списка директорий в директории
33      * @param path Путь к директории
34      * @return Поток директорий
35      */
36     fun getDirsFlow(path: String): Flow<DataPage<IDirectory>>
37     suspend fun getFileInfo(path: String): IFile
38     suspend fun getDirInfo(path: String): IDirectory
39     suspend fun setHidden(path: String, hidden: Boolean)
40     suspend fun touchFile(path: String)
41     suspend fun touchDir(path: String)
42     suspend fun delete(path: String, recordSyncJournal: Boolean = true)
```

```

43 suspend fun openWrite(path: String, recordSyncJournal: Boolean = true): OutputStream
44 suspend fun openRead(path: String): InputStream
45 suspend fun moveToTrash(path: String, recordSyncJournal: Boolean = true)
46
47 /**
48  * Системный sidcar-файл для логических нужд хранилища (мета, ключи и т.п.).
49  * Конкретный accessor решает, где он физически живёт, но он не должен
50  * попадать в выдачу [getFiles]/[getDirs]/[size]/[numberOfFiles].
51  */
52 suspend fun openReadSystemFile(name: String): InputStream
53 suspend fun openWriteSystemFile(name: String): OutputStream
54
55 suspend fun readSyncJournal(): StorageSyncJournal
56
57 /** Сбрасывает отложенные записи журнала на носитель (перед sync и при закрытии
storage). */
58 suspend fun flushPendingSyncJournal() = Unit
59
60 suspend fun putSyncJournalEntries(entries: StorageSyncJournal)
61
62 suspend fun readSyncLock(): StorageSyncLock?
63 suspend fun tryAcquireSyncLock(holderId: String, leaseUntil: Instant): Boolean
64 suspend fun releaseSyncLock(holderId: String)
65
66 /**
67  * Сбрасывает lock синхронизации без проверки [holderId] (снятие «залипшей» блокировки).
68  * Не использовать в обычном цикле синка — только для ручного вмешательства.
69  */
70 suspend fun forceClearSyncLock()
71 }

```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/IUnlockManager.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
4 import kotlinx.coroutines.flow.StateFlow
5 import java.util.UUID
6
7 /**
8  * Управляет виртуальными (расшифрованными) представлениями хранилищ.
9  * Не является vault-провайдером.
10 */
11 interface IUnlockManager {
12     /**
13      * Хранилища, для которых есть ключ шифрования
14      */
15     val openedStorages: StateFlow<Map<UUID, IStorage>>
16     fun getOpenedStorageKey(uuid: UUID): EncryptKey?
17
18     suspend fun open(storage: IStorage, key: EncryptKey, rememberPassword: Boolean = true):
19     IStorage
20     /** Сохранить ключ для auto-open без открытия виртуального storage. */
21     suspend fun rememberKey(storage: IStorage, key: EncryptKey)
22     suspend fun close(storage: IStorage)
23     suspend fun close(uuid: UUID)
24 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/interfaces/IVault.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
4 import kotlinx.coroutines.flow.StateFlow
5
6 /**
7  * Контракт хранилища ([IVault]): коллекция [IStorage] с реактивным состоянием.
8  *
9  * Слой domain не различает локальные/удалённые/Yandex и т.д. – это общий порт.
10 */
11 interface IVault : IVaultInfo {
12     val storages: StateFlow<List<IStorage>>
13     /**
14      * Идёт загрузка/пересканирование списка storages (например, листинг удалённого vault и
15      * init каждого storage).
16      */
17     val storagesScanInProgress: StateFlow<Boolean>
18     val isAvailable: StateFlow<Boolean>
19     val totalSpace: StateFlow<Long?>
20     val availableSpace: StateFlow<Long?>
21
22     suspend fun createStorage(): IStorage
23     suspend fun createStorage(enc: StorageEncryptionInfo): IStorage
24     suspend fun remove(storage: IStorage)
25
26     /** Пересканировать список storages (для удалённых vault – повторный листинг и init). */
27     suspend fun rescanStorages() {}
28 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/IVaultInfo.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 import java.util.UUID
4
5 /**
6  * Минимальная идентификация хранилища ([IVaultInfo]).
7  *
8  * Намеренно «голая»: доменный слой не знает о брендах, локальности или статусе –
9  * вся категоризация лежит во внешнем кольце (':vault-api: VaultDescriptor').
10 */
11 interface IVaultInfo {
12     val uuid: UUID
13 }
14
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/IVaultsManager.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 import kotlinx.coroutines.flow.StateFlow
4
5 /**
6  * Единая точка доступа ко всем подключённым хранилищам приложения.
7  *
8  * Доменный слой не различает категории vault – потребители в UI (presentation)
9  * фильтруют [vaults] через `:vault-api` (`VaultDescriptor` / `DescribedVault`).
10 */
11 interface IVaultsManager {
12     val vaults: StateFlow<List<IVault>>
13     val allStorages: StateFlow<List<IStorage>>
14     val unlockManager: IUnlockManager
15 }
16
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
interfaces/StorageSyncContracts.kt

```
1 package com.github.nullptroma.wallenc.domain.interfaces
2
3 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
4 import java.util.UUID
5
6 enum class StorageSyncGroupEncryptionKind {
7     UNSET,
8     NONE,
9     PASSWORD,
10 }
11
12 data class StorageSyncGroup(
13     val id: String,
14     val storageUuids: Set<UUID>,
15     val encryptionKind: StorageSyncGroupEncryptionKind =
16         StorageSyncGroupEncryptionKind.UNSET,
17     /** Локально сохранённый секрет группы для сопоставления совместимости (не уходит
18         наружу). */
19     val encryptionSecret: String? = null,
20 )
21
22 interface IStorageSyncGroupStore {
23     suspend fun getGroups(): List<StorageSyncGroup>
24     suspend fun putGroup(group: StorageSyncGroup)
25     suspend fun removeGroup(groupId: String)
26 }
27
28 interface IStorageSyncEngine {
29     suspend fun syncAllGroups(
30         reportProgress: (suspend (fraction: Float?, label: TaskProgressLabel?) -> Unit)? =
31         null,
32     )
33     suspend fun syncGroup(
34         groupId: String,
35         reportProgress: (suspend (fraction: Float?, label: TaskProgressLabel?) -> Unit)? =
36         null,
37     )
38 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/IStorageSyncTaskTitleFormatter.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 /** Локализованный заголовок задачи синхронизации хранилищ по источнику запуска. */
4 fun interface IStorageSyncTaskTitleFormatter {
5     fun format(reason: StorageSyncTriggerReason): String
6 }
7
```


Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/ITaskOrchestrator.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 import kotlinx.coroutines.flow.StateFlow
4 import kotlinx.coroutines.CoroutineDispatcher
5 import java.util.UUID
6
7 interface ITaskOrchestrator {
8     val pipelineState: StateFlow<PipelineState>
9     val logLines: StateFlow<List<TaskLogLine>>
10    val foregroundUi: StateFlow<TaskForegroundUiState>
11
12    fun enqueue(
13        title: String,
14        dispatcher: CoroutineDispatcher,
15        work: PipelineWork,
16        busyStorageUuid: UUID? = null,
17        locksVaultStorageList: Boolean = false,
18    ): TaskId
19
20    fun cancel(taskId: TaskId): Boolean
21
22    fun cancelAll()
23
24    /** Запись в общий лог пайплайна вне контекста [TaskContext] (например, WorkManager
25    sync). */
26    fun appendPipelineLog(level: TaskLogLevel, key: TaskLogKey)
27 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/PipelineState.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 data class PipelineState(
4     val tasks: List<PipelineTask>,
5     val runningTaskIds: Set<TaskId>,
6 )
7
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/PipelineTask.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 import kotlinx.coroutines.CoroutineDispatcher
4 import java.util.UUID
5
6 data class PipelineTask(
7     val id: TaskId,
8     val title: String,
9     val enqueuedAtMs: Long,
10    val dispatcher: CoroutineDispatcher,
11    val state: TaskRunState,
12    /** UUID storage, для которого идёт задача (кнопки только этой строки в UI). */
13    val busyStorageUuid: UUID? = null,
14    /** Задача меняет список storages в vault (например создание) – блокируем FAB «+». */
15    val locksVaultStorageList: Boolean = false,
16 )
17
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/PipelineWork.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 fun interface PipelineWork {
4     suspend fun run(ctx: TaskContext)
5 }
6
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/StorageSyncTriggerReason.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 /** Источник запуска синхронизации хранилищ (для логов пайплайна задач). */
4 enum class StorageSyncTriggerReason {
5     Debounce,
6     SyncTab,
7     Background,
8 }
9
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskContext.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4
5 interface TaskContext {
6     val taskId: TaskId
7
8     suspend fun reportProgress(fraction: Float?, label: TaskProgressLabel? = null)
9
10    suspend fun reportProgress(progress: TaskProgress) = reportProgress(progress.fraction,
11    progress.label)
12
13    fun log(level: TaskLogLevel, message: String)
14
15    fun log(level: TaskLogLevel, key: TaskLogKey)
16
17    fun fail(error: WallencException): Nothing
18
19    /** Проверяет, что задача не отменена; бросает
20    [kotlinx.coroutines.CancellationException] при отмене. */
21    suspend fun ensureNotCancelled()
22 }
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskForegroundUiState.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 data class TaskForegroundItem(
4     val taskId: TaskId,
5     val title: String,
6     val progress: TaskProgress?,
7 )
8
9 sealed class TaskForegroundUiState {
10     data object Hidden : TaskForegroundUiState()
11     data class Visible(
12         val tasks: List<TaskForegroundItem>,
13     ) : TaskForegroundUiState()
14 }
15
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskId.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 import java.util.UUID
4
5 data class TaskId(val uuid: UUID = UUID.randomUUID())
6
```


Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskLogKey.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4
5 sealed class TaskLogKey {
6     data class SyncStarted(val reason: StorageSyncTriggerReason) : TaskLogKey()
7     data class SyncFinished(val reason: StorageSyncTriggerReason) : TaskLogKey()
8     data class SyncFailed(
9         val error: WallencException,
10        val reason: StorageSyncTriggerReason,
11    ) : TaskLogKey()
12 }
13
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskLogLevel.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 enum class TaskLogLevel {
4     Debug,
5     Info,
6     Warn,
7     Error,
8 }
9
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskLogLine.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 data class TaskLogLine(
4     val timestampMs: Long,
5     val level: TaskLogLevel,
6     val message: String = "",
7     val logKey: TaskLogKey? = null,
8 )
9
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskProgress.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 data class TaskProgress(
4     /** 0f..1f or null if indeterminate */
5     val fraction: Float?,
6     val label: TaskProgressLabel? = null,
7 )
8
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskProgressLabel.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 sealed class TaskProgressLabel {
4     data object SyncNoGroups : TaskProgressLabel()
5     data object SyncStarted : TaskProgressLabel()
6     data object SyncCompleted : TaskProgressLabel()
7     data class SyncPreparing(val groupCount: Int) : TaskProgressLabel()
8
9     data class SyncGroupPreparing(val groupId: String) : TaskProgressLabel()
10    data class SyncGroupNotFound(val groupId: String) : TaskProgressLabel()
11    data class SyncGroupSkippedTooFewStorages(val groupId: String) : TaskProgressLabel()
12    data class SyncGroupSkippedIncompatibleEncryption(val groupId: String, val count: Int) :
TaskProgressLabel()
13    data class SyncGroupAcquiringLocks(val groupId: String) : TaskProgressLabel()
14    data class SyncGroupLockProgress(val groupId: String, val current: Int, val total:
Int) : TaskProgressLabel()
15    data class SyncGroupLockFailed(val groupId: String) : TaskProgressLabel()
16    data class SyncGroupReadingJournals(val groupId: String) : TaskProgressLabel()
17    data class SyncGroupCancelled(val groupId: String) : TaskProgressLabel()
18    data class SyncGroupJournalProgress(val groupId: String, val current: Int, val total:
Int) : TaskProgressLabel()
19    data class SyncGroupNoJournalEntries(val groupId: String) : TaskProgressLabel()
20    data class SyncGroupProcessingEntries(val groupId: String, val count: Int) :
TaskProgressLabel()
21    data class SyncGroupEntryProgress(val groupId: String, val current: Int, val total:
Int) : TaskProgressLabel()
22    data class SyncGroupCompleted(val groupId: String) : TaskProgressLabel()
23    data class SyncGroupEntriesFailed(val groupId: String, val failedCount: Int) :
TaskProgressLabel()
24    data class SyncGroupRenewingLocks(val groupId: String) : TaskProgressLabel()
25    data class SyncGroupLockRenewalFailed(val groupId: String) : TaskProgressLabel()
26
27    data class ClearContentProgress(val done: Int, val total: Int) : TaskProgressLabel()
28
29    data class VaultTask(val step: VaultTaskStep) : TaskProgressLabel()
30
31    data class TestElapsed(val elapsedSec: Int, val totalSec: Int) : TaskProgressLabel()
32 }
33
34 enum class VaultTaskStep {
35     DumpStorageLog,
36     CreateStorage,
37     EnableEncryption,
38     DecryptRunning,
```

```
39         CloseStorage,
40         DisableEncryption,
41         RenameStorage,
42         RemoveStorage,
43         ClearSyncLock,
44         AddRemoteVault,
45         RemoveRemoteVault,
46         RetryRemoteVault,
47         RescanVaultStorages,
48         Save2FaToken,
49         Delete2FaToken,
50         SaveTextSecret,
51         DeleteTextSecret,
52     }
53
```

Исходный файл domain/src/main/java/com/github/nullptroma/wallenc/domain/
tasks/TaskRunState.kt

```
1 package com.github.nullptroma.wallenc.domain.tasks
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4
5 sealed class TaskRunState {
6     data object Queued : TaskRunState()
7     data class Running(val progress: TaskProgress?) : TaskRunState()
8     data object Completed : TaskRunState()
9     data object Cancelled : TaskRunState()
10    data class Failed(val error: WallencException) : TaskRunState()
11 }
12
```

Исходный файл domain/src/test/java/com/github/nullptroma/wallenc/domain/ EncryptorTest.kt

```
1 package com.github.nullptroma.wallenc.domain
2
3 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
4 import com.github.nullptroma.wallenc.domain.encrypt.Encryptor
5 import junit.framework.TestCase.assertEquals
6 import junit.framework.TestCase.fail
7 import org.hamcrest.MatcherAssert.assertThat
8 import org.hamcrest.core.Is
9 import org.hamcrest.core.IsEqual
10 import org.hamcrest.core.IsNot
11 import org.junit.Assert.assertArrayEquals
12 import org.junit.Test
13 import java.io.ByteArrayInputStream
14 import java.io.ByteArrayOutputStream
15 import java.util.Random
16
17 class EncryptorTest {
18     val key1 = EncryptKey("key1")
19     val key2 = EncryptKey("key2")
20     val rnd = Random()
21
22     @Test
23     fun `test correct key for StorageEncryptionInfo`() {
24         val encInfo = Encryptor.generateEncryptionInfo(key1)
25         val res = Encryptor.checkKey(key = key1, encInfo = encInfo)
26         assertEquals(true, res)
27     }
28
29     @Test
30     fun `test incorrect key for StorageEncryptionInfo`() {
31         val encInfo = Encryptor.generateEncryptionInfo(key1)
32         val res = Encryptor.checkKey(key = key2, encInfo = encInfo)
33         assertEquals(false, res)
34     }
35
36     @Test
37     fun `test string encryption with the same key`() {
38         val text = "Hello world, my name is Wallenc!"
39         val encryptor = Encryptor(key1.toAesKey())
40         val encryptedText = encryptor.encryptString(text)
41
42         val newEncryptor = Encryptor(key1.toAesKey())
```



```

43         val decryptedText = newEncryptor.decryptString(encryptedText)
44
45         assertEquals(text, decryptedText)
46     }
47
48     @Test
49     fun `test string encryption with the wrong key`() {
50         val text = "Hello world, my name is Wallenc!"
51         val encryptor = Encryptor(key1.toAesKey())
52         val encryptedText = encryptor.encryptString(text)
53
54         val newEncryptor = Encryptor(key2.toAesKey())
55         try {
56             newEncryptor.decryptString(encryptedText)
57             fail("There is not exception on decrypt with wrong key")
58         }
59         catch (e: Exception) {
60             // good
61         }
62     }
63
64     @Test
65     fun `test bytes encryption with the same key`() {
66         val bytes = ByteArray(512)
67         rnd.nextBytes(bytes)
68         val encryptor = Encryptor(key1.toAesKey())
69         val encryptedBytes = encryptor.encryptBytes(bytes)
70
71         val newEncryptor = Encryptor(key1.toAesKey())
72         val decryptedBytes = newEncryptor.decryptBytes(encryptedBytes)
73
74         assertThat(bytes, IsNot.not(IsEqual.equalTo(encryptedBytes)))
75         assertThat(bytes, Is.`is`(IsEqual.equalTo(decryptedBytes)))
76     }
77
78     @Test
79     fun `test bytes encryption with the wrong key`() {
80         val bytes = ByteArray(512)
81         rnd.nextBytes(bytes)
82         val encryptor = Encryptor(key1.toAesKey())
83         val encryptedBytes = encryptor.encryptBytes(bytes)
84
85         val newEncryptor = Encryptor(key2.toAesKey())
86         try {
87             newEncryptor.decryptBytes(encryptedBytes)

```

```

88         fail("There is not exception on decrypt with wrong key")
89     }
90     catch (e: Exception) {
91         // good
92     }
93 }
94
95 @Test
96 fun `test stream encryption with the same key`() {
97     val dataLen = 1500
98     val origData = ByteArray(dataLen)
99     rnd.nextBytes(origData)
100    val encryptor = Encryptor(key1.toAesKey())
101
102    val streamForEncrypt = ByteArrayOutputStream(dataLen*3)
103    val encryptedStream = encryptor.encryptStream(streamForEncrypt)
104    encryptedStream.write(origData)
105    encryptedStream.close()
106    val encryptedData = streamForEncrypt.toByteArray()
107
108    val newEncryptor = Encryptor(key1.toAesKey())
109    val streamForDecrypt = ByteArrayInputStream(encryptedData)
110    val decryptedStream = newEncryptor.decryptStream(streamForDecrypt)
111    val decryptedData = decryptedStream.readAllBytes()
112
113    assertEquals(origData, decryptedData)
114 }
115
116 @Test
117 fun `test stream encryption with the wrong key`() {
118     val dataLen = 1500
119     val origData = ByteArray(dataLen)
120     rnd.nextBytes(origData)
121     val encryptor = Encryptor(key1.toAesKey())
122
123     val streamForEncrypt = ByteArrayOutputStream(dataLen*3)
124     val encryptedStream = encryptor.encryptStream(streamForEncrypt)
125     encryptedStream.write(origData)
126     encryptedStream.close()
127     val encryptedData = streamForEncrypt.toByteArray()
128
129     val newEncryptor = Encryptor(key2.toAesKey())
130     try {
131         val streamForDecrypt = ByteArrayInputStream(encryptedData)
132         val decryptedStream = newEncryptor.decryptStream(streamForDecrypt)

```

```
133         decryptedStream.readAllBytes()
134         fail("There is not exception on decrypt with wrong key")
135     }
136     catch (e: Exception) {
137         // good
138     }
139 }
140 }
```

Исходный файл domain/src/test/java/com/github/nullptroma/wallenc/domain/
errors/WallencExceptionMappingTest.kt

```
1 package com.github.nullptroma.wallenc.domain.errors
2
3 import org.junit.Assert.assertEquals
4 import org.junit.Assert.assertTrue
5 import org.junit.Test
6 import java.io.FileNotFoundException
7 import java.io.IOException
8
9 class WallencExceptionMappingTest {
10
11     @Test
12     fun preservesWallencException() {
13         val original = WallencException.Feature.StorageNotFound()
14         assertEquals(original, original.toWallencException())
15     }
16
17     @Test
18     fun mapsFileNotFoundException() {
19         val mapped = FileNotFoundException("missing").toWallencException()
20         assertTrue(mapped is WallencException.Storage.FileNotFoundException)
21     }
22
23     @Test
24     fun mapsIOExceptionToIoFailed() {
25         val cause = IOException("disk")
26         val mapped = cause.toWallencException()
27         assertTrue(mapped is WallencException.Network.IoFailed)
28         assertEquals(cause, (mapped as WallencException.Network.IoFailed).cause)
29     }
30
31     @Test
32     fun mapsGenericExceptionToUnknown() {
33         val cause = Exception("boom")
34         val mapped = cause.toWallencException()
35         assertTrue(mapped is WallencException.Unknown)
36         assertEquals(cause, (mapped as WallencException.Unknown).cause)
37     }
38 }
39
```

ПРИЛОЖЕНИЕ А.4

Модуль :usecases

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/
FindStorageUseCase.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
4 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
5 import java.util.UUID
6 import javax.inject.Inject
7 import javax.inject.Singleton
8
9 @Singleton
10 class FindStorageUseCase @Inject constructor(
11     private val vaultsManager: IVaultsManager,
12 ) {
13     fun find(storageUuid: UUID): IStorage? {
14         return (
15             vaultsManager.allStorages.value +
16             vaultsManager.unlockManager.openedStorages.value.values
17         )
18             .distinctBy { it.uuid }
19             .firstOrNull { it.uuid == storageUuid }
20     }
21 }
22
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/
GetOpenedStoragesUseCase.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
4  import com.github.nullptroma.wallenc.domain.interfaces.IUnlockManager
5  import kotlinx.coroutines.flow.StateFlow
6  import java.util.UUID
7  import javax.inject.Inject
8  import javax.inject.Singleton
9
10 @Singleton
11 class GetOpenedStoragesUseCase @Inject constructor(
12     private val unlockManager: IUnlockManager,
13 ) {
14     val openedStorages: StateFlow<Map<UUID, IStorageInfo>>
15         get() = unlockManager.openedStorages
16 }
17
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/
ManageStorageSyncGroupsUseCase.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncGroupStore
4 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroup
5 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroupEncryptionKind
6 import java.util.UUID
7 import javax.inject.Inject
8 import javax.inject.Singleton
9
10 data class StorageSyncCompatibilityInput(
11     val isEncrypted: Boolean,
12     val encryptionSecret: String? = null,
13 )
14
15 sealed interface AddStorageToSyncGroupResult {
16     data object Added : AddStorageToSyncGroupResult
17     data object GroupNotFound : AddStorageToSyncGroupResult
18     data object AlreadyInGroup : AddStorageToSyncGroupResult
19     data object EncryptedStorageNotAllowed : AddStorageToSyncGroupResult
20     data object IncompatibleEncryption : AddStorageToSyncGroupResult
21     data object MissingEncryptionSecret : AddStorageToSyncGroupResult
22 }
23
24 @Singleton
25 class ManageStorageSyncGroupsUseCase @Inject constructor(
26     private val groupStore: IStorageSyncGroupStore,
27 ) {
28     suspend fun getGroups(): List<StorageSyncGroup> = groupStore.getGroups()
29
30     suspend fun createGroup(): StorageSyncGroup {
31         val existingIds = getGroups().map { it.id }.toSet()
32         var index = 1
33         var candidate = "group-$index"
34         while (candidate in existingIds) {
35             index++
36             candidate = "group-$index"
37         }
38         val group = StorageSyncGroup(
39             id = candidate,
40             storageUuids = emptySet(),
41             encryptionKind = StorageSyncGroupEncryptionKind.UNSET,
42             encryptionSecret = null,
```

```

43         )
44         groupStore.putGroup(group)
45         return group
46     }
47
48     suspend fun removeGroup(groupId: String) {
49         groupStore.removeGroup(groupId.trim())
50     }
51
52     suspend fun addStorageToGroup(
53         groupId: String,
54         storageUuid: UUID,
55         compatibility: StorageSyncCompatibilityInput,
56     ): AddStorageToSyncGroupResult {
57         val current = getGroups().firstOrNull { it.id == groupId }
58             ?: return AddStorageToSyncGroupResult.GroupNotFound
59
60         if (storageUuid in current.storageUuids) {
61             return AddStorageToSyncGroupResult.AlreadyInGroup
62         }
63         if (compatibility.isEncrypted) {
64             return AddStorageToSyncGroupResult.EncryptedStorageNotAllowed
65         }
66
67         val effectiveEncryption = StorageSyncGroupEncryptionKind.NONE to null
68
69         val (nextKind, nextSecret) = when (current.encryptionKind) {
70             StorageSyncGroupEncryptionKind.UNSET -> effectiveEncryption
71             StorageSyncGroupEncryptionKind.NONE -> {
72                 if (effectiveEncryption.first != StorageSyncGroupEncryptionKind.NONE) {
73                     return AddStorageToSyncGroupResult.IncompatibleEncryption
74                 }
75                 StorageSyncGroupEncryptionKind.NONE to null
76             }
77
78             StorageSyncGroupEncryptionKind.PASSWORD -> {
79                 return AddStorageToSyncGroupResult.IncompatibleEncryption
80             }
81         }
82
83         groupStore.putGroup(
84             current.copy(
85                 storageUuids = current.storageUuids + storageUuid,
86                 encryptionKind = nextKind,
87                 encryptionSecret = nextSecret,

```



```

88         ),
89     )
90     return AddStorageToSyncGroupResult.Added
91 }
92
93 suspend fun removeStorageFromGroup(groupId: String, storageUuid: UUID) {
94     val current = getGroups().firstOrNull { it.id == groupId } ?: return
95     val remaining = current.storageUuids - storageUuid
96     groupStore.putGroup(
97         current.copy(
98             storageUuids = remaining,
99             encryptionKind = if (remaining.isEmpty()) {
100                 StorageSyncGroupEncryptionKind.UNSET
101             } else {
102                 current.encryptionKind
103             },
104             encryptionSecret = if (remaining.isEmpty()) null else
current.encryptionSecret,
105         ),
106     )
107 }
108 }
109

```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ ManageStoragesEncryptionUseCase.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
4  import com.github.nullptroma.wallenc.domain.encrypt.Encryptor
5  import com.github.nullptroma.wallenc.domain.interfaces.IStorage
6  import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
7  import com.github.nullptroma.wallenc.domain.interfaces.IUnlockManager
8  import com.github.nullptroma.wallenc.domain.tasks.TaskProgress
9  import kotlinx.coroutines.flow.first
10 import javax.inject.Inject
11 import javax.inject.Singleton
12
13 @Singleton
14 class ManageStoragesEncryptionUseCase @Inject constructor(
15     private val unlockManager: IUnlockManager,
16 ) {
17     sealed interface CanEncryptResult {
18         data object Allowed : CanEncryptResult
19         data object UnsupportedStorageType : CanEncryptResult
20         data object AlreadyEncrypted : CanEncryptResult
21         data object StorageIsNotEmpty : CanEncryptResult
22         data object StorageStateUnknown : CanEncryptResult
23     }
24
25     suspend fun canEncrypt(storage: IStorageInfo): CanEncryptResult {
26         if (storage !is IStorage) return CanEncryptResult.UnsupportedStorageType
27         if (storage.metaInfo.value.encInfo != null) return CanEncryptResult.AlreadyEncrypted
28
29         val isEmpty = storage.isEmpty.first()
30         return when (isEmpty) {
31             true -> CanEncryptResult.Allowed
32             false -> CanEncryptResult.StorageIsNotEmpty
33             null -> CanEncryptResult.StorageStateUnknown
34         }
35     }
36
37     suspend fun enableEncryption(storage: IStorageInfo, key: EncryptKey, encryptPath:
38 Boolean) {
39         when (val result = canEncrypt(storage)) {
40             CanEncryptResult.Allowed -> (storage as IStorage).setEncInfo(
41                 Encryptor.generateEncryptionInfo(key, encryptPath)
42             )
43         }
44     }
45 }
```

```

42         CanEncryptResult.AlreadyEncrypted -> throw IllegalStateException("Storage is
already encrypted")
43         CanEncryptResult.StorageIsNotEmpty -> throw IllegalStateException("Storage is
not empty")
44         CanEncryptResult.StorageStateUnknown -> throw IllegalStateException("Storage
state is unknown")
45         CanEncryptResult.UnsupportedStorageType -> throw
IllegalStateException("Unsupported storage type")
46     }
47 }
48
49 suspend fun rememberStorageKey(storage: IStorageInfo, key: EncryptKey) {
50     if (storage is IStorage) {
51         unlockManager.rememberKey(storage, key)
52         return
53     }
54     throw IllegalStateException("Unsupported storage type")
55 }
56
57 suspend fun openStorage(storage: IStorageInfo, key: EncryptKey, rememberPassword:
Boolean): IStorageInfo {
58     if (storage is IStorage) return unlockManager.open(storage, key, rememberPassword)
59     throw IllegalStateException("Unsupported storage type")
60 }
61
62 suspend fun closeStorage(storage: IStorageInfo) {
63     if (storage is IStorage) {
64         unlockManager.close(storage)
65     }
66 }
67
68 suspend fun clearAndDisableEncryption(
69     storage: IStorageInfo,
70     onClearProgress: suspend (TaskProgress) -> Unit = {},
71 ) {
72     if (storage !is IStorage) return
73     storage.clearAllContent(onClearProgress)
74     storage.setEncInfo(null)
75     unlockManager.close(storage)
76 }
77
78 }
79

```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ ManageTextSecretsUseCase.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  import com.github.nullptroma.wallenc.domain.datatypes.TextSecretEntryRecord
4  import com.github.nullptroma.wallenc.domain.datatypes.TextSecretRecord
5  import com.github.nullptroma.wallenc.domain.interfaces.IStorage
6  import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
7  import kotlinx.coroutines.flow.Flow
8  import kotlinx.coroutines.flow.distinctUntilChanged
9  import kotlinx.coroutines.flow.filter
10 import kotlinx.coroutines.flow.flowOf
11 import kotlinx.coroutines.flow.map
12 import kotlinx.coroutines.flow.merge
13 import kotlinx.coroutines.sync.Mutex
14 import kotlinx.coroutines.sync.withLock
15 import kotlinx.serialization.json.JsonArray
16 import kotlinx.serialization.json.JsonElement
17 import kotlinx.serialization.json.JsonObject
18 import kotlinx.serialization.json.JsonPrimitive
19 import kotlinx.serialization.json.buildJsonArray
20 import kotlinx.serialization.json.buildJsonObject
21 import kotlinx.serialization.json.contentOrNull
22 import kotlinx.serialization.json.jsonPrimitive
23 import java.util.UUID
24 import javax.inject.Inject
25 import javax.inject.Singleton
26
27 @Singleton
28 class ManageTextSecretsUseCase @Inject constructor() {
29     private val mutex = Mutex()
30
31     fun observe(storageInfo: IStorageInfo): Flow<List<TextSecretRecord>> {
32         val storage = storageInfo as? IStorage ?: return flowOf(emptyList())
33         return merge(
34             flowOf(Unit),
35             storage.accessor.filesUpdates
36                 .filter { page ->
37                     page.data.any { file ->
38                         domainFilePathEquals(file.metaInfo.path)
39                     }
40                 }
41                 .map {},
42         ).map {
```

```

43         mutex.withLock { readAll(storage) }
44     }.distinctUntilChanged()
45 }
46
47 suspend fun get(storageInfo: IStorageInfo, id: String): TextSecretRecord? =
mutex.withLock {
48     val storage = storageInfo as? IStorage ?: return@withLock null
49     readAll(storage).firstOrNull { it.id == id }
50 }
51
52 suspend fun create(
53     storageInfo: IStorageInfo,
54     title: String,
55     items: List<TextSecretEntryRecord>,
56     id: String = UUID.randomUUID().toString(),
57 ): TextSecretRecord =
58     mutex.withLock {
59         val storage = storageInfo as? IStorage ?: error("Storage is not writable")
60         val next = TextSecretRecord(
61             id = id,
62             title = title.trim(),
63             items = items.normalizeItems(),
64         )
65         val updated = readAll(storage).toMutableList().apply { add(next) }
66         writeAll(storage, updated)
67         next
68     }
69
70 suspend fun update(storageInfo: IStorageInfo, secret: TextSecretRecord): Boolean =
mutex.withLock {
71     val storage = storageInfo as? IStorage ?: return@withLock false
72     val current = readAll(storage)
73     val index = current.indexOfFirst { it.id == secret.id }
74     if (index < 0) return@withLock false
75     val updatedSecret = secret.copy(
76         title = secret.title.trim(),
77         items = secret.items.normalizeItems(),
78     )
79     val updated = current.toMutableList().apply { this[index] = updatedSecret }
80     writeAll(storage, updated)
81     true
82 }
83
84 suspend fun delete(storageInfo: IStorageInfo, id: String): Boolean = mutex.withLock {
85     val storage = storageInfo as? IStorage ?: return@withLock false

```

```

86         val current = readAll(storage)
87         val updated = current.filterNot { it.id == id }
88         if (updated.size == current.size) return@withLock false
89         writeAll(storage, updated)
90         true
91     }
92
93     private suspend fun readAll(storage: IStorage): List<TextSecretRecord> {
94         return StorageDomainJsonIo.readArray(storage,
95             StorageDomainDataFiles.TEXT_SECRETS_FILE)
96             .mapNotNull { parseSecret(it) }
97     }
98
99     private suspend fun writeAll(storage: IStorage, records: List<TextSecretRecord>) {
100         StorageDomainJsonIo.writeArray(
101             storage = storage,
102             fileName = StorageDomainDataFiles.TEXT_SECRETS_FILE,
103             data = records.map { encodeSecret(it) },
104         )
105     }
106
107     private fun parseSecret(element: JsonElement): TextSecretRecord? {
108         val obj = element as? JsonObject ?: return null
109         val id = obj["id"]?.jsonPrimitive?.contentOrNull?.takeIf { it.isNotBlank() } ?:
110         return null
111         val title = obj["title"]?.jsonPrimitive?.contentOrNull ?: return null
112         val itemsElement = obj["items"] ?: JsonArray(emptyList())
113         val items = (itemsElement as? JsonArray)?.mapNotNull { parseItem(it) } ?:
114         emptyList()
115         return TextSecretRecord(
116             id = id,
117             title = title,
118             items = items,
119         )
120     }
121
122     private fun parseItem(element: JsonElement): TextSecretEntryRecord? {
123         val obj = element as? JsonObject ?: return null
124         val value = obj["value"]?.jsonPrimitive?.contentOrNull ?: return null
125         val label = obj["label"]?.jsonPrimitive?.contentOrNull
126         return TextSecretEntryRecord(
127             label = label,
128             value = value,
129         )
130     }
131 }

```

```

129 private fun encodeSecret(record: TextSecretRecord): JsonElement = buildJsonObject {
130     put("id", JsonPrimitive(record.id))
131     put("title", JsonPrimitive(record.title))
132     put(
133         "items",
134         buildJsonArray {
135             record.items.forEach { item ->
136                 add(
137                     buildJsonObject {
138                         item.label?.let { put("label", JsonPrimitive(it)) }
139                         put("value", JsonPrimitive(item.value))
140                     },
141                 )
142             }
143         },
144     )
145 }
146
147 private fun List<TextSecretEntryRecord>.normalizeItems(): List<TextSecretEntryRecord> =
148     this.mapNotNull { item ->
149         val value = item.value.trim()
150         if (value.isBlank()) {
151             null
152         } else {
153             TextSecretEntryRecord(
154                 label = item.label?.trim().takeUnless { it.isNullOrBlank() },
155                 value = value,
156             )
157         }
158     }
159
160 private fun domainFilePathEquals(path: String): Boolean =
161     path.trimStart('/') == StorageDomainDataFiles.TEXT_SECRETS_FILE.trimStart('/')
162 }
163

```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ ManageTwoFaTokensUseCase.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  import com.github.nullptroma.wallenc.domain.datatypes.TwoFaTokenRecord
4  import com.github.nullptroma.wallenc.domain.interfaces.IStorage
5  import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
6  import kotlinx.coroutines.flow.Flow
7  import kotlinx.coroutines.flow.distinctUntilChanged
8  import kotlinx.coroutines.flow.filter
9  import kotlinx.coroutines.flow.flowOf
10 import kotlinx.coroutines.flow.map
11 import kotlinx.coroutines.flow.merge
12 import kotlinx.coroutines.sync.Mutex
13 import kotlinx.coroutines.sync.withLock
14 import kotlinx.serialization.json.JsonElement
15 import kotlinx.serialization.json.JsonObject
16 import kotlinx.serialization.json.JsonPrimitive
17 import kotlinx.serialization.json.buildJsonObject
18 import kotlinx.serialization.json.contentOrNull
19 import kotlinx.serialization.json.jsonPrimitive
20 import java.util.UUID
21 import javax.inject.Inject
22 import javax.inject.Singleton
23
24 @Singleton
25 class ManageTwoFaTokensUseCase @Inject constructor() {
26     private val mutex = Mutex()
27
28     fun observe(storageInfo: IStorageInfo): Flow<List<TwoFaTokenRecord>> {
29         val storage = storageInfo as? IStorage ?: return flowOf(emptyList())
30         return merge(
31             flowOf(Unit),
32             storage.accessor.filesUpdates
33                 .filter { page ->
34                     page.data.any { file ->
35                         domainFilePathEquals(file.metaInfo.path)
36                     }
37                 }
38                 .map {},
39             ).map {
40                 mutex.withLock { readAll(storage) }
41             }.distinctUntilChanged()
42     }
```



```

43
44     suspend fun get(storageInfo: IStorageInfo, id: String): TwoFaTokenRecord? =
mutex.withLock {
45         val storage = storageInfo as? IStorage ?: return@withLock null
46         readAll(storage).firstOrNull { it.id == id }
47     }
48
49     suspend fun create(
50         storageInfo: IStorageInfo,
51         issuer: String,
52         account: String,
53         secret: String,
54         notes: String? = null,
55         digits: Int = 6,
56         periodSeconds: Int = 30,
57         algorithm: String = "SHA1",
58     ): TwoFaTokenRecord = mutex.withLock {
59         val storage = storageInfo as? IStorage ?: error("Storage is not writable")
60         val next = TwoFaTokenRecord(
61             id = UUID.randomUUID().toString(),
62             issuer = issuer.trim(),
63             account = account.trim(),
64             secret = secret.trim(),
65             notes = notes?.trim().takeUnless { it.isNullOrBlank() },
66             digits = digits.coerceIn(6, 8),
67             periodSeconds = periodSeconds.coerceAtLeast(1),
68             algorithm = normalizeAlgorithm(algorithm),
69         )
70         val updated = readAll(storage).toMutableList().apply { add(next) }
71         writeAll(storage, updated)
72         next
73     }
74
75     suspend fun update(storageInfo: IStorageInfo, token: TwoFaTokenRecord): Boolean =
mutex.withLock {
76         val storage = storageInfo as? IStorage ?: return@withLock false
77         val updatedToken = token.copy(
78             issuer = token.issuer.trim(),
79             account = token.account.trim(),
80             secret = token.secret.trim(),
81             notes = token.notes?.trim().takeUnless { it.isNullOrBlank() },
82         )
83         val current = readAll(storage)
84         val index = current.indexOfFirst { it.id == token.id }
85         if (index < 0) return@withLock false

```

```

86         val updated = current.toMutableList().apply { this[index] = updatedToken }
87         writeAll(storage, updated)
88         true
89     }
90
91     suspend fun delete(storageInfo: IStorageInfo, id: String): Boolean = mutex.withLock {
92         val storage = storageInfo as? IStorage ?: return@withLock false
93         val current = readAll(storage)
94         val updated = current.filterNot { it.id == id }
95         if (updated.size == current.size) return@withLock false
96         writeAll(storage, updated)
97         true
98     }
99
100     private suspend fun readAll(storage: IStorage): List<TwoFaTokenRecord> {
101         return StorageDomainJsonIo.readArray(storage,
102             StorageDomainDataFiles.TWO_FA_TOKENS_FILE)
103             .mapNotNull { parseToken(it) }
104     }
105
106     private suspend fun writeAll(storage: IStorage, records: List<TwoFaTokenRecord>) {
107         StorageDomainJsonIo.writeArray(
108             storage = storage,
109             fileName = StorageDomainDataFiles.TWO_FA_TOKENS_FILE,
110             data = records.map { record -> encodeToken(record) },
111         )
112     }
113
114     private fun parseToken(element: JsonElement): TwoFaTokenRecord? {
115         val obj = element as? JsonObject ?: return null
116         val id = obj["id"]?.jsonPrimitive?.contentOrNull?.takeIf { it.isNotBlank() } ?:
117         return null
118         val issuer = obj["issuer"]?.jsonPrimitive?.contentOrNull ?: return null
119         val account = obj["account"]?.jsonPrimitive?.contentOrNull ?: return null
120         val secret = obj["secret"]?.jsonPrimitive?.contentOrNull ?: return null
121         val notes = obj["notes"]?.jsonPrimitive?.contentOrNull
122         val digits = obj["digits"]?.jsonPrimitive?.contentOrNull?.toIntOrNull()?.coerceIn(6,
123             8) ?: 6
124         val periodSeconds =
125         obj["periodSeconds"]?.jsonPrimitive?.contentOrNull?.toIntOrNull()?.coerceAtLeast(1) ?: 30
126         val algorithm = normalizeAlgorithm(obj["algorithm"]?.jsonPrimitive?.contentOrNull ?:
127             "SHA1")
128         return TwoFaTokenRecord(
129             id = id,
130             issuer = issuer,
131             account = account,
132             secret = secret,

```

```

128         notes = notes,
129         digits = digits,
130         periodSeconds = periodSeconds,
131         algorithm = algorithm,
132     )
133 }
134
135 private fun encodeToken(record: TwoFaTokenRecord): JsonElement = buildJsonObject {
136     put("id", JsonPrimitive(record.id))
137     put("issuer", JsonPrimitive(record.issuer))
138     put("account", JsonPrimitive(record.account))
139     put("secret", JsonPrimitive(record.secret))
140     record.notes?.let { put("notes", JsonPrimitive(it)) }
141     put("digits", JsonPrimitive(record.digits))
142     put("periodSeconds", JsonPrimitive(record.periodSeconds))
143     put("algorithm", JsonPrimitive(normalizeAlgorithm(record.algorithm)))
144 }
145
146 private fun normalizeAlgorithm(algorithm: String): String {
147     val normalized = algorithm
148         .trim()
149         .uppercase()
150         .replace("HMAC", "")
151         .replace("-", "")
152         .replace("_", "")
153     return when (normalized) {
154         "SHA1", "SHA256", "SHA512" -> normalized
155         else -> "SHA1"
156     }
157 }
158
159 private fun domainFilePathEquals(path: String): Boolean =
160     path.trimStart('/') == StorageDomainDataFiles.TWO_FA_TOKENS_FILE.trimStart('/')
161 }
162

```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/
ManageVaultUseCase.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
4 import com.github.nullptroma.wallenc.domain.interfaces.IVault
5 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
6 import kotlinx.coroutines.ExperimentalCoroutinesApi
7 import kotlinx.coroutines.flow.Flow
8 import kotlinx.coroutines.flow.flatMapLatest
9 import kotlinx.coroutines.flow.flowOf
10 import kotlinx.coroutines.flow.map
11 import java.util.UUID
12 import javax.inject.Inject
13 import javax.inject.Singleton
14
15 @OptIn(ExperimentalCoroutinesApi::class)
16 @Singleton
17 class ManageVaultUseCase @Inject constructor(
18     private val manager: IVaultsManager,
19 ) {
20
21     /** Найти vault по идентификатору в текущем состоянии. */
22     fun find(vaultUuid: UUID): IVault? =
23         manager.vaults.value.firstOrNull { it.uuid == vaultUuid }
24
25     /** Реактивно отдаёт сам vault, либо null если он отсутствует. */
26     fun observe(vaultUuid: UUID): Flow<IVault?> =
27         manager.vaults.map { list -> list.firstOrNull { it.uuid == vaultUuid } }
28
29     /** Реактивно отдаёт storages указанного vault'a; пустой список, если vault не найден. */
30     fun storagesOf(vaultUuid: UUID): Flow<List<IStorage>> =
31         observe(vaultUuid).flatMapLatest { vault -> vault?.storages ?: flowOf(emptyList()) }
32
33     /** Идёт листинг/пересканирование storages vault'a. */
34     fun storagesScanInProgressOf(vaultUuid: UUID): Flow<Boolean> =
35         observe(vaultUuid).flatMapLatest { vault -> vault?.storagesScanInProgress ?: flowOf(false) }
36
37     /** Создать новое хранилище в указанном vault'e. */
38     suspend fun createStorage(vaultUuid: UUID): IStorage {
39         val vault = find(vaultUuid)
40         ?: throw IllegalStateException("Vault $vaultUuid is not registered")
41         return vault.createStorage()
```

```
42     }
43
44     /** Пересканировать storages vault'a (листинг на Диске и повторный init). */
45     suspend fun rescanStorages(vaultUuid: UUID) {
46         val vault = find(vaultUuid)
47         ?: throw IllegalStateException("Vault $vaultUuid is not registered")
48         vault.rescanStorages()
49     }
50 }
51
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ RemoveStorageUseCase.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  import com.github.nullptroma.wallenc.domain.interfaces.IStorage
4  import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
5  import com.github.nullptroma.wallenc.domain.interfaces.IUnlockManager
6  import com.github.nullptroma.wallenc.domain.interfaces.IVault
7  import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
8  import java.util.UUID
9  import javax.inject.Inject
10 import javax.inject.Singleton
11
12 @Singleton
13 class RemoveStorageUseCase @Inject constructor(
14     private val vaultsManager: IVaultsManager,
15     private val unlockManager: IUnlockManager,
16     private val manageStoragesEncryptionUseCase: ManageStoragesEncryptionUseCase,
17 ) {
18
19     suspend fun remove(storage: IStorageInfo) {
20         if (storage !is IStorage) return
21
22         if (!storage.isVirtualStorage) {
23             unlockManager.close(storage)
24             findOwningVault(storage.uuid)?.remove(storage)
25             return
26         }
27
28         val parent = findParentStorage(storage) ?: return
29         manageStoragesEncryptionUseCase.clearAndDisableEncryption(parent)
30     }
31
32     private fun findOwningVault(storageUuid: UUID): IVault? =
33         vaultsManager.vaults.value.firstOrNull { v ->
34             v.storages.value.any { it.uuid == storageUuid }
35         }
36
37     private fun findParentStorage(storage: IStorage): IStorage? {
38         val opened = unlockManager.openedStorages.value
39         val parentUuid = opened.entries.firstOrNull { it.value.uuid == storage.uuid }?.key
40         ?: return null
41         val realParent = vaultsManager.vaults.value
42             .flatMap { it.storages.value }
```

```
43         .firstOrNull { it.uuid == parentUuid }
44     return realParent ?: opened[parentUuid]
45 }
46 }
47
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/
RenameStorageUseCase.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  import com.github.nullptroma.wallenc.domain.interfaces.IStorage
4  import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
5  import javax.inject.Inject
6  import javax.inject.Singleton
7
8  @Singleton
9  class RenameStorageUseCase @Inject constructor() {
10      suspend fun rename(storage: IStorageInfo, newName: String) {
11          when (storage) {
12              is IStorage -> storage.rename(newName)
13          }
14      }
15  }
16
```


Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ RunStorageSyncUseCase.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.domain.errors.toWallencException
5 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncEngine
6 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
7 import com.github.nullptroma.wallenc.domain.tasks.IStorageSyncTaskTitleFormatter
8 import com.github.nullptroma.wallenc.domain.tasks.StorageSyncTriggerReason
9 import com.github.nullptroma.wallenc.domain.tasks.TaskId
10 import com.github.nullptroma.wallenc.domain.tasks.TaskLogKey
11 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLevel
12 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
13 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
14 import kotlinx.coroutines.Dispatchers
15 import kotlinx.coroutines.TimeoutCancellationException
16 import kotlinx.coroutines.flow.MutableStateFlow
17 import kotlinx.coroutines.flow.StateFlow
18 import kotlinx.coroutines.flow.asStateFlow
19 import kotlinx.coroutines.flow.filter
20 import kotlinx.coroutines.flow.first
21 import kotlinx.coroutines.withTimeout
22 import java.util.concurrent.atomic.AtomicBoolean
23 import javax.inject.Inject
24 import javax.inject.Singleton
25
26 @Singleton
27 class RunStorageSyncUseCase @Inject constructor(
28     private val orchestrator: ITaskOrchestrator,
29     private val syncEngine: IStorageSyncEngine,
30     private val syncReadiness: StorageSyncReadiness,
31     private val taskTitleFormatter: IStorageSyncTaskTitleFormatter,
32 ) {
33     private val running = AtomicBoolean(false)
34
35     private val _syncRunning = MutableStateFlow(false)
36     val syncRunning: StateFlow<Boolean> = _syncRunning.asStateFlow()
37
38     private val _activeSyncTaskId = MutableStateFlow<TaskId?>(null)
39     val activeSyncTaskId: StateFlow<TaskId?> = _activeSyncTaskId.asStateFlow()
40
41     /** Не реагировать на debounce до этого момента (мс с эпохи) после завершения sync. */
42     private val _debounceSuppressUntilMs = MutableStateFlow(0L)
```

```

43     val debounceSuppressUntilMs: StateFlow<Long> = _debounceSuppressUntilMs.asStateFlow()
44
45     /**
46      * @param reason источник запуска – заголовок задачи и лог пайплайна
47      * @return false, если синхронизация уже в очереди или выполняется – новая задача не
48      *      создана
49      */
50     fun enqueue(reason: StorageSyncTriggerReason): Boolean {
51         if (!running.compareAndSet(false, true)) {
52             return false
53         }
54         _syncRunning.value = true
55         try {
56             val taskId = orchestrator.enqueue(
57                 title = taskTitleFormatter.format(reason),
58                 dispatcher = Dispatchers.IO,
59                 work = { ctx ->
60                     try {
61                         executeSync(
62                             reason = reason,
63                             reportProgress = { fraction, label ->
64                                 ctx.ensureNotCancelled()
65                                 ctx.reportProgress(fraction, label)
66                             },
67                             log = { level, key -> ctx.log(level, key) },
68                         )
69                     } catch (e: Exception) {
70                         ctx.fail(e.toWallencException())
71                     } finally {
72                         clearRunningState()
73                     }
74                 },
75             )
76             _activeSyncTaskId.value = taskId
77             return true
78         } catch (t: Throwable) {
79             clearRunningState()
80             throw t
81         }
82     }
83
84     /**
85      * Ставит sync в пайплайн задач (как debounce / sync-tab) и ждёт завершения.
86      * Для WorkManager и других фоновых запусков без отдельного «orphan»-лога.
87      */

```

```

87     suspend fun enqueueAndAwait(reason: StorageSyncTriggerReason): StorageSyncRunOutcome {
88         if (!enqueue(reason)) {
89             return StorageSyncRunOutcome.SkippedAlreadyRunning
90         }
91         val taskId = _activeSyncTaskId.value
92         ?: return StorageSyncRunOutcome.Completed
93         return try {
94             withTimeout(SYNC_AWAIT_TIMEOUT_MS) {
95                 syncRunning.filter { !it }.first()
96             }
97             val state = orchestrator.pipelineState.value.tasks.find { it.id ==
taskId }?.state
98             when (state) {
99                 is TaskRunState.Failed -> StorageSyncRunOutcome.Failed(state.error)
100                 TaskRunState.Cancelled -> StorageSyncRunOutcome.Cancelled
101                 else -> StorageSyncRunOutcome.Completed
102             }
103         } catch (_: TimeoutCancellationException) {
104             orchestrator.cancel(taskId)
105             clearRunningState()
106             StorageSyncRunOutcome.Failed(
107                 WallencException.Unknown(
108                     cause = IllegalStateException("Storage sync await timed out after
109 ${SYNC_AWAIT_TIMEOUT_MS}ms"),
110                     ),
111                 )
112         }
113
114     private suspend fun executeSync(
115         reason: StorageSyncTriggerReason,
116         reportProgress: suspend (fraction: Float?, label: TaskProgressLabel?) -> Unit,
117         log: (TaskLogLevel, TaskLogKey) -> Unit,
118     ) {
119         try {
120             syncReadiness.awaitReady()
121             log(TaskLogLevel.Info, TaskLogKey.SyncStarted(reason))
122             reportProgress(null, TaskProgressLabel.SyncStarted)
123             try {
124                 syncEngine.syncAllGroups { fraction, label ->
125                     reportProgress(fraction, label)
126                 }
127             }
128             log(TaskLogLevel.Info, TaskLogKey.SyncFinished(reason))
129             reportProgress(null, TaskProgressLabel.SyncCompleted)
130         } catch (e: Exception) {

```

```

130         val err = e.toWallencException()
131         log(TaskLogLevel.Error, TaskLogKey.SyncFailed(err, reason))
132         throw e
133     }
134     } finally {
135         extendDebounceSuppress()
136     }
137 }
138
139 private fun extendDebounceSuppress() {
140     _debounceSuppressUntilMs.value = System.currentTimeMillis() +
DEBOUNCE_AFTER_CHANGE_MS
141 }
142
143 private fun clearRunningState() {
144     running.set(false)
145     _syncRunning.value = false
146     _activeSyncTaskId.value = null
147 }
148
149 companion object {
150     /** Пауза после последнего изменения перед debounce-sync; же окно подавления после
sync. */
151     const val DEBOUNCE_AFTER_CHANGE_MS = 60_000L
152
153     /** Максимальное ожидание фонового worker (WorkManager) – не блокировать periodic
work навсегда. */
154     private const val SYNC_AWAIT_TIMEOUT_MS = 25L * 60 * 1000
155 }
156 }
157

```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ StorageDomainDataFiles.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  /**
4   * Единая точка с путями обычных JSON-файлов пользовательских доменных данных.
5   */
6  object StorageDomainDataFiles {
7      const val TWO_FA_TOKENS_FILE = "/wallenc-data/two-fa-tokens.json"
8      const val TEXT_SECRETS_FILE = "/wallenc-data/text-secrets.json"
9  }
10
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ StorageDomainJsonIo.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  import com.github.nullptroma.wallenc.domain.interfaces.IStorage
4  import kotlinx.serialization.json.Json
5  import kotlinx.serialization.json.JsonArray
6  import kotlinx.serialization.json.JsonElement
7  import kotlinx.serialization.json.buildJsonArray
8
9  internal object StorageDomainJsonIo {
10     val json: Json = Json {
11         prettyPrint = true
12         ignoreUnknownKeys = true
13         explicitNulls = false
14     }
15
16     suspend fun readArray(storage: IStorage, fileName: String): JsonArray {
17         return try {
18             val text = storage.accessor.openRead(fileName).use { stream ->
19                 stream.readBytes().decodeToString()
20             }
21             if (text.isBlank()) {
22                 JsonArray(emptyList())
23             } else {
24                 when (val parsed = json.parseToJsonElement(text)) {
25                     is JsonArray -> parsed
26                     else -> JsonArray(emptyList())
27                 }
28             }
29         } catch (_: Exception) {
30             JsonArray(emptyList())
31         }
32     }
33
34     suspend fun writeArray(storage: IStorage, fileName: String, data: List<JsonElement>) {
35         val payload = buildJsonArray { data.forEach { add(it) } }
36         storage.accessor.openWrite(fileName).use { stream ->
37             stream.write(json.encodeToString(JsonArray.serializer(),
38                 payload).encodeToByteArray())
39         }
40     }
41 }
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/
StorageFileManagementUseCase.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IDirectory
4 import com.github.nullptroma.wallenc.domain.interfaces.IFile
5 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
6 import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
7 import javax.inject.Inject
8 import javax.inject.Singleton
9
10 @Singleton
11 class StorageFileManagementUseCase @Inject constructor() {
12     private var _storage: IStorage? = null
13
14     fun setStorage(storage: IStorageInfo) {
15         when (storage) {
16             is IStorage -> _storage = storage
17         }
18     }
19
20     suspend fun getAllFiles(): List<IFile> {
21         val storage = _storage ?: return listOf()
22         return storage.accessor.getAllFiles()
23     }
24
25     suspend fun getAllDirs(): List<IDirectory> {
26         val storage = _storage ?: return listOf()
27         return storage.accessor.getAllDirs()
28     }
29 }
30
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ StorageSyncEncryptionCompat.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
4 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroup
5 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroupEncryptionKind
6
7 /** Совместим, если у storage нет активного шифрования ([encInfo] == null). */
8 fun isStorageCompatibleWithGroup(
9     storage: IStorage,
10    group: StorageSyncGroup,
11 ): Boolean {
12     if (storage.metaInfo.value.encInfo != null) {
13         return false
14     }
15     return when (group.encryptionKind) {
16         StorageSyncGroupEncryptionKind.UNSET -> true
17         StorageSyncGroupEncryptionKind.NONE -> true
18         StorageSyncGroupEncryptionKind.PASSWORD -> false
19     }
20 }
21
```


Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ StorageSyncEngine.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournal
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalMerge
6 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation
7 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths
8 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
9 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
10 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncEngine
11 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncGroupStore
12 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
13 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
14 import kotlinx.coroutines.CancellationException
15 import kotlinx.coroutines.Dispatchers
16 import kotlinx.coroutines.async
17 import kotlinx.coroutines.awaitAll
18 import kotlinx.coroutines.coroutineScope
19 import kotlin.coroutines.coroutineContext
20 import kotlinx.coroutines.ensureActive
21 import kotlinx.coroutines.sync.Mutex
22 import kotlinx.coroutines.sync.withLock
23 import kotlinx.coroutines.withContext
24 import java.time.Instant
25 import java.util.UUID
26 import java.util.concurrent.ConcurrentHashMap
27 import java.util.concurrent.atomic.AtomicLong
28 import javax.inject.Inject
29 import javax.inject.Singleton
30
31 /**
32  * Синхронизация по журналам storage в группе.
33  * Блокировка на Yandex Disk – best-effort (см. [IStorageAccessor.tryAcquireSyncLock]);
34  * сериализация внутри процесса – [groupMutexes].
35  */
36 @Singleton
37 class StorageSyncEngine @Inject constructor(
38     private val vaultsManager: IVaultsManager,
39     private val groupStore: IStorageSyncGroupStore,
40     private val findStorageUseCase: FindStorageUseCase,
41 ) : IStorageSyncEngine {
42     private val holderId: String = UUID.randomUUID().toString()
```

```

43     private val groupMutexes = ConcurrentHashMap<String, Mutex>()
44     private val syncGeneration = AtomicLong(0)
45
46     override suspend fun syncAllGroups(
47         reportProgress: (suspend (fraction: Float?, label: TaskProgressLabel?) -> Unit)?,
48     ): Unit = withContext(Dispatchers.IO) {
49         val reporter = reportProgress ?: { _: Float?, _: TaskProgressLabel? -> }
50         val groups = groupStore.getGroups()
51         if (groups.isEmpty()) {
52             reporter(null, TaskProgressLabel.SyncNoGroups)
53             return@withContext
54         }
55         reporter(null, TaskProgressLabel.SyncPreparing(groups.size))
56         for (group in groups) {
57             coroutineContext.ensureActive()
58             syncGroupInternal(
59                 groupId = group.id,
60                 reportProgress = reporter,
61             )
62         }
63         reporter(null, TaskProgressLabel.SyncCompleted)
64     }
65
66     override suspend fun syncGroup(
67         groupId: String,
68         reportProgress: (suspend (fraction: Float?, label: TaskProgressLabel?) -> Unit)?,
69     ): Unit = withContext(Dispatchers.IO) {
70         val reporter = reportProgress ?: { _: Float?, _: TaskProgressLabel? -> }
71         syncGroupInternal(
72             groupId = groupId,
73             reportProgress = reporter,
74         )
75     }
76
77     private suspend fun syncGroupInternal(
78         groupId: String,
79         reportProgress: suspend (fraction: Float?, label: TaskProgressLabel?) -> Unit,
80     ) {
81         reportProgress(null, TaskProgressLabel.SyncGroupPreparing(groupId))
82         val mutex = groupMutexes.getOrPut(groupId) { Mutex() }
83         mutex.withLock {
84             val generationSnapshot = syncGeneration.incrementAndGet()
85             val group = groupStore.getGroups().firstOrNull { it.id == groupId }
86             if (group == null) {
87                 reportProgress(null, TaskProgressLabel.SyncGroupNotFound(groupId))

```

```

88         return
89     }
90     val storages = resolveStorages(group.storageUuids)
91     if (storages.size < 2) {
92         reportProgress(null,
TaskProgressLabel.SyncGroupSkippedTooFewStorages(groupId))
93         return
94     }
95     val incompatible = storages.filterNot { storage ->
96         isStorageCompatibleWithGroup(
97             storage = storage,
98             group = group,
99         )
100     }
101     if (incompatible.isNotEmpty()) {
102         reportProgress(
103             null,
104             TaskProgressLabel.SyncGroupSkippedIncompatibleEncryption(groupId,
incompatible.size),
105         )
106         return
107     }
108
109     var leaseUntil = Instant.now().plusSeconds(SYNC_LOCK_LEASE_SECONDS)
110     val lockedAccessors = mutableList0f<IStorageAccessor>()
111     try {
112         reportProgress(null, TaskProgressLabel.SyncGroupAcquiringLocks(groupId))
113         for ((lockIndex, storage) in storages.withIndex()) {
114             coroutineContext.ensureActive()
115             reportProgress(
116                 null,
117                 TaskProgressLabel.SyncGroupLockProgress(groupId, lockIndex + 1,
storages.size),
118             )
119             val locked = storage.accessor.tryAcquireSyncLock(holderId, leaseUntil)
120             if (!locked) {
121                 reportProgress(null, TaskProgressLabel.SyncGroupLockFailed(groupId))
122                 return
123             }
124             lockedAccessors.add(storage.accessor)
125         }
126
127         val mergedByPath = mutableMap0f<String, StorageSyncJournalEntry>()
128         val entriesByStorage = mutableMap0f<UUID, StorageSyncJournal>()
129

```

```

130         reportProgress(null, TaskProgressLabel.SyncGroupReadingJournals(groupId))
131         leaseUntil = renewLocksIfNeeded(
132             groupId = groupId,
133             lockedAccessors = lockedAccessors,
134             currentLeaseUntil = leaseUntil,
135             reportProgress = reportProgress,
136         ) ?: return
137         if (syncGeneration.get() != generationSnapshot) {
138             reportProgress(null, TaskProgressLabel.SyncGroupCancelled(groupId))
139             return
140         }
141         val journalReads = coroutineScope {
142             storages.mapIndexed { journalIndex, storage ->
143                 async {
144                     coroutineContext.ensureActive()
145                     reportProgress(
146                         null,
147                         TaskProgressLabel.SyncGroupJournalProgress(
148                             groupId,
149                             journalIndex + 1,
150                             storages.size,
151                         ),
152                     )
153                     storage.accessor.flushPendingSyncJournal()
154                     storage to
155                     filterSyncableJournal(storage.accessor.readSyncJournal())
156                 }.awaitAll()
157             }
158             for ((storage, journal) in journalReads) {
159                 entriesByStorage[storage.uuid] = journal
160                 mergedByPath.putAll(
161                     StorageSyncJournalMerge.merge(mergedByPath, journal),
162                 )
163             }
164
165             val mergedEntries = mergedByPath.entries.toList()
166             if (mergedEntries.isEmpty()) {
167                 reportProgress(null,
168                     TaskProgressLabel.SyncGroupNoJournalEntries(groupId))
169                 return
170             }
171
172             reportProgress(
173                 null,

```

```

173         TaskProgressLabel.SyncGroupProcessingEntries(groupId,
mergedEntries.size),
174     )
175     var applyFailures = 0
176     for ((pathIndex, merged) in mergedEntries.withIndex()) {
177         coroutineContext.ensureActive()
178         leaseUntil = renewLocksIfNeeded(
179             groupId = groupId,
180             lockedAccessors = lockedAccessors,
181             currentLeaseUntil = leaseUntil,
182             reportProgress = reportProgress,
183         ) ?: return
184         val path = merged.key
185         val winnerEntry = merged.value
186         reportProgress(
187             null,
188             TaskProgressLabel.SyncGroupEntryProgress(groupId, pathIndex + 1,
mergedEntries.size),
189         )
190         if (syncGeneration.get() != generationSnapshot) {
191             reportProgress(null, TaskProgressLabel.SyncGroupCancelled(groupId))
192             return
193         }
194         val sourceStorage = findSourceStorage(storages, entriesByStorage, path,
winnerEntry)
195         if (sourceStorage == null &&
196             winnerEntry.operation == StorageSyncOperation.UPSERT
197         ) {
198             continue
199         }
200
201         for (target in storages) {
202             coroutineContext.ensureActive()
203             if (target.uuid == sourceStorage?.uuid) {
204                 continue
205             }
206             val targetEntry = entriesByStorage[target.uuid]?.get(path)
207             if (targetEntry != null && compareEntries(targetEntry, winnerEntry)
>= 0) {
208                 continue
209             }
210             val applied = applyEntry(
211                 source = sourceStorage,
212                 target = target,
213                 entry = winnerEntry,
214             )

```

```

215             if (applied) {
216                 target.accessor.putSyncJournalEntries(mapOf(path to
winnerEntry))
217             } else {
218                 applyFailures++
219             }
220         }
221     }
222     if (applyFailures > 0) {
223         reportProgress(
224             null,
225             TaskProgressLabel.SyncGroupEntriesFailed(groupId, applyFailures),
226         )
227     }
228     reportProgress(null, TaskProgressLabel.SyncGroupCompleted(groupId))
229 } finally {
230     for (accessor in lockedAccessors) {
231         runCatching {
232             accessor.releaseSyncLock(holderId)
233         }
234     }
235 }
236 }
237 }
238
239 private fun filterSyncableJournal(journal: StorageSyncJournal): StorageSyncJournal {
240     return journal.filterKeys { StorageSyncPaths.isSyncableUserPath(it) }
241 }
242
243 private suspend fun renewLocksIfNeeded(
244     groupId: String,
245     lockedAccessors: List<IStorageAccessor>,
246     currentLeaseUntil: Instant,
247     reportProgress: suspend (fraction: Float?, label: TaskProgressLabel?) -> Unit,
248 ): Instant? {
249     val now = Instant.now()
250     if (currentLeaseUntil.isAfter(now.plusSeconds(SYNC_LOCK_RENEW_MARGIN_SECONDS))) {
251         return currentLeaseUntil
252     }
253     val nextLeaseUntil = now.plusSeconds(SYNC_LOCK_LEASE_SECONDS)
254     reportProgress(null, TaskProgressLabel.SyncGroupRenewingLocks(groupId))
255     for (accessor in lockedAccessors) {
256         coroutineContext.ensureActive()
257         val renewed = runCatching {
258             accessor.tryAcquireSyncLock(holderId, nextLeaseUntil)

```

```

259         }.getOrElse { false }
260         if (!renewed) {
261             reportProgress(null, TaskProgressLabel.SyncGroupLockRenewalFailed(groupId))
262             return null
263         }
264     }
265     return nextLeaseUntil
266 }
267
268 private fun resolveStorages(uuids: Set<UUID>): List<IStorage> {
269     return uuids.mapNotNull(findStorageUseCase::find)
270 }
271
272 private fun findSourceStorage(
273     storages: List<IStorage>,
274     entriesByStorage: Map<UUID, StorageSyncJournal>,
275     path: String,
276     winnerEntry: StorageSyncJournalEntry,
277 ): IStorage? {
278     if (winnerEntry.operation == StorageSyncOperation.DELETE ||
279         winnerEntry.operation == StorageSyncOperation.TRASH
280     ) {
281         return storages.firstOrNull { storage ->
282             entriesByStorage[storage.uuid]?.get(path) != null
283         } ?: storages.firstOrNull()
284     }
285     return storages.firstOrNull { storage ->
286         val entry = entriesByStorage[storage.uuid]?.get(path) ?: return@firstOrNull
287         false
288         compareEntries(entry, winnerEntry) == 0
289     }
290 }
291
292 private suspend fun applyEntry(
293     source: IStorage?,
294     target: IStorage,
295     entry: StorageSyncJournalEntry,
296 ): Boolean {
297     val result = when (entry.operation) {
298         StorageSyncOperation.DELETE -> {
299             runCatching {
300                 target.accessor.delete(entry.path, recordSyncJournal = false)
301             }
302         }

```

```

303         StorageSyncOperation.TRASH -> {
304             runCatching {
305                 target.accessor.moveToTrash(entry.path, recordSyncJournal = false)
306             }
307         }
308
309         StorageSyncOperation.UPSERT -> {
310             val sourceAccessor = source?.accessor ?: return false
311             runCatching {
312                 sourceAccessor.openRead(entry.path).use { input ->
313                     target.accessor.openWrite(entry.path, recordSyncJournal = false).use
314 { output ->
315                     input.copyTo(output)
316                 }
317             }
318         }
319     }
320     val error = result.exceptionOrNull() ?: return result.isSuccess
321     if (error is CancellationException) {
322         throw error
323     }
324     System.err.println(
325         "StorageSyncEngine: apply ${entry.operation} ${entry.path} " +
326         "target=${target.uuid}: ${error.message}",
327     )
328     return false
329 }
330
331 private fun compareEntries(a: StorageSyncJournalEntry, b: StorageSyncJournalEntry): Int
332 {
333     val seqCmp = a.revision.sequence.compareTo(b.revision.sequence)
334     if (seqCmp != 0) {
335         return seqCmp
336     }
337     val actorCmp = a.revision.actorId.compareTo(b.revision.actorId)
338     if (actorCmp != 0) {
339         return actorCmp
340     }
341     return a.revision.createdAt.compareTo(b.revision.createdAt)
342 }
343
344 private companion object {
345     private const val SYNC_LOCK_LEASE_SECONDS: Long = 30 * 60
346     private const val SYNC_LOCK_RENEW_MARGIN_SECONDS: Long = 60
347 }

```


347 }
348

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ StorageSyncReadiness.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncGroupStore
4 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
5 import kotlinx.coroutines.delay
6 import kotlin.coroutines.coroutineContext
7 import kotlinx.coroutines.ensureActive
8 import javax.inject.Inject
9 import javax.inject.Singleton
10
11 /**
12  * Ожидание готовности хранилищ перед синхронизацией (холодный старт из WorkManager).
13  */
14 @Singleton
15 class StorageSyncReadiness @Inject constructor(
16     private val vaultsManager: IVaultsManager,
17     private val groupStore: IStorageSyncGroupStore,
18     private val findStorageUseCase: FindStorageUseCase,
19 ) {
20     suspend fun awaitReady(
21         timeoutMs: Long = DEFAULT_TIMEOUT_MS,
22         pollIntervalMs: Long = POLL_INTERVAL_MS,
23     ) {
24         val groups = groupStore.getGroups()
25         if (groups.isEmpty()) {
26             return
27         }
28         val requiredUuids = groups.flatMap { it.storageUuids }.toSet()
29         val deadline = System.currentTimeMillis() + timeoutMs
30
31         while (System.currentTimeMillis() < deadline) {
32             coroutineContext.ensureActive()
33             if (!isAnyVaultScanning()) {
34                 break
35             }
36             delay(pollIntervalMs)
37         }
38
39         while (System.currentTimeMillis() < deadline) {
40             coroutineContext.ensureActive()
41             if (requiredUuids.all { uuid -> findStorageUseCase.find(uuid) != null }) {
42                 return
43             }
44         }
45     }
46 }
```

```
43         }
44         delay(pollIntervalMs)
45     }
46 }
47
48 private fun isAnyVaultScanning(): Boolean =
49     vaultsManager.vaults.value.any { it.storagesScanInProgress.value }
50
51 private companion object {
52     private const val DEFAULT_TIMEOUT_MS = 120_000L
53     private const val POLL_INTERVAL_MS = 250L
54 }
55 }
56
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/
StorageSyncRunOutcome.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4
5 /** Результат [RunStorageSyncUseCase.enqueueAndAwait]. */
6 sealed class StorageSyncRunOutcome {
7     /** Синхронизация уже выполнялась — новая задача не создана. */
8     data object SkippedAlreadyRunning : StorageSyncRunOutcome()
9
10    data object Completed : StorageSyncRunOutcome()
11
12    /** Задача синхронизации отменена пользователем или пайплайном. */
13    data object Cancelled : StorageSyncRunOutcome()
14
15    data class Failed(val error: WallencException) : StorageSyncRunOutcome()
16 }
17
```

Исходный файл usecases/src/main/java/com/github/nullptroma/wallenc/usecases/ TwoFaTotp.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.eatthepath.otp.TimeBasedOneTimePasswordGenerator
4 import com.github.nullptroma.wallenc.domain.datatypes.TwoFaTokenRecord
5 import java.time.Duration
6 import java.time.Instant
7 import javax.crypto.spec.SecretKeySpec
8
9 data class TwoFaCodeState(
10     val code: String,
11     val secondsUntilRefresh: Int,
12 )
13
14 /** Доля прошедшего TOTP-периода [0f, 1f] для плавного progress bar. */
15 fun totpPeriodProgress(nowMillis: Long, periodSeconds: Int): Float {
16     val period = periodSeconds.coerceAtLeast(1)
17     val elapsedSec = (nowMillis / 1000.0) % period
18     return (elapsedSec / period).toFloat().coerceIn(0f, 1f)
19 }
20
21 fun totpSecondsUntilRefresh(nowMillis: Long, periodSeconds: Int): Int {
22     val period = periodSeconds.coerceAtLeast(1)
23     return (period - ((nowMillis / 1000L) % period)).toInt().coerceAtLeast(0)
24 }
25
26 fun buildTwoFaCodeState(token: TwoFaTokenRecord, nowMillis: Long): TwoFaCodeState? {
27     val period = token.periodSeconds.coerceAtLeast(1)
28     val remaining = totpSecondsUntilRefresh(nowMillis, period)
29     val code = generateTotpCode(
30         secret = token.secret,
31         nowMillis = nowMillis,
32         digits = token.digits.coerceIn(6, 8),
33         periodSeconds = token.periodSeconds.coerceAtLeast(1),
34         algorithm = token.algorithm,
35     ) ?: return null
36     return TwoFaCodeState(
37         code = code,
38         secondsUntilRefresh = remaining,
39     )
40 }
41
42 private fun generateTotpCode(
```

```

43     secret: String,
44     nowMillis: Long,
45     digits: Int,
46     periodSeconds: Int,
47     algorithm: String,
48 ): String? {
49     val key = decodeBase32(secret) ?: return null
50     val macAlgorithm = when (algorithm.trim().uppercase().replace("-", "").replace("_", ""))
51 {
52         "SHA256" -> "HmacSHA256"
53         "SHA512" -> "HmacSHA512"
54         else -> "HmacSHA1"
55     }
56     return runCatching {
57         val generator = TimeBasedOneTimePasswordGenerator(
58             Duration.ofSeconds(periodSeconds.toLong()),
59             digits,
60             macAlgorithm,
61         )
62         val secretKey = SecretKeySpec(key, macAlgorithm)
63         val otp = generator.generateOneTimePassword(
64             secretKey,
65             Instant.ofEpochMilli(nowMillis),
66         )
67         otp.toString().padStart(digits, '0')
68     }.getOrNull()
69 }
70 private fun decodeBase32(input: String): ByteArray? {
71     val clean = input
72         .trim()
73         .replace(" ", "")
74         .replace("-", "")
75         .replace("=", "")
76         .uppercase()
77     if (clean.isEmpty()) return null
78
79     var buffer = 0
80     var bitsLeft = 0
81     val out = ArrayList<Byte>(clean.length * 5 / 8)
82     for (ch in clean) {
83         val value = BASE32_ALPHABET.index0f(ch)
84         if (value < 0) return null
85         buffer = (buffer shl 5) or value
86         bitsLeft += 5

```

```
87         if (bitsLeft >= 8) {
88             out.add(((buffer shr (bitsLeft - 8)) and 0xFF).toByte())
89             bitsLeft -= 8
90         }
91     }
92     return out.toByteArray()
93 }
94
95 private const val BASE32_ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ234567"
96
```

Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/
StorageDomainUseCasesTest.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretEntryRecord
4 import com.github.nullptroma.wallenc.usecases.fakes.FakeStorage
5 import kotlinx.coroutines.flow.first
6 import kotlinx.coroutines.test.runTest
7 import org.junit.Assert.assertEquals
8 import org.junit.Assert.assertNotNull
9 import org.junit.Assert.assertNull
10 import org.junit.Assert.assertTrue
11 import org.junit.Test
12
13 class StorageDomainUseCasesTest {
14
15     @Test
16     fun twoFaCrudWorksAndPersists() = runTest {
17         val storage = FakeStorage()
18         val useCase = ManageTwoFaTokensUseCase()
19
20         val created = useCase.create(
21             storageInfo = storage,
22             issuer = "GitHub",
23             account = "user@example.com",
24             secret = "SECRET",
25             notes = "primary",
26         )
27         assertNotNull(created.id)
28         assertEquals(1, useCase.observe(storage).first().size)
29
30         val updated = useCase.update(
31             storageInfo = storage,
32             token = created.copy(issuer = "GitHubUpdated"),
33         )
34         assertTrue(updated)
35         assertEquals("GitHubUpdated", useCase.get(storage, created.id)?.issuer)
36
37         val removed = useCase.delete(storage, created.id)
38         assertTrue(removed)
39         assertTrue(useCase.observe(storage).first().isEmpty())
40     }
41
42     @Test
```



```

43     fun twoFaInvalidJsonFallsBackToEmptyList() = runTest {
44         val storage = FakeStorage().apply {
45             setDomainFile(StorageDomainDataFiles.TWO_FA_TOKENS_FILE, "not-json")
46         }
47         val useCase = ManageTwoFaTokensUseCase()
48         assertTrue(useCase.observe(storage).first().isEmpty())
49     }
50
51     @Test
52     fun textSecretsCrudWorksOptionalLabels() = runTest {
53         val storage = FakeStorage()
54         val useCase = ManageTextSecretsUseCase()
55
56         val created = useCase.create(
57             storageInfo = storage,
58             title = "Server Credentials",
59             items = listOf(
60                 TextSecretEntryRecord(label = "username", value = "admin"),
61                 TextSecretEntryRecord(label = null, value = "password"),
62             ),
63         )
64         assertEquals(1, useCase.observe(storage).first().size)
65
66         val updated = useCase.update(
67             storageInfo = storage,
68             secret = created.copy(
69                 title = "Prod Credentials",
70                 items = created.items + TextSecretEntryRecord(label = "url", value =
71 "https://x.dev"),
72             ),
73         )
74         assertTrue(updated)
75         val loaded = useCase.get(storage, created.id)
76         assertEquals("Prod Credentials", loaded?.title)
77         assertEquals(3, loaded?.items?.size)
78
79         val deleted = useCase.delete(storage, created.id)
80         assertTrue(deleted)
81         assertNull(useCase.get(storage, created.id))
82     }
83
84     @Test
85     fun textSecretsInvalidJsonFallsBackToEmptyList() = runTest {
86         val storage = FakeStorage().apply {
87             setDomainFile(StorageDomainDataFiles.TEXT_SECRETS_FILE, "{broken}")

```

```
87         }
88         val useCase = ManageTextSecretsUseCase()
89         assertTrue(useCase.observe(storage).first().isEmpty())
90     }
91 }
92
```

Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/
StorageSyncEncryptionCompatTest.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
4 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroup
5 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroupEncryptionKind
6 import com.github.nullptroma.wallenc.usecases.fakes.FakeMetaInfo
7 import com.github.nullptroma.wallenc.usecases.fakes.FakeStorage
8 import org.junit.Assert.assertFalse
9 import org.junit.Assert.assertTrue
10 import org.junit.Test
11 import java.util.UUID
12
13 class StorageSyncEncryptionCompatTest {
14
15     @Test
16     fun storageWithoutEncInfoIsCompatible() {
17         val storage = FakeStorage(uuid = UUID.randomUUID(), meta = FakeMetaInfo(encInfo =
18 null))
19         val group = StorageSyncGroup(
20             id = "g1",
21             storageUuids = setOf(storage.uuid),
22             encryptionKind = StorageSyncGroupEncryptionKind.NONE,
23         )
24         assertTrue(isStorageCompatibleWithGroup(storage = storage, group = group))
25     }
26
27     @Test
28     fun storageWithEncInfoIsIncompatible() {
29         val storage = FakeStorage(
30             uuid = UUID.randomUUID(),
31             meta = FakeMetaInfo(
32                 encInfo = StorageEncryptionInfo(encryptedTestData = "x", pathIv =
33 ByteArray(16)),
34             ),
35         )
36         val group = StorageSyncGroup(
37             id = "g1",
38             storageUuids = setOf(storage.uuid),
39             encryptionKind = StorageSyncGroupEncryptionKind.NONE,
40         )
41         assertFalse(isStorageCompatibleWithGroup(storage = storage, group = group))
42     }
43 }
```


Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/ StorageSyncEngineTest.kt

```
1  package com.github.nullptroma.wallenc.usecases
2
3  import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
4  import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation
5  import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths
6  import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncRevision
7  import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroup
8  import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroupEncryptionKind
9  import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
10 import com.github.nullptroma.wallenc.usecases.fakes.FakeStorage
11 import com.github.nullptroma.wallenc.usecases.fakes.FakeStorageAccessor
12 import com.github.nullptroma.wallenc.usecases.fakes.FakeStorageSyncGroupStore
13 import com.github.nullptroma.wallenc.usecases.fakes.FakeVaultsManager
14 import kotlinx.coroutines.CancellationException
15 import kotlinx.coroutines.async
16 import kotlinx.coroutines.runBlocking
17 import org.junit.Assert.assertEquals
18 import org.junit.Assert.assertEquals
19 import org.junit.Assert.assertNull
20 import org.junit.Assert.assertTrue
21 import org.junit.Test
22 import java.time.Instant
23
24 class StorageSyncEngineTest {
25
26     private fun norm(path: String): String = StorageSyncPaths.normalize(path)
27
28     @Test
29     fun syncAllGroupsReportsNoGroupsWhenEmpty() = runBlocking {
30         val labels = mutableListOf<TaskProgressLabel?>()
31         val engine = createEngine(storages = emptyList(), groups = emptyList())
32         engine.syncAllGroups { _, label -> labels.add(label) }
33         assertTrue(labels.any { it is TaskProgressLabel.SyncNoGroups })
34     }
35
36     @Test
37     fun syncGroupCopiesFileFromSourceToTarget() = runBlocking {
38         val source = FakeStorage()
39         val target = FakeStorage()
40         val path = "shared/doc.txt"
41         val payload = "sync-payload".encodeToByteArray()
42         source.putFile(path, payload)
```

```

43
44     val entry = StorageSyncJournalEntry(
45         path = path,
46         operation = StorageSyncOperation.UPSERT,
47         revision = StorageSyncRevision(
48             sequence = 1L,
49             actorId = "actor-a",
50             createdAt = Instant.parse("2024-01-01T00:00:00Z"),
51         ),
52         size = payload.size.toLong(),
53         originStorageUuid = source.uuid,
54     )
55     source.addSyncJournalEntry(entry)
56
57     val groupId = "group-1"
58     val group = StorageSyncGroup(
59         id = groupId,
60         storageUuids = setOf(source.uuid, target.uuid),
61         encryptionKind = StorageSyncGroupEncryptionKind.NONE,
62     )
63     val labels = mutableList0f<TaskProgressLabel?>()
64     val engine = createEngine(
65         storages = list0f(source, target),
66         groups = list0f(group),
67     )
68     engine.syncGroup(groupId) { _, label -> labels.add(label) }
69
70     assertArrayEquals(payload, target.fileBytes(path))
71     assertTrue(labels.any { it is TaskProgressLabel.SyncGroupCompleted })
72     assertTrue(target.syncJournalEntries().any { it.path == norm(path) })
73 }
74
75 @Test
76 fun syncGroupSkippedWhenFewerThanTwoStorages() = runBlocking {
77     val only = FakeStorage()
78     val groupId = "solo"
79     val group = StorageSyncGroup(
80         id = groupId,
81         storageUuids = setOf(only.uuid),
82         encryptionKind = StorageSyncGroupEncryptionKind.NONE,
83     )
84     val labels = mutableList0f<TaskProgressLabel?>()
85     val engine = createEngine(storages = list0f(only), groups = list0f(group))
86     engine.syncGroup(groupId) { _, label -> labels.add(label) }
87     assertTrue(labels.any { it is TaskProgressLabel.SyncGroupSkippedTooFewStorages })

```

```

88     }
89
90     @Test
91     fun syncGroupDeleteRemovesFileOnTarget() = runBlocking {
92         val source = FakeStorage()
93         val target = FakeStorage()
94         val path = "obsolete.bin"
95         target.putFile(path, "old".encodeToByteArray())
96
97         val entry = StorageSyncJournalEntry(
98             path = path,
99             operation = StorageSyncOperation.DELETE,
100             revision = StorageSyncRevision(
101                 sequence = 2L,
102                 actorId = "actor-b",
103                 createdAt = Instant.parse("2024-06-01T00:00:00Z"),
104             ),
105         )
106         source.addSyncJournalEntry(entry)
107
108         val group = StorageSyncGroup(
109             id = "delete-group",
110             storageUuids = setOf(source.uuid, target.uuid),
111             encryptionKind = StorageSyncGroupEncryptionKind.NONE,
112         )
113         val engine = createEngine(
114             storages = listOf(source, target),
115             groups = listOf(group),
116         )
117         engine.syncGroup(group.id) { _, _ -> }
118
119         assertNull(target.fileBytes(path))
120         val targetEntry = target.syncJournalEntries().single { it.path == norm(path) }
121         assertEquals(2L, targetEntry.revision.sequence)
122         assertEquals("actor-b", targetEntry.revision.actorId)
123         assertEquals(StorageSyncOperation.DELETE, targetEntry.operation)
124     }
125
126     @Test
127     fun syncSkipsWhenTargetRevisionAlreadyWinner() = runBlocking {
128         val source = FakeStorage()
129         val target = FakeStorage()
130         val path = "already-synced.txt"
131         val payload = "same".encodeToByteArray()
132         source.putFile(path, payload)

```

```

133         target.putFile(path, payload)
134
135         val winner = StorageSyncJournalEntry(
136             path = path,
137             operation = StorageSyncOperation.UPSERT,
138             revision = StorageSyncRevision(
139                 sequence = 5L,
140                 actorId = "winner-actor",
141                 createdAt = Instant.parse("2024-08-01T00:00:00Z"),
142             ),
143             size = payload.size.toLong(),
144         )
145         source.addSyncJournalEntry(winner)
146         target.addSyncJournalEntry(winner)
147
148         val group = StorageSyncGroup(
149             id = "skip-group",
150             storageUuids = setOf(source.uuid, target.uuid),
151             encryptionKind = StorageSyncGroupEncryptionKind.NONE,
152         )
153         val engine = createEngine(
154             storages = listOf(source, target),
155             groups = listOf(group),
156         )
157         engine.syncGroup(group.id) { _, _ -> }
158
159         val targetJournal = target.syncJournalEntries().single { it.path == norm(path) }
160         assertEquals(winner.revision, targetJournal.revision)
161         assertEquals(winner.operation, targetJournal.operation)
162     }
163
164     @Test
165     fun openReadDoesNotChangeJournal() = runBlocking {
166         val storage = FakeStorage()
167         val path = "read-only.txt"
168         storage.putFile(path, "data".encodeToByteArray())
169         val before = storage.syncJournalEntries().size
170
171         storage.accessor.openRead(path).use { it.readBytes() }
172
173         assertEquals(before, storage.syncJournalEntries().size)
174     }
175
176     @Test
177     fun deleteWithRecordSyncJournalFalseDoesNotBumpSequence() = runBlocking {

```



```

178     val storage = FakeStorage()
179     val path = "to-delete.txt"
180     storage.putFile(path, "x".encodeToByteArray())
181     storage.addSyncJournalEntry(
182         StorageSyncJournalEntry(
183             path = path,
184             operation = StorageSyncOperation.UPSERT,
185             revision = StorageSyncRevision(10L, "prior", Instant.EPOCH),
186         ),
187     )
188
189     storage.accessor.delete(path, recordSyncJournal = false)
190
191     assertNull(storage.fileBytes(path))
192     val entry = storage.syncJournalEntries().single { it.path == norm(path) }
193     assertEquals(10L, entry.revision.sequence)
194     assertEquals(StorageSyncOperation.UPSERT, entry.operation)
195 }
196
197 @Test
198 fun syncGroupTrashSoftDeletesOnTarget() = runBlocking {
199     val source = FakeStorage()
200     val target = FakeStorage()
201     val path = "trashed/doc.txt"
202     val payload = "keep-in-trash".encodeToByteArray()
203     source.putFile(path, payload)
204     target.putFile(path, payload)
205
206     val entry = StorageSyncJournalEntry(
207         path = path,
208         operation = StorageSyncOperation.TRASH,
209         revision = StorageSyncRevision(
210             sequence = 3L,
211             actorId = "actor-trash",
212             createdAt = Instant.parse("2024-07-01T00:00:00Z"),
213         ),
214     )
215     source.addSyncJournalEntry(entry)
216
217     val group = StorageSyncGroup(
218         id = "trash-group",
219         storageUuids = setOf(source.uuid, target.uuid),
220         encryptionKind = StorageSyncGroupEncryptionKind.NONE,
221     )
222     val engine = createEngine(

```

```

223         storages = listOf(source, target),
224         groups = listOf(group),
225     )
226     engine.syncGroup(group.id) { _, _ -> }
227
228     assertEquals(payload, target.fileBytes(path))
229     assertTrue(norm(path) in (target.accessor as FakeStorageAccessor).trashedPaths)
230     val targetEntry = target.syncJournalEntries().single { it.path == norm(path) }
231     assertEquals(3L, targetEntry.revision.sequence)
232     assertEquals("actor-trash", targetEntry.revision.actorId)
233     assertEquals(StorageSyncOperation.TRASH, targetEntry.operation)
234 }
235
236 @Test
237 fun syncGroupStopsWhenLockCannotBeAcquired() = runBlocking {
238     val first = FakeStorage()
239     val second = FakeStorage().apply { setAcquireLockResult(false) }
240     val group = StorageSyncGroup(
241         id = "lock-fail",
242         storageUuids = setOf(first.uuid, second.uuid),
243         encryptionKind = StorageSyncGroupEncryptionKind.NONE,
244     )
245     first.addSyncJournalEntry(
246         StorageSyncJournalEntry(
247             path = "a.txt",
248             operation = StorageSyncOperation.UPSERT,
249             revision = StorageSyncRevision(1L, "x", Instant.EPOCH),
250         ),
251     )
252     val labels = mutableList<TaskProgressLabel?>()
253     val engine = createEngine(
254         storages = listOf(first, second),
255         groups = listOf(group),
256     )
257     engine.syncGroup(group.id) { _, label -> labels.add(label) }
258     assertTrue(labels.any { it is TaskProgressLabel.SyncGroupLockFailed })
259     assertNull((first.accessor as FakeStorageAccessor).syncLock)
260     assertNull((second.accessor as FakeStorageAccessor).syncLock)
261 }
262
263 @Test
264 fun syncGroupReleasesLocksAfterSuccessfulSync() = runBlocking {
265     val source = FakeStorage()
266     val target = FakeStorage()
267     source.addSyncJournalEntry(

```

```

268         StorageSyncJournalEntry(
269             path = "a.txt",
270             operation = StorageSyncOperation.UPSERT,
271             revision = StorageSyncRevision(1L, "x", Instant.EPOCH),
272         ),
273     )
274     source.putFile("a.txt", "x".encodeToByteArray())
275     val group = StorageSyncGroup(
276         id = "ok",
277         storageUuids = setOf(source.uuid, target.uuid),
278         encryptionKind = StorageSyncGroupEncryptionKind.NONE,
279     )
280     val engine = createEngine(
281         storages = listOf(source, target),
282         groups = listOf(group),
283     )
284     engine.syncGroup(group.id) { _, _ -> }
285     assertNull((source.accessor as FakeStorageAccessor).syncLock)
286     assertNull((target.accessor as FakeStorageAccessor).syncLock)
287 }
288
289 @Test
290 fun syncGroupReleasesLocksWhenJournalReadFails() = runBlocking {
291     val first = FakeStorage()
292     val second = FakeStorage()
293     (first.accessor as FakeStorageAccessor).readSyncJournalThrows =
294         IllegalStateException("journal read failed")
295     val group = StorageSyncGroup(
296         id = "journal-fail",
297         storageUuids = setOf(first.uuid, second.uuid),
298         encryptionKind = StorageSyncGroupEncryptionKind.NONE,
299     )
300     val engine = createEngine(
301         storages = listOf(first, second),
302         groups = listOf(group),
303     )
304     runCatching { engine.syncGroup(group.id) { _, _ -> } }
305     assertNull((first.accessor as FakeStorageAccessor).syncLock)
306     assertNull((second.accessor as FakeStorageAccessor).syncLock)
307 }
308
309 @Test
310 fun syncGroupCooperativeCancellationReleasesLocks() = runBlocking {
311     val source = FakeStorage()
312     val target = FakeStorage()

```

```

313     val path = "slow.txt"
314     val payload = "payload".encodeToByteArray()
315     source.putFile(path, payload)
316     (source.accessor as FakeStorageAccessor).openReadDelayMs = 5_000
317     source.addSyncJournalEntry(
318         StorageSyncJournalEntry(
319             path = path,
320             operation = StorageSyncOperation.UPSERT,
321             revision = StorageSyncRevision(1L, "actor", Instant.EPOCH),
322             size = payload.size.toLong(),
323         ),
324     )
325     val group = StorageSyncGroup(
326         id = "cancel-group",
327         storageUuids = setOf(source.uuid, target.uuid),
328         encryptionKind = StorageSyncGroupEncryptionKind.NONE,
329     )
330     val engine = createEngine(
331         storages = listOf(source, target),
332         groups = listOf(group),
333     )
334     val job = async {
335         engine.syncGroup(group.id) { _, _ -> }
336     }
337     kotlinx.coroutines.delay(50)
338     job.cancel()
339     try {
340         job.await()
341     } catch (_: CancellationException) {
342         // expected
343     }
344     assertNull((source.accessor as FakeStorageAccessor).syncLock)
345     assertNull((target.accessor as FakeStorageAccessor).syncLock)
346 }
347
348 @Test
349 fun syncGroupReleasesLocksWhenJournalEmpty() = runBlocking {
350     val first = FakeStorage()
351     val second = FakeStorage()
352     val group = StorageSyncGroup(
353         id = "empty-journal",
354         storageUuids = setOf(first.uuid, second.uuid),
355         encryptionKind = StorageSyncGroupEncryptionKind.NONE,
356     )

```

```

357         val labels = mutableListOf<TaskProgressLabel?>()
358         val engine = createEngine(
359             storages = listOf(first, second),
360             groups = listOf(group),
361         )
362         engine.syncGroup(group.id) { _, label -> labels.add(label) }
363         assertTrue(labels.any { it is TaskProgressLabel.SyncGroupNoJournalEntries })
364         assertNull((first.accessor as FakeStorageAccessor).syncLock)
365         assertNull((second.accessor as FakeStorageAccessor).syncLock)
366     }
367
368     private fun createEngine(
369         storages: List<FakeStorage>,
370         groups: List<StorageSyncGroup>,
371     ): StorageSyncEngine {
372         val vaultsManager = FakeVaultsManager(storages)
373         val findStorage = FindStorageUseCase(vaultsManager)
374         return StorageSyncEngine(
375             vaultsManager = vaultsManager,
376             groupStore = FakeStorageSyncGroupStore(groups),
377             findStorageUseCase = findStorage,
378         )
379     }
380 }
381

```

Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/
StorageSyncJournalMergeTest.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalMerge
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation
6 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths
7 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncRevision
8 import org.junit.Assert.assertEquals
9 import org.junit.Assert.assertFalse
10 import org.junit.Assert.assertTrue
11 import org.junit.Test
12 import java.time.Instant
13
14 class StorageSyncJournalMergeTest {
15
16     @Test
17     fun mergeKeepsSingleEntryPerPath() {
18         val older = entry(path = "/a.txt", sequence = 1L)
19         val newer = entry(path = "/a.txt", sequence = 2L)
20         val merged = StorageSyncJournalMerge.merge(
21             mapOf("/a.txt" to older),
22             mapOf("/a.txt" to newer),
23         )
24         assertEquals(1, merged.size)
25         assertEquals(2L, merged["/a.txt"]?.revision?.sequence)
26     }
27
28     @Test
29     fun isSyncableUserPathExcludesEncDirAndJournal() {
30         assertFalse(StorageSyncPaths.isSyncableUserPath("/88a048ed-enc-dir/sync-
31 journal.json"))
32         assertFalse(StorageSyncPaths.isSyncableUserPath("/wallenc-yandex-system/sync-
33 journal.json"))
34         assertTrue(StorageSyncPaths.isSyncableUserPath("/wallenc-data/text-secrets.json"))
35     }
36
37     private fun entry(path: String, sequence: Long): StorageSyncJournalEntry =
38         StorageSyncJournalEntry(
39             path = path,
40             operation = StorageSyncOperation.UPSERT,
41             revision = StorageSyncRevision(
42                 sequence = sequence,
43                 actorId = "actor",
44             )
45         )
46 }
```

```
42         createdAt = Instant.EPOCH,
43     ),
44 )
45 }
46
```

Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/
TwoFaTotpTest.kt

```
1 package com.github.nullptroma.wallenc.usecases
2
3 import com.github.nullptroma.wallenc.domain.datatypes.TwoFaTokenRecord
4 import com.eatthepath.otp.TimeBasedOneTimePasswordGenerator
5 import org.junit.Assert.assertEquals
6 import org.junit.Assert.assertNotNull
7 import org.junit.Assert.assertTrue
8 import org.junit.Test
9 import java.time.Duration
10 import java.time.Instant
11 import javax.crypto.spec.SecretKeySpec
12
13 class TwoFaTotpTest {
14
15     @Test
16     fun buildTwoFaCodeStateMatchesJava0tpForKnownSecret() {
17         val secretBase32 = "JBSWY3DPEHPK3PXP"
18         val token = TwoFaTokenRecord(
19             id = "1",
20             issuer = "Test",
21             account = "user",
22             secret = secretBase32,
23             digits = 6,
24             periodSeconds = 30,
25             algorithm = "SHA1",
26             notes = null,
27         )
28         val nowMillis = 1_700_000_000_000L
29         val state = buildTwoFaCodeState(token, nowMillis)
30         assertNotNull(state)
31         val expected = generateReferenceTotp(secretBase32, nowMillis, 6, 30, "HmacSHA1")
32         assertEquals(expected, state!!.code)
33         assertTrue(state.secondsUntilRefresh in 1..30)
34     }
35
36     @Test
37     fun totpPeriodProgressIsContinuousWithinPeriod() {
38         val period = 30
39         val periodStartMillis = 1_700_000_100_000L // epoch sec кратна period
40         assertEquals(0f, totpPeriodProgress(periodStartMillis, period), 0.001f)
41         assertEquals(0.5f, totpPeriodProgress(periodStartMillis + 15_000L, period), 0.001f)
```



```

42         assertEquals(29f / 30f, totpPeriodProgress(periodStartMillis + 29_000L, period),
0.001f)
43     }
44
45     @Test
46     fun totpSecondsUntilRefreshCountsDownWithinPeriod() {
47         val period = 30
48         val t0 = 1_700_000_100_000L
49         assertEquals(30, totpSecondsUntilRefresh(t0, period))
50         assertEquals(15, totpSecondsUntilRefresh(t0 + 15_000L, period))
51         assertEquals(1, totpSecondsUntilRefresh(t0 + 29_000L, period))
52     }
53
54     @Test
55     fun buildTwoFaCodeStateReturnsNullForInvalidSecret() {
56         val token = TwoFaTokenRecord(
57             id = "1",
58             issuer = "Test",
59             account = "user",
60             secret = "!!!not-base32!!!",
61             digits = 6,
62             periodSeconds = 30,
63             algorithm = "SHA1",
64             notes = null,
65         )
66         assertEquals(null, buildTwoFaCodeState(token, System.currentTimeMillis()))
67     }
68
69     private fun generateReferenceTotp(
70         secretBase32: String,
71         nowMillis: Long,
72         digits: Int,
73         periodSeconds: Int,
74         macAlgorithm: String,
75     ): String {
76         val key = decodeBase32(secretBase32)
77         val generator = TimeBasedOneTimePasswordGenerator(
78             Duration.ofSeconds(periodSeconds.toLong()),
79             digits,
80             macAlgorithm,
81         )
82         val otp = generator.generateOneTimePassword(
83             SecretKeySpec(key, "RAW"),
84             Instant.ofEpochMilli(nowMillis),
85         )

```

```

86         return otp.toString().padStart(digits, '0')
87     }
88
89     private fun decodeBase32(input: String): ByteArray {
90         val alphabet = "ABCDEFGHJKLMNPQRSTUVWXYZ234567"
91         val clean = input.uppercase().replace(" ", "").replace("=", "")
92         var buffer = 0
93         var bitsLeft = 0
94         val out = ArrayList<Byte>()
95         for (ch in clean) {
96             val value = alphabet.indexOf(ch)
97             buffer = (buffer shl 5) or value
98             bitsLeft += 5
99             if (bitsLeft >= 8) {
100                 out.add(((buffer shr (bitsLeft - 8)) and 0xFF).toByte())
101                 bitsLeft -= 8
102             }
103         }
104         return out.toByteArray()
105     }
106 }
107

```

Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/
fakes/FakeStorage.kt

```
1 package com.github.nullptroma.wallenc.usecases.fakes
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageMetaLoadState
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
6 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalMerge
7 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
8 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
9 import com.github.nullptroma.wallenc.domain.interfaces.IStorageMetaInfo
10 import com.github.nullptroma.wallenc.domain.tasks.TaskProgress
11 import kotlinx.coroutines.flow.Flow
12 import kotlinx.coroutines.flow.MutableStateFlow
13 import kotlinx.coroutines.flow.StateFlow
14 import kotlinx.coroutines.flow.flowOf
15 import java.time.Instant
16 import java.util.UUID
17
18 class FakeStorage(
19     override val uuid: UUID = UUID.randomUUID(),
20     private val accessorImpl: FakeStorageAccessor = FakeStorageAccessor(),
21     private val meta: FakeMetaInfo = FakeMetaInfo(),
22 ) : IStorage {
23     override val isAvailable: StateFlow<Boolean> = MutableStateFlow(true)
24     override val size: StateFlow<Long?> = MutableStateFlow(0L)
25     override val numberOfFiles: StateFlow<Int?> = MutableStateFlow(0)
26     override val isEmpty: Flow<Boolean?> = flowOf(true)
27     override val metaInfo: StateFlow<IStorageMetaInfo> = MutableStateFlow(meta)
28     override val metaLoadState: StateFlow<StorageMetaLoadState> =
29         MutableStateFlow(StorageMetaLoadState.Ready)
30     override val isVirtualStorage: Boolean = false
31     override val accessor: IStorageAccessor = accessorImpl
32
33     override suspend fun rename(newName: String) = Unit
34
35     override suspend fun setEncInfo(encInfo: StorageEncryptionInfo?) = Unit
36
37     override suspend fun clearAllContent(onProgress: suspend (TaskProgress) -> Unit) = Unit
38
39     fun setDomainFile(path: String, value: String) {
40         putFile(path, value.encodeToByteArray())
41     }
42 }
```

```

43         fun putFile(path: String, bytes: ByteArray) {
44             accessorImpl.dataFiles[com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths.normal]
45                 =
46                 bytes
47         }
48
49         fun fileBytes(path: String): ByteArray? =
50             accessorImpl.dataFiles[com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths.normal]
51
52         fun addSyncJournalEntry(entry: StorageSyncJournalEntry) {
53             accessorImpl.syncJournal = StorageSyncJournalMerge.merge(
54                 accessorImpl.syncJournal,
55                 mapOf(entry.path to entry),
56             )
57         }
58
59         fun syncJournalEntries(): List<StorageSyncJournalEntry> =
60             accessorImpl.syncJournal.values.toList()
61
62         fun setAcquireLockResult(result: Boolean) {
63             accessorImpl.acquireLockResult = result
64         }
65     }
66
67     class FakeMetaInfo(
68         override val encInfo: StorageEncryptionInfo? = null,
69         override val name: String = "Fake",
70         override val lastModified: Instant = Instant.now(),
71     ) : IStorageMetaInfo

```

Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/
fakes/FakeStorageAccessor.kt

```
1 package com.github.nullptroma.wallenc.usecases.fakes
2
3 import com.github.nullptroma.wallenc.domain.datatypes.DataPage
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournal
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
6 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalMerge
7 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncLock
8 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation
9 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths
10 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncRevision
11 import com.github.nullptroma.wallenc.domain.interfaces.IDirectory
12 import com.github.nullptroma.wallenc.domain.interfaces.IFile
13 import com.github.nullptroma.wallenc.domain.interfaces.IMetaInfo
14 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
15 import kotlinx.coroutines.delay
16 import kotlinx.coroutines.flow.Flow
17 import kotlinx.coroutines.flow.MutableSharedFlow
18 import kotlinx.coroutines.flow.MutableStateFlow
19 import kotlinx.coroutines.flow.SharedFlow
20 import kotlinx.coroutines.flow.StateFlow
21 import kotlinx.coroutines.flow.emptyFlow
22 import java.io.ByteArrayInputStream
23 import java.io.ByteArrayOutputStream
24 import java.io.InputStream
25 import java.io.OutputStream
26 import java.time.Instant
27
28 class FakeStorageAccessor : IStorageAccessor {
29     val dataFiles: MutableMap<String, ByteArray> = mutableMapOf()
30
31     private fun norm(path: String): String = StorageSyncPaths.normalize(path)
32     val trashedPaths: MutableSet<String> = mutableSetOf()
33     private val systemFiles: MutableMap<String, ByteArray> = mutableMapOf()
34     private val _filesUpdates = MutableSharedFlow<DataPage<IFile>>(extraBufferCapacity = 16)
35
36     var syncJournal: StorageSyncJournal = emptyMap()
37     var syncLock: StorageSyncLock? = null
38     var acquireLockResult: Boolean = true
39     var readSyncJournalThrows: Throwable? = null
40     var openReadDelayMs: Long = 0
41
42     override val size: StateFlow<Long?> = MutableStateFlow(0L)
```

```

43     override val numberOfFiles: StateFlow<Int?> = MutableStateFlow(0)
44     override val isAvailable: StateFlow<Boolean> = MutableStateFlow(true)
45     override val filesUpdates: SharedFlow<DataPage<IFile>> = _filesUpdates
46     override val dirsUpdates: SharedFlow<DataPage<IDirectory>> = MutableSharedFlow()
47
48     override suspend fun getAllFiles(): List<IFile> = emptyList()
49
50     override suspend fun getFiles(path: String): List<IFile> = emptyList()
51
52     override fun getFilesFlow(path: String): Flow<DataPage<IFile>> = emptyFlow()
53
54     override suspend fun getAllDirs(): List<IDirectory> = emptyList()
55
56     override suspend fun getDirs(path: String): List<IDirectory> = emptyList()
57
58     override fun getDirsFlow(path: String): Flow<DataPage<IDirectory>> = emptyFlow()
59
60     override suspend fun getFileInfo(path: String): IFile {
61         val key = norm(path)
62         val bytes = dataFiles[key] ?: throw IllegalStateException("File not found: $path")
63         return FakeFile(key, bytes.size.toLong())
64     }
65
66     override suspend fun getDirInfo(path: String): IDirectory {
67         error("Not implemented in tests")
68     }
69
70     override suspend fun setHidden(path: String, hidden: Boolean) = Unit
71
72     override suspend fun touchFile(path: String) = Unit
73
74     override suspend fun touchDir(path: String) = Unit
75
76     override suspend fun delete(path: String, recordSyncJournal: Boolean) {
77         dataFiles.remove(norm(path))
78         if (recordSyncJournal) {
79             recordDeleteJournal(path)
80         }
81     }
82
83     override suspend fun openWrite(path: String, recordSyncJournal: Boolean): OutputStream {
84         return object : ByteArrayOutputStream() {
85             override fun close() {
86                 dataFiles[norm(path)] = toByteArray()
87                 _filesUpdates.tryEmit(

```

```

88         DataPage(
89             listOf(FakeFile(path)),
90             pageLength = 1,
91             pageIndex = 0,
92         ),
93     )
94 }
95 }
96 }
97
98 override suspend fun openRead(path: String): InputStream {
99     if (openReadDelayMs > 0) {
100         delay(openReadDelayMs)
101     }
102     val bytes = dataFiles[norm(path)] ?: throw IllegalStateException("File not found:
$path")
103     return ByteArrayInputStream(bytes)
104 }
105
106 override suspend fun moveToTrash(path: String, recordSyncJournal: Boolean) {
107     val key = norm(path)
108     if (key in dataFiles) {
109         trashedPaths.add(key)
110         if (recordSyncJournal) {
111             recordTrashJournal(path)
112         }
113     }
114 }
115
116 override suspend fun openReadSystemFile(name: String): InputStream {
117     val bytes = systemFiles[name] ?: ByteArray(0)
118     return ByteArrayInputStream(bytes)
119 }
120
121 override suspend fun openWriteSystemFile(name: String): OutputStream {
122     return object : ByteArrayOutputStream() {
123         override fun close() {
124             systemFiles[name] = toByteArray()
125         }
126     }
127 }
128
129 override suspend fun readSyncJournal(): StorageSyncJournal {
130     readSyncJournalThrows?.let { throw it }
131     return syncJournal

```

```

132     }
133
134     override suspend fun flushPendingSyncJournal() = Unit
135
136     override suspend fun putSyncJournalEntries(entries: StorageSyncJournal) {
137         syncJournal = StorageSyncJournalMerge.merge(syncJournal, entries)
138     }
139
140     private suspend fun recordDeleteJournal(path: String) {
141         appendJournalEntry(path, StorageSyncOperation.DELETE)
142     }
143
144     private suspend fun recordTrashJournal(path: String) {
145         appendJournalEntry(path, StorageSyncOperation.TRASH)
146     }
147
148     private suspend fun appendJournalEntry(path: String, operation: StorageSyncOperation) {
149         val cleaned = norm(path)
150         val nextSequence = (syncJournal.values.maxOfOrNull { it.revision.sequence } ?: 0L) +
1L
151         putSyncJournalEntries(
152             mapOf(
153                 cleaned to StorageSyncJournalEntry(
154                     path = cleaned,
155                     operation = operation,
156                     revision = StorageSyncRevision(
157                         sequence = nextSequence,
158                         actorId = "fake-actor",
159                         createdAt = Instant.now(),
160                     ),
161                 ),
162             ),
163         )
164     }
165
166     override suspend fun readSyncLock(): StorageSyncLock? = syncLock
167
168     override suspend fun tryAcquireSyncLock(holderId: String, leaseUntil: Instant): Boolean
169     {
170         if (!acquireLockResult) return false
171         syncLock = StorageSyncLock(holderId, leaseUntil, Instant.now())
172         return true
173     }
174
175     override suspend fun releaseSyncLock(holderId: String) {

```



```

175         if (syncLock?.holderId == holderId) {
176             syncLock = null
177         }
178     }
179
180     override suspend fun forceClearSyncLock() {
181         syncLock = null
182     }
183 }
184
185 class FakeFile(
186     path: String,
187     size: Long = 0L,
188 ) : IFile {
189     override val metaInfo: IMetaInfo = object : IMetaInfo {
190         override val size: Long = size
191         override val isDeleted: Boolean = false
192         override val isHidden: Boolean = false
193         override val lastModified: Instant = Instant.now()
194         override val path: String = path
195     }
196 }
197

```

Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/
fakes/FakeStorageSyncGroupStore.kt

```
1 package com.github.nullptroma.wallenc.usecases.fakes
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncGroupStore
4 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroup
5
6 class FakeStorageSyncGroupStore(
7     private var groups: List<StorageSyncGroup> = emptyList(),
8 ) : IStorageSyncGroupStore {
9     override suspend fun getGroups(): List<StorageSyncGroup> = groups
10
11     override suspend fun putGroup(group: StorageSyncGroup) {
12         groups = groups.filterNot { it.id == group.id } + group
13     }
14
15     override suspend fun removeGroup(groupId: String) {
16         groups = groups.filterNot { it.id == groupId }
17     }
18 }
19
```

Исходный файл usecases/src/test/java/com/github/nullptroma/wallenc/usecases/
fakes/FakeVaultsManager.kt

```
1 package com.github.nullptroma.wallenc.usecases.fakes
2
3 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
4 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
5 import com.github.nullptroma.wallenc.domain.interfaces.IUnlockManager
6 import com.github.nullptroma.wallenc.domain.interfaces.IVault
7 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
8 import kotlinx.coroutines.flow.MutableStateFlow
9 import kotlinx.coroutines.flow.StateFlow
10 import java.util.UUID
11
12 class FakeVaultsManager(
13     storages: List<IStorage>,
14 ) : IVaultsManager {
15     override val vaults: StateFlow<List<IVault>> = MutableStateFlow(emptyList())
16     override val allStorages: StateFlow<List<IStorage>> = MutableStateFlow(storages)
17     override val unlockManager: IUnlockManager = FakeUnlockManager()
18 }
19
20 class FakeUnlockManager : IUnlockManager {
21     override val openedStorages: StateFlow<Map<UUID, IStorage>> =
22         MutableStateFlow(emptyMap())
23
24     override fun getOpenedStorageKey(uuid: UUID): EncryptKey? = null
25
26     override suspend fun open(storage: IStorage, key: EncryptKey, rememberPassword:
27         Boolean): IStorage = storage
28
29     override suspend fun rememberKey(storage: IStorage, key: EncryptKey) = Unit
30
31     override suspend fun close(storage: IStorage) = Unit
32
33     override suspend fun close(uuid: UUID) = Unit
34 }
```

ПРИЛОЖЕНИЕ А.5

Модуль :ui

Исходный файл ui/consumer-rules.pro

Исходный файл ui/proguard-rules.pro

```
1  # Add project specific ProGuard rules here.
2  # You can control the set of applied configuration files using the
3  # proguardFiles setting in build.gradle.
4  #
5  # For more details, see
6  #   http://developer.android.com/guide/developing/tools/proguard.html
7
8  # If your project uses WebView with JS, uncomment the following
9  # and specify the fully qualified class name to the JavaScript interface
10 # class:
11 #-keepclassmembers class fqcn.of.javascript.interface.for.webview {
12 #   public *;
13 #}
14
15 # Uncomment this to preserve the line number information for
16 # debugging stack traces.
17 #-keepattributes SourceFile,LineNumberTable
18
19 # If you keep the line number information, uncomment this to
20 # hide the original source file name.
21 #-renamesourcefileattribute SourceFile
```

Исходный файл ui/src/androidTest/java/com/github/nullptroma/wallenc/ui/screens/main/screens/storage/secrets/TextSecretsScreenContentTest.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.compose.material3.MaterialTheme
4 import androidx.compose.ui.test.assertIsDisplayed
5 import androidx.compose.ui.test.junit4.createComposeRule
6 import androidx.compose.ui.test.onNodeWithText
7 import androidx.test.ext.junit.runners.AndroidJUnit4
8 import androidx.test.platform.app.InstrumentationRegistry
9 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretEntryRecord
10 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretRecord
11 import com.github.nullptroma.wallenc.ui.R
12 import org.junit.Rule
13 import org.junit.Test
14 import org.junit.runner.RunWith
15
16 @RunWith(AndroidJUnit4::class)
17 class TextSecretsScreenContentTest {
18
19     @get:Rule
20     val composeRule = createComposeRule()
21
22     private val context get() = InstrumentationRegistry.getInstrumentation().targetContext
23
24     @Test
25     fun showsEmptyStateWhenNoSecrets() {
26         composeRule.setContent {
27             MaterialTheme {
28                 TextSecretsScreenContent(
29                     uiState = TextSecretsScreenState(
30                         isLoading = false,
31                         isAvailable = true,
32                         items = emptyList(),
33                     ),
34                 )
35             }
36         }
37         composeRule.onNodeWithText(context.getString(R.string.text_secret_empty_state)).assertIsDisplayed()
38     }
39
40     @Test
41     fun showsSecretTitle() {
42         val secret = TextSecretRecord(
```

```

43         id = "s1",
44         title = "Production DB",
45         items = listOf(TextSecretEntryRecord(label = "user", value = "admin")),
46     )
47     composeRule.setContent {
48         MaterialTheme {
49             TextSecretsScreenContent(
50                 uiState = TextSecretsScreenState(
51                     isLoading = false,
52                     isAvailable = true,
53                     items = listOf(secret),
54                 ),
55             ) {
56                 TextSecretListCard(
57                     secret = secret,
58                     onClick = {},
59                     enabled = true,
60                 )
61             }
62         }
63     }
64     composeRule.onNodeWithText("Production DB").assertIsDisplayed()
65 }
66 }
67

```

Исходный файл ui/src/androidTest/java/com/github/nullptroma/wallenc/ui/screens/
main/screens/storage/twofa/TwoFaTokensScreenContentTest.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa
2
3 import androidx.compose.foundation.layout.padding
4 import androidx.compose.material3.MaterialTheme
5 import androidx.compose.ui.Modifier
6 import androidx.compose.ui.test.junit4.createComposeRule
7 import androidx.compose.ui.test.onNodeWithText
8 import androidx.compose.ui.test.assertIsDisplayed
9 import androidx.compose.ui.unit.dp
10 import androidx.test.ext.junit.runners.AndroidJUnit4
11 import androidx.test.platform.app.InstrumentationRegistry
12 import com.github.nullptroma.wallenc.domain.datatypes.TwoFaTokenRecord
13 import com.github.nullptroma.wallenc.ui.R
14 import org.junit.Rule
15 import org.junit.Test
16 import org.junit.runner.RunWith
17
18 @RunWith(AndroidJUnit4::class)
19 class TwoFaTokensScreenContentTest {
20
21     @get:Rule
22     val composeRule = createComposeRule()
23
24     private val context get() = InstrumentationRegistry.getInstrumentation().targetContext
25
26     @Test
27     fun showsEmptyStateWhenNoTokens() {
28         composeRule.setContent {
29             MaterialTheme {
30                 TwoFaTokensScreenContent(
31                     uiState = TwoFaTokensScreenState(
32                         isLoading = false,
33                         isAvailable = true,
34                         items = emptyList(),
35                     ),
36                 )
37             }
38         }
39         composeRule.onNodeWithText(context.getString(R.string.two_fa_empty_state)).assertIsDisplayed()
40     }
41
42     @Test
```



```

43     fun showsIssuerAndAccountForToken() {
44         val token = TwoFaTokenRecord(
45             id = "1",
46             issuer = "GitHub",
47             account = "user@example.com",
48             secret = "SECRET",
49         )
50         composeRule.setContent {
51             MaterialTheme {
52                 TwoFaTokensScreenContent(
53                     uiState = TwoFaTokensScreenState(
54                         isLoading = false,
55                         isAvailable = true,
56                         items = listOf(token),
57                     ),
58                 ) {
59                     TwoFaTokenListHeader(
60                         issuer = token.issuer,
61                         account = token.account,
62                         modifier = Modifier.padding(8.dp),
63                     )
64                 }
65             }
66         }
67         composeRule.onNodeWithText("GitHub").assertIsDisplayed()
68         composeRule.onNodeWithText("user@example.com").assertIsDisplayed()
69     }
70 }
71

```

Исходный файл ui/src/main/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest>
3
4  </manifest>
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/
ViewModelBase.kt

```
1 package com.github.nullptroma.wallenc.ui
2
3
4 import androidx.lifecycle.ViewModel
5 import kotlinx.coroutines.flow.MutableStateFlow
6 import kotlinx.coroutines.flow.StateFlow
7
8 abstract class ViewModelBase<TState>(initState: TState) : ViewModel() {
9     private val _state = MutableStateFlow(initState)
10
11     val state: StateFlow<TState>
12         get() = _state
13
14     protected fun updateState(newState: TState) {
15         _state.value = newState
16     }
17 }
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/WallencUi.kt

```
1 package com.github.nullptroma.wallenc.ui
2
3 import android.content.Intent
4 import androidx.compose.animation.core.tween
5 import androidx.compose.animation.fadeIn
6 import androidx.compose.animation.fadeOut
7 import androidx.compose.foundation.layout.Box
8 import androidx.compose.foundation.layout.navigationBarsPadding
9 import androidx.compose.foundation.layout.padding
10 import androidx.compose.material.icons.Icons
11 import androidx.compose.material.icons.automirrored.rounded.List
12 import androidx.compose.material.icons.rounded.Menu
13 import androidx.compose.material.icons.rounded.Settings
14 import androidx.compose.material.icons.rounded.Sync
15 import androidx.compose.material3.ExperimentalMaterial3Api
16 import androidx.compose.material3.Scaffold
17 import androidx.compose.material3.Surface
18 import androidx.compose.runtime.Composable
19 import androidx.compose.runtime.LaunchedEffect
20 import androidx.compose.runtime.getValue
21 import androidx.compose.runtime.remember
22 import androidx.compose.ui.Modifier
23 import androidx.compose.ui.unit.dp
24 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
25 import androidx.navigation.compose.NavHost
26 import androidx.navigation.compose.composable
27 import androidx.navigation.compose.currentBackStackEntryAsState
28 import androidx.navigation.navDeepLink
29 import com.github.nullptroma.wallenc.ui.elements.FloatingWallencNavigationBar
30 import com.github.nullptroma.wallenc.ui.navigation.NavBarItemData
31 import com.github.nullptroma.wallenc.ui.navigation.WallencDeepLinks
32 import com.github.nullptroma.wallenc.ui.navigation.matchesWallencDeepLink
33 import com.github.nullptroma.wallenc.ui.navigation.rememberNavigationState
34 import com.github.nullptroma.wallenc.ui.screens.main.MainRoute
35 import com.github.nullptroma.wallenc.ui.screens.main.MainScreen
36 import com.github.nullptroma.wallenc.ui.screens.main.MainViewModel
37 import com.github.nullptroma.wallenc.ui.screens.main.screens.tasks.TaskPipelineRoute
38 import com.github.nullptroma.wallenc.ui.screens.main.screens.tasks.TaskPipelineScreen
39 import com.github.nullptroma.wallenc.ui.screens.settings.SettingsRoute
40 import com.github.nullptroma.wallenc.ui.screens.settings.SettingsScreen
41 import com.github.nullptroma.wallenc.ui.screens.settings.SettingsViewModel
42 import com.github.nullptroma.wallenc.ui.screens.sync.StorageSyncRoute
43 import com.github.nullptroma.wallenc.ui.screens.sync.StorageSyncScreen
```

```

44 import com.github.nullptroma.wallenc.ui.screens.sync.StorageSyncViewModel
45 import com.github.nullptroma.wallenc.ui.theme.WallencTheme
46
47
48 @Composable
49 fun WallencUi(
50     deepLinkPulse: Int = 0,
51     deepLinkIntent: Intent = Intent(),
52 ) {
53     WallencTheme {
54         Surface {
55             WallencNavRoot(
56                 deepLinkPulse = deepLinkPulse,
57                 deepLinkIntent = deepLinkIntent,
58             )
59         }
60     }
61 }
62
63 @OptIn(ExperimentalMaterial3Api::class)
64 @Composable
65 fun WallencNavRoot(
66     viewModel: WallencViewModel = hiltViewModel(),
67     deepLinkPulse: Int = 0,
68     deepLinkIntent: Intent = Intent(),
69 ) {
70     val navState = rememberNavigationState()
71
72     val mainViewModel: MainViewModel = hiltViewModel()
73     val settingsViewModel: SettingsViewModel = hiltViewModel()
74     val storageSyncViewModel: StorageSyncViewModel = hiltViewModel()
75
76     val topLevelRoutes = viewModel.routes
77
78     LaunchedEffect(deepLinkPulse) {
79         if (deepLinkPulse <= 0) return@LaunchedEffect
80         if (!deepLinkIntent.matchesWallencDeepLink()) return@LaunchedEffect
81         navState.navHostController.handleDeepLink(deepLinkIntent)
82     }
83
84     val topLevelNavBarItems = remember {
85         mapOf(
86             MainRoute::class.qualifiedName!! to NavBarItemData(
87                 R.string.nav_label_main, MainRoute::class.qualifiedName!!,
88                 Icons.Rounded.Menu

```

```

88         ),
89         TaskPipelineRoute::class.qualifiedName!! to NavBarItemData(
90             R.string.task_pipeline_title,
91             TaskPipelineRoute::class.qualifiedName!!,
92             Icons.AutoMirrored.Rounded.List,
93             R.string.task_pipeline_open,
94         ),
95         StorageSyncRoute::class.qualifiedName!! to NavBarItemData(
96             R.string.nav_label_sync,
97             StorageSyncRoute::class.qualifiedName!!,
98             Icons.Rounded.Sync,
99         ),
100        SettingsRoute::class.qualifiedName!! to NavBarItemData(
101            R.string.nav_label_settings,
102            SettingsRoute::class.qualifiedName!!,
103            Icons.Rounded.Settings,
104        ),
105    )
106 }
107
108 val navBackStackEntry by navState.navHostController.currentBackStackEntryAsState()
109 val currentRoute = navBackStackEntry?.destination?.route
110
111 Scaffold(
112     bottomBar = {
113         Box(
114             modifier = Modifier
115                 .navigationBarsPadding()
116                 .padding(horizontal = 12.dp)
117                 .padding(top = 4.dp, bottom = 6.dp),
118         ) {
119             FloatingWallencNavigationBar(
120                 items = topLevelNavBarItems,
121                 routes = topLevelRoutes,
122                 currentRoute = currentRoute,
123                 onNavigate = { item ->
124                     val route = topLevelRoutes[item.screenRouteClass]
125                     ?: error("Route ${item.screenRouteClass} not found")
126                     if (currentRoute?.startsWith(item.screenRouteClass) != true) {
127                         navState.changeTop(route)
128                     }
129                 },
130             )
131         }
132     },

```

```

133     ) { innerPaddings ->
134         NavHost(
135             navState.navHostController,
136             startDestination = topLevelRoutes[MainRoute::class.qualifiedName]!!,
137             modifier = Modifier.padding(innerPaddings),
138         ) {
139             composable<MainRoute>(
140                 deepLinks = listOf(
141                     navDeepLink { uriPattern = WallencDeepLinks.MAIN_URI_PATTERN },
142                 ),
143                 enterTransition = {
144                     fadeIn(tween(200))
145                 },
146                 exitTransition = {
147                     fadeOut(tween(200))
148                 },
149             ) {
150                 MainScreen(
151                     modifier = Modifier,
152                     viewModel = mainViewModel,
153                 )
154             }
155             composable<SettingsRoute>(
156                 deepLinks = listOf(
157                     navDeepLink { uriPattern = WallencDeepLinks.SETTINGS_URI_PATTERN },
158                 ),
159                 enterTransition = {
160                     fadeIn(tween(200))
161                 },
162                 exitTransition = {
163                     fadeOut(tween(200))
164                 },
165             ) {
166                 SettingsScreen(Modifier, settingsViewModel)
167             }
168             composable<StorageSyncRoute>(
169                 deepLinks = listOf(
170                     navDeepLink { uriPattern = WallencDeepLinks.SYNC_URI_PATTERN },
171                 ),
172                 enterTransition = {
173                     fadeIn(tween(200))
174                 },
175                 exitTransition = {
176                     fadeOut(tween(200))
177                 },

```

```

178         ) {
179             StorageSyncScreen(
180                 modifier = Modifier,
181                 viewModel = storageSyncViewModel,
182             )
183         }
184     composable<TaskPipelineRoute>(
185         deepLinks = listOf(
186             navDeepLink { uriPattern = WallencDeepLinks.TASKS_URI_PATTERN },
187         ),
188         enterTransition = {
189             fadeIn(tween(200))
190         },
191         exitTransition = {
192             fadeOut(tween(200))
193         },
194     ) {
195         TaskPipelineScreen(modifier = Modifier)
196     }
197 }
198 }
199 }
200

```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/
WallencViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui
2
3 import androidx.compose.runtime.mutableStateOf
4 import androidx.lifecycle.SavedStateHandle
5 import androidx.lifecycle.viewmodel.compose.SavedStateHandleSaveableApi
6 import androidx.lifecycle.viewmodel.compose.saveable
7 import com.github.nullptroma.wallenc.ui.screens.main.MainRoute
8 import com.github.nullptroma.wallenc.ui.screens.main.screens.tasks.TaskPipelineRoute
9 import com.github.nullptroma.wallenc.ui.screens.settings.SettingsRoute
10 import com.github.nullptroma.wallenc.ui.screens.sync.StorageSyncRoute
11 import dagger.hilt.android.lifecycle.HiltViewModel
12
13 @HiltViewModel
14 class WallencViewModel @javax.inject.Inject constructor(savedStateHandle:
15 SavedStateHandle) :
16     ViewModelBase<Unit>(Unit) {
17     @OptIn(SavedStateHandleSaveableApi::class)
18     var routes by savedStateHandle.saveable {
19         mutableStateOf(
20             mapOf(
21                 MainRoute::class.qualifiedName!! to MainRoute(),
22                 TaskPipelineRoute::class.qualifiedName!! to TaskPipelineRoute(),
23                 StorageSyncRoute::class.qualifiedName!! to StorageSyncRoute(),
24                 SettingsRoute::class.qualifiedName!! to SettingsRoute()
25             )
26         )
27     }
28     private set
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/elements/ Dialogs.kt

```
1 package com.github.nullptroma.wallenc.ui.elements
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Row
6 import androidx.compose.foundation.layout.Spacer
7 import androidx.compose.foundation.layout.fillMaxWidth
8 import androidx.compose.foundation.layout.height
9 import androidx.compose.foundation.layout.padding
10 import androidx.compose.foundation.layout.width
11 import androidx.compose.material3.BasicAlertDialog
12 import androidx.compose.material3.Button
13 import androidx.compose.material3.Card
14 import androidx.compose.material3.Checkbox
15 import androidx.compose.material3.ExperimentalMaterial3Api
16 import androidx.compose.material3.MaterialTheme
17 import androidx.compose.material3.Text
18 import androidx.compose.material3.TextField
19 import androidx.compose.runtime.Composable
20 import androidx.compose.runtime.LaunchedEffect
21 import androidx.compose.runtime.getValue
22 import androidx.compose.runtime.mutableStateOf
23 import androidx.compose.runtime.remember
24 import androidx.compose.runtime.setValue
25 import androidx.compose.ui.Alignment
26 import androidx.compose.ui.Modifier
27 import androidx.compose.ui.focus.FocusRequester
28 import androidx.compose.ui.focus.focusRequester
29 import androidx.compose.ui.res.stringResource
30 import androidx.compose.ui.unit.dp
31 import com.github.nullptroma.wallenc.ui.R
32
33 @OptIn(ExperimentalMaterial3Api::class)
34 @Composable
35 fun TextEditCancelOkDialog(
36     onDismiss: () -> Unit,
37     onConfirmation: (String) -> Unit,
38     title: String,
39     startString: String = "",
40 ) {
41     var name by remember { mutableStateOf(startString) }
42     val focusRequester = remember { FocusRequester() }
```

```

43     BasicAlertDialog(
44         onDismissRequest = { onDismiss() },
45     ) {
46         Card {
47             Column(modifier = Modifier.padding(12.dp)) {
48                 Text(title, style = MaterialTheme.typography.titleLarge)
49                 TextField(
50                     modifier = Modifier.focusRequester(focusRequester),
51                     value = name,
52                     onValueChange = { name = it },
53                 )
54                 Spacer(modifier = Modifier.height(24.dp))
55                 Row(
56                     modifier = Modifier.fillMaxWidth(),
57                     horizontalArrangement = Arrangement.SpaceEvenly,
58                 ) {
59                     Button(modifier = Modifier.weight(1f), onClick = onDismiss) {
60                         Text(stringResource(R.string.dialog_cancel))
61                     }
62                     Spacer(modifier = Modifier.width(12.dp))
63                     Button(
64                         modifier = Modifier.weight(1f),
65                         onClick = {
66                             onConfirmation(name)
67                         },
68                     ) {
69                         Text(stringResource(R.string.dialog_ok))
70                     }
71                 }
72             }
73         }
74     }
75
76     LaunchedEffect(Unit) {
77         focusRequester.requestFocus()
78     }
79 }
80
81 @OptIn(ExperimentalMaterial3Api::class)
82 @Composable
83 fun ConfirmationCancelOkDialog(onDismiss: () -> Unit, onConfirmation: () -> Unit, title:
String) {
84     BasicAlertDialog(
85         onDismissRequest = { onDismiss() },
86     ) {

```

```

87         Card {
88             Column(modifier = Modifier.padding(12.dp)) {
89                 Text(title, style = MaterialTheme.typography.titleLarge)
90                 Spacer(modifier = Modifier.height(24.dp))
91                 Row(
92                     modifier = Modifier.fillMaxWidth(),
93                     horizontalArrangement = Arrangement.SpaceEvenly,
94                 ) {
95                     Button(modifier = Modifier.weight(1f), onClick = onDismiss) {
96                         Text(stringResource(R.string.dialog_cancel))
97                     }
98                     Spacer(modifier = Modifier.width(12.dp))
99                     Button(
100                         modifier = Modifier.weight(1f),
101                         onClick = {
102                             onConfirmation()
103                         },
104                     ) {
105                         Text(stringResource(R.string.dialog_ok))
106                     }
107                 }
108             }
109         }
110     }
111 }
112
113 @OptIn(ExperimentalMaterial3Api::class)
114 @Composable
115 fun EncryptionSetupDialog(
116     onDismiss: () -> Unit,
117     onConfirmation: (password: String, encryptPath: Boolean, rememberPassword: Boolean) -> Unit,
118 ) {
119     var password by remember { mutableStateOf("") }
120     var encryptPath by remember { mutableStateOf(false) }
121     var rememberPassword by remember { mutableStateOf(true) }
122     BasicAlertDialog(onDismissRequest = onDismiss) {
123         Card {
124             Column(modifier = Modifier.padding(12.dp)) {
125                 Text(
126                     stringResource(R.string.dialog_encryption_enable_title),
127                     style = MaterialTheme.typography.titleLarge,
128                 )
129                 Spacer(modifier = Modifier.height(12.dp))
130                 TextField(

```

```

131         value = password,
132         onChange = { password = it },
133         label = { Text(stringResource(R.string.dialog_password_label)) },
134     )
135     Row(verticalAlignment = Alignment.CenterVertically) {
136         Checkbox(checked = encryptPath, onChange = { encryptPath = it })
137         Text(stringResource(R.string.dialog_encrypt_paths))
138     }
139     Row(verticalAlignment = Alignment.CenterVertically) {
140         Checkbox(checked = rememberPassword, onChange = {
141             rememberPassword = it })
142         Text(stringResource(R.string.dialog_remember_password))
143     }
144     Spacer(modifier = Modifier.height(16.dp))
145     Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement =
Arrangement.SpaceEvenly) {
146         Button(modifier = Modifier.weight(1f), onClick = onDismiss) {
147             Text(stringResource(R.string.dialog_cancel))
148         }
149         Spacer(modifier = Modifier.width(12.dp))
150         Button(
151             modifier = Modifier.weight(1f),
152             onClick = { onConfirmation(password, encryptPath,
rememberPassword) },
153             enabled = password.isNotEmpty(),
154         ) {
155             Text(stringResource(R.string.dialog_apply))
156         }
157     }
158 }
159 }
160 }
161
162 @OptIn(ExperimentalMaterial3Api::class)
163 @Composable
164 fun OpenEncryptedStorageDialog(
165     onDismiss: () -> Unit,
166     onConfirmation: (password: String, rememberPassword: Boolean) -> Unit,
167 ) {
168     var password by remember { mutableStateOf("") }
169     var rememberPassword by remember { mutableStateOf(false) }
170     BasicAlertDialog(onDismissRequest = onDismiss) {
171         Card {
172             Column(modifier = Modifier.padding(12.dp)) {

```

```

173         Text(
174             stringResource(R.string.dialog_open_encrypted_title),
175             style = MaterialTheme.typography.titleLarge,
176         )
177         Spacer(modifier = Modifier.height(12.dp))
178         TextField(
179             value = password,
180             onChange = { password = it },
181             label = { Text(stringResource(R.string.dialog_password_label)) },
182         )
183         Row(verticalAlignment = Alignment.CenterVertically) {
184             Checkbox(checked = rememberPassword, onCheckedChange =
{ rememberPassword = it })
185             Text(stringResource(R.string.dialog_remember_password))
186         }
187         Spacer(modifier = Modifier.height(16.dp))
188         Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement =
Arrangement.SpaceEvenly) {
189             Button(modifier = Modifier.weight(1f), onClick = onDismiss) {
190                 Text(stringResource(R.string.dialog_cancel))
191             }
192             Spacer(modifier = Modifier.width(12.dp))
193             Button(
194                 modifier = Modifier.weight(1f),
195                 onClick = { onConfirmation(password, rememberPassword) },
196                 enabled = password.isNotEmpty(),
197             ) {
198                 Text(stringResource(R.string.dialog_open))
199             }
200         }
201     }
202 }
203 }
204 }
205
206 @OptIn(ExperimentalMaterial3Api::class)
207 @Composable
208 fun StorageEncryptionActionsDialog(
209     onDismiss: () -> Unit,
210     title: String,
211     isOpened: Boolean,
212     onOpen: () -> Unit,
213     onClose: () -> Unit,
214     onDisable: () -> Unit,
215 ) {
216     BasicAlertDialog(onDismissRequest = onDismiss) {

```

```

217     Card {
218         Column(modifier = Modifier.padding(12.dp)) {
219             Text(title, style = MaterialTheme.typography.titleLarge)
220             Spacer(modifier = Modifier.height(12.dp))
221             if (isOpened) {
222                 Button(onClick = onClose, modifier = Modifier.fillMaxWidth()) {
223                     Text(stringResource(R.string.dialog_close))
224                 }
225             } else {
226                 Button(onClick = onOpen, modifier = Modifier.fillMaxWidth()) {
227                     Text(stringResource(R.string.dialog_open))
228                 }
229             }
230             Spacer(modifier = Modifier.height(8.dp))
231             Button(onClick = onDisable, modifier = Modifier.fillMaxWidth()) {
232                 Text(stringResource(R.string.dialog_disable_encryption))
233             }
234             Spacer(modifier = Modifier.height(12.dp))
235             Button(onClick = onDismiss, modifier = Modifier.fillMaxWidth()) {
236                 Text(stringResource(R.string.dialog_done))
237             }
238         }
239     }
240 }
241 }
242

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/elements/ FloatingBackButton.kt

```
1 package com.github.nullptroma.wallenc.ui.elements
2
3 import androidx.compose.animation.AnimatedVisibility
4 import androidx.compose.animation.core.tween
5 import androidx.compose.animation.fadeIn
6 import androidx.compose.animation.fadeOut
7 import androidx.compose.animation.slideInVertically
8 import androidx.compose.animation.slideOutVertically
9 import androidx.compose.foundation.layout.padding
10 import androidx.compose.foundation.layout.size
11 import androidx.compose.foundation.shape.CircleShape
12 import androidx.compose.material.icons.Icons
13 import androidx.compose.material.icons.automirrored.rounded.ArrowBack
14 import androidx.compose.material3.ExperimentalMaterial3Api
15 import androidx.compose.material3.Icon
16 import androidx.compose.material3.MaterialTheme
17 import androidx.compose.material3.Surface
18 import androidx.compose.runtime.Composable
19 import androidx.compose.ui.Modifier
20 import androidx.compose.ui.res.stringResource
21 import androidx.compose.ui.unit.dp
22 import com.github.nullptroma.wallenc.ui.R
23
24 private const val BackButtonAnimMillis = 200
25
26 private val backButtonEnter = fadeIn(tween(BackButtonAnimMillis)) +
27     slideInVertically(
28         animationSpec = tween(BackButtonAnimMillis),
29         initialOffsetY = { fullHeight -> fullHeight / 2 },
30     )
31
32 private val backButtonExit = fadeOut(tween(BackButtonAnimMillis)) +
33     slideOutVertically(
34         animationSpec = tween(BackButtonAnimMillis),
35         targetOffsetY = { fullHeight -> -fullHeight / 2 },
36     )
37
38 @OptIn(ExperimentalMaterial3Api::class)
39 @Composable
40 fun FloatingBackButton(
41     onClick: () -> Unit,
42     modifier: Modifier = Modifier,
```



```

43     ) {
44         Surface(
45             onClick = onClick,
46             modifier = modifier.size(44.dp),
47             shape = CircleShape,
48             color = MaterialTheme.colorScheme.surfaceContainerHigh,
49             shadowElevation = 4.dp,
50             tonalElevation = 2.dp,
51             contentColor = MaterialTheme.colorScheme.onSurface,
52         ) {
53             Icon(
54                 imageVector = Icons.AutoMirrored.Rounded.ArrowBack,
55                 contentDescription = stringResource(R.string.nav_cd_back),
56                 modifier = Modifier.padding(10.dp),
57             )
58         }
59     }
60
61     @Composable
62     fun AnimatedFloatingBackButton(
63         visible: Boolean,
64         onClick: () -> Unit,
65         modifier: Modifier = Modifier,
66     ) {
67         AnimatedVisibility(
68             visible = visible,
69             modifier = modifier,
70             enter = backButtonEnter,
71             exit = backButtonExit,
72         ) {
73             FloatingBackButton(onClick = onClick)
74         }
75     }
76

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/elements/ FloatingWallencNavigationBar.kt

```
1 package com.github.nullptroma.wallenc.ui.elements
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Box
5 import androidx.compose.foundation.layout.Row
6 import androidx.compose.foundation.layout.fillMaxWidth
7 import androidx.compose.foundation.layout.height
8 import androidx.compose.foundation.layout.padding
9 import androidx.compose.foundation.layout.size
10 import androidx.compose.foundation.layout.widthIn
11 import androidx.compose.foundation.shape.RoundedCornerShape
12 import androidx.compose.material3.ExperimentalMaterial3Api
13 import androidx.compose.material3.Icon
14 import androidx.compose.material3.MaterialTheme
15 import androidx.compose.material3.Surface
16 import androidx.compose.runtime.Composable
17 import androidx.compose.ui.Alignment
18 import androidx.compose.ui.Modifier
19 import androidx.compose.ui.graphics.vector.ImageVector
20 import androidx.compose.ui.hapticfeedback.HapticFeedbackType
21 import androidx.compose.ui.platform.LocalHapticFeedback
22 import androidx.compose.ui.res.stringResource
23 import androidx.compose.ui.semantics.Role
24 import androidx.compose.ui.semantics.contentDescription
25 import androidx.compose.ui.semantics.role
26 import androidx.compose.ui.semantics.semantics
27 import androidx.compose.ui.unit.dp
28 import com.github.nullptroma.wallenc.ui.navigation.NavBarItemData
29
30 /** Вертикальный зазор между вложенной и корневой плавающими панелями навигации. */
31 val WallencNestedNavBarGap = 2.dp
32
33 @Composable
34 fun FloatingWallencNavigationBar(
35     items: Map<String, NavBarItemData>,
36     routes: Map<String, *>,
37     currentRoute: String?,
38     onNavigate: (NavBarItemData) -> Unit,
39     modifier: Modifier = Modifier,
40     compact: Boolean = false,
41 ) {
42     val haptic = LocalHapticFeedback.current
```

```

43     val barHeight = if (compact) 48.dp else 56.dp
44     val barShape = if (compact) RoundedCornerShape(22.dp) else RoundedCornerShape(28.dp)
45     val barHorizontalPadding = if (compact) 4.dp else 6.dp
46
47     val barSurface: @Composable () -> Unit = {
48         Surface(
49             modifier = Modifier
50                 .then(
51                     if (compact) {
52                         Modifier
53                             .widthIn(max = 300.dp)
54                             .fillMaxWidth(0.68f)
55                     } else {
56                         Modifier.fillMaxWidth()
57                     },
58                 ),
59             shape = barShape,
60             color = MaterialTheme.colorScheme.surfaceContainerHigh,
61             shadowElevation = 6.dp,
62             tonalElevation = 2.dp,
63         ) {
64             Row(
65                 modifier = Modifier
66                     .fillMaxWidth()
67                     .height(barHeight)
68                     .padding(horizontal = barHorizontalPadding, vertical = 4.dp),
69                 verticalAlignment = Alignment.CenterVertically,
70             ) {
71                 items.forEach { (routeClassName, navBarItemData) ->
72                     val iconVector = navBarItemData.icon ?: return@forEach
73                     val selected = currentRoute?.startsWith(routeClassName) == true
74                     val enabled = routes[navBarItemData.screenRouteClass] != null
75                     FloatingNavItem(
76                         modifier = Modifier
77                             .weight(1f)
78                             .fillMaxWidth(),
79                         icon = iconVector,
80                         label = stringResource(navBarItemData.nameStringResourceId),
81                         contentDescription =
stringResource(navBarItemData.iconContentDescriptionResourceId),
82                         selected = selected,
83                         enabled = enabled,
84                         compact = compact,
85                         onClick = {
86                             if (!selected && enabled) {

```

```

87     haptic.performHapticFeedback(HapticFeedbackType.TextHandleMove)
88         navigate(navBarItemData)
89     },
90     ),
91     )
92 }
93 }
94 }
95 }
96
97 if (compact) {
98     Box(
99         modifier = modifier.fillMaxWidth(),
100         contentAlignment = Alignment.Center,
101     ) {
102         barSurface()
103     }
104 } else {
105     Box(modifier = modifier.fillMaxWidth()) {
106         barSurface()
107     }
108 }
109 }
110
111 @OptIn(ExperimentalMaterial3Api::class)
112 @Composable
113 private fun FloatingNavItem(
114     icon: ImageVector,
115     label: String,
116     contentDescription: String,
117     selected: Boolean,
118     enabled: Boolean,
119     compact: Boolean,
120     onClick: () -> Unit,
121     modifier: Modifier = Modifier,
122 ) {
123     val iconSize = if (compact) 22.dp else 24.dp
124     val itemPaddingH = if (compact) 6.dp else 8.dp
125     val itemPaddingV = if (compact) 6.dp else 8.dp
126     val labelStyle = if (compact) {
127         MaterialTheme.typography.labelSmall
128     } else {
129         MaterialTheme.typography.labelMedium
130     }

```

```

131     val labelVelocity = if (compact) 24.dp else 28.dp
132     val itemShape = if (compact) RoundedCornerShape(16.dp) else RoundedCornerShape(20.dp)
133     val containerColor = if (selected) {
134         MaterialTheme.colorScheme.secondaryContainer
135     } else {
136         MaterialTheme.colorScheme.surfaceContainerHigh
137     }
138     val contentColor = if (selected) {
139         MaterialTheme.colorScheme.onSecondaryContainer
140     } else {
141         MaterialTheme.colorScheme.onSurfaceVariant
142     }
143     Surface(
144         onClick = onClick,
145         enabled = enabled,
146         modifier = modifier
147             .padding(horizontal = if (compact) 1.dp else 2.dp)
148             .semantics {
149                 role = Role.Tab
150                 this.contentDescription = contentDescription
151             },
152         shape = itemShape,
153         color = containerColor,
154         contentColor = contentColor,
155     ) {
156         Row(
157             modifier = Modifier
158                 .fillMaxWidth()
159                 .padding(horizontal = itemPaddingH, vertical = itemPaddingV),
160             horizontalArrangement = if (selected) Arrangement.Start else Arrangement.Center,
161             verticalAlignment = Alignment.CenterVertically,
162         ) {
163             Icon(
164                 imageVector = icon,
165                 contentDescription = null,
166                 modifier = Modifier.size(iconSize),
167             )
168             if (selected) {
169                 NavigationBarMarqueeText(
170                     text = label,
171                     modifier = Modifier
172                         .weight(1f)
173                         .padding(start = if (compact) 4.dp else 6.dp),
174                     style = labelStyle,
175                     velocity = labelVelocity,

```

```
176                                     )
177                                     }
178                                 }
179                            }
180                        }
181
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/elements/ NavigationBarMarqueeText.kt

```
1 package com.github.nullptroma.wallenc.ui.elements
2
3 import androidx.compose.foundation.ExperimentalFoundationApi
4 import androidx.compose.foundation.basicMarquee
5 import androidx.compose.foundation.layout.fillMaxWidth
6 import androidx.compose.material3.LocalTextStyle
7 import androidx.compose.material3.Text
8 import androidx.compose.runtime.Composable
9 import androidx.compose.ui.Modifier
10 import androidx.compose.ui.text.TextStyle
11 import androidx.compose.ui.text.style.TextAlign
12 import androidx.compose.ui.text.style.TextOverflow
13 import androidx.compose.ui.unit.Dp
14 import androidx.compose.ui.unit.dp
15
16 /**
17  * Однострочная подпись таба: при нехватке ширины текст циклически прокручивается.
18  */
19 @OptIn(ExperimentalFoundationApi::class)
20 @Composable
21 fun NavigationBarMarqueeText(
22     text: String,
23     modifier: Modifier = Modifier,
24     style: TextStyle = LocalTextStyle.current,
25     velocity: Dp = 28.dp,
26 ) {
27     Text(
28         text = text,
29         modifier = modifier
30             .fillMaxWidth()
31             .basicMarquee(
32                 iterations = Int.MAX_VALUE,
33                 repeatDelayMillis = 1_200,
34                 velocity = velocity,
35             ),
36         style = style,
37         maxLines = 1,
38         softWrap = false,
39         overflow = TextOverflow.Clip,
40         textAlign = TextAlign.Center,
41     )
42 }
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/elements/
QrImageDecoder.kt

```
1 package com.github.nullptroma.wallenc.ui.elements
2
3 import androidx.camera.core.ImageProxy
4 import com.google.zxing.BarcodeFormat
5 import com.google.zxing.BinaryBitmap
6 import com.google.zxing.ChecksumException
7 import com.google.zxing.DecodeHintType
8 import com.google.zxing.FormatException
9 import com.google.zxing.NotFoundException
10 import com.google.zxing.PlanarYUVLuminanceSource
11 import com.google.zxing.common.HybridBinarizer
12 import com.google.zxing.qrcode.QRCodeReader
13
14 internal object QrImageDecoder {
15     private val hints = mapOf(
16         DecodeHintType.POSSIBLE_FORMATS to listOf(BarcodeFormat.QR_CODE),
17         DecodeHintType.CHARACTER_SET to "UTF-8",
18     )
19     private val reader = QRCodeReader()
20
21     fun decode(imageProxy: ImageProxy): String? {
22         val yPlane = imageProxy.planes[0]
23         val yBuffer = yPlane.buffer
24         yBuffer.rewind()
25         val yRowStride = yPlane.rowStride
26         val yPixelStride = yPlane.pixelStride
27         val width = imageProxy.width
28         val height = imageProxy.height
29         val yBytes = ByteArray(width * height)
30         var outputOffset = 0
31         if (yPixelStride == 1 && yRowStride == width) {
32             yBuffer.get(yBytes, 0, yBytes.size.coerceAtMost(yBuffer.remaining()))
33         } else {
34             for (row in 0 until height) {
35                 var inputOffset = row * yRowStride
36                 for (col in 0 until width) {
37                     yBytes[outputOffset++] = yBuffer.get(inputOffset)
38                     inputOffset += yPixelStride
39                 }
40             }
41         }
42         val (luminance, decodeWidth, decodeHeight) = orientLuminance(
```

```

43         yBytes,
44         width,
45         height,
46         imageProxy.imageInfo.rotationDegrees,
47     )
48     val source = PlanarYUVLuminanceSource(
49         luminance,
50         decodeWidth,
51         decodeHeight,
52         0,
53         0,
54         decodeWidth,
55         decodeHeight,
56         false,
57     )
58     return try {
59         reader.decode(BinaryBitmap(HybridBinarizer(source)), hints).text
60     } catch (_: NotFoundException) {
61         null
62     } catch (_: ChecksumException) {
63         null
64     } catch (_: FormatException) {
65         null
66     }
67 }
68
69 private fun orientLuminance(
70     yBytes: ByteArray,
71     width: Int,
72     height: Int,
73     rotationDegrees: Int,
74 ): Triple<ByteArray, Int, Int> = when (rotationDegrees) {
75     90 -> Triple(rotateY90(yBytes, width, height, clockwise = true), height, width)
76     180 -> Triple(rotateY180(yBytes, width, height), width, height)
77     270 -> Triple(rotateY90(yBytes, width, height, clockwise = false), height, width)
78     else -> Triple(yBytes, width, height)
79 }
80
81 private fun rotateY90(data: ByteArray, width: Int, height: Int, clockwise: Boolean):
82     ByteArray {
83     val output = ByteArray(data.size)
84     for (y in 0 until height) {
85         for (x in 0 until width) {
86             val srcIndex = y * width + x
87             val dstIndex = if (clockwise) {

```

```

87         x * height + (height - 1 - y)
88     } else {
89         (width - 1 - x) * height + y
90     }
91     output[dstIndex] = data[srcIndex]
92 }
93 }
94 return output
95 }
96
97 private fun rotateY180(data: ByteArray, width: Int, height: Int): ByteArray {
98     val output = ByteArray(data.size)
99     for (y in 0 until height) {
100         for (x in 0 until width) {
101             output[(height - 1 - y) * width + (width - 1 - x)] = data[y * width + x]
102         }
103     }
104     return output
105 }
106 }
107

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/elements/ QrScannerDialog.kt

```
1 package com.github.nullptroma.wallenc.ui.elements
2
3 import android.view.ViewGroup
4 import androidx.camera.core.CameraSelector
5 import androidx.camera.core.ImageAnalysis
6 import androidx.camera.core.Preview
7 import androidx.camera.lifecycle.ProcessCameraProvider
8 import androidx.camera.view.PreviewView
9 import androidx.compose.foundation.layout.Arrangement
10 import androidx.compose.foundation.layout.Column
11 import androidx.compose.foundation.layout.Spacer
12 import androidx.compose.foundation.layout.fillMaxWidth
13 import androidx.compose.foundation.layout.height
14 import androidx.compose.foundation.layout.padding
15 import androidx.compose.material3.BasicAlertDialog
16 import androidx.compose.material3.Button
17 import androidx.compose.material3.Card
18 import androidx.compose.material3.ExperimentalMaterial3Api
19 import androidx.compose.material3.MaterialTheme
20 import androidx.compose.material3.Text
21 import androidx.compose.runtime.Composable
22 import androidx.compose.runtime.DisposableEffect
23 import androidx.compose.runtime.LaunchedEffect
24 import androidx.compose.runtime.getValue
25 import androidx.compose.runtime.mutableStateOf
26 import androidx.compose.runtime.remember
27 import androidx.compose.runtime.setValue
28 import androidx.compose.ui.Modifier
29 import androidx.compose.ui.platform.LocalContext
30 import androidx.compose.ui.res.stringResource
31 import androidx.compose.ui.unit.dp
32 import androidx.compose.ui.viewinterop.AndroidView
33 import com.github.nullptroma.wallenc.ui.R
34 import java.util.concurrent.Executors
35 import java.util.concurrent.atomic.AtomicBoolean
36
37 @OptIn(ExperimentalMaterial3Api::class)
38 @Composable
39 fun QrScannerDialog(
40     onDismiss: () -> Unit,
41     onScanned: (String) -> Unit,
42 ) {
```

```

43     val context = LocalContext.current
44     val lifecycleOwner = androidx.lifecycle.compose.LocalLifecycleOwner.current
45     val cameraProviderFuture = remember { ProcessCameraProvider.getInstance(context) }
46     val cameraExecutor = remember { Executors.newSingleThreadExecutor() }
47     val consumed = remember { AtomicBoolean(false) }
48     var previewViewRef by remember { mutableStateOf<PreviewView?>(null) }
49
50     DisposableEffect(Unit) {
51         onDispose {
52             runCatching { cameraProviderFuture.get().unbindAll() }
53             cameraExecutor.shutdown()
54         }
55     }
56
57     LaunchedEffect(previewViewRef) {
58         val previewView = previewViewRef ?: return@LaunchedEffect
59         val cameraProvider = cameraProviderFuture.get()
60         val preview = Preview.Builder().build().also {
61             it.surfaceProvider = previewView.surfaceProvider
62         }
63         val analysis = ImageAnalysis.Builder()
64             .setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST)
65             .build()
66             .also { imageAnalysis ->
67                 imageAnalysis.targetRotation = previewView.display.rotation
68                 imageAnalysis.setAnalyzer(cameraExecutor) { imageProxy ->
69                     try {
70                         if (consumed.get()) return@setAnalyzer
71                         val raw = QrImageDecoder.decode(imageProxy)?.trim().orEmpty()
72                         if (raw.isNotBlank() && consumed.compareAndSet(false, true)) {
73                             onScanned(raw)
74                         }
75                     } finally {
76                         imageProxy.close()
77                     }
78                 }
79             }
80         cameraProvider.unbindAll()
81         cameraProvider.bindToLifecycle(
82             lifecycleOwner,
83             CameraSelector.DEFAULT_BACK_CAMERA,
84             preview,
85             analysis,
86         )
87     }

```

```

88
89     BasicAlertDialog(onDismissRequest = onDismiss) {
90         Card {
91             Column(
92                 modifier = Modifier.padding(12.dp),
93                 verticalArrangement = Arrangement.spacedBy(10.dp),
94             ) {
95                 Text(
96                     text = stringResource(R.string.two_fa_scan_qr_title),
97                     style = MaterialTheme.typography.titleLarge,
98                 )
99                 AndroidView(
100                     modifier = Modifier
101                         .fillMaxWidth()
102                         .height(380.dp),
103                     factory = { ctx ->
104                         PreviewView(ctx).apply {
105                             layoutParams = ViewGroup.LayoutParams(
106                                 ViewGroup.LayoutParams.MATCH_PARENT,
107                                 ViewGroup.LayoutParams.MATCH_PARENT,
108                             )
109                             previewViewRef = this
110                         }
111                     },
112                 )
113                 Spacer(modifier = Modifier.height(4.dp))
114                 Button(
115                     modifier = Modifier.fillMaxWidth(),
116                     onClick = onDismiss,
117                 ) {
118                     Text(text = stringResource(R.string.cancel))
119                 }
120             }
121         }
122     }
123 }
124

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/elements/ StorageTree.kt

```
1 package com.github.nullptroma.wallenc.ui.elements
2
3 import androidx.compose.foundation.background
4 import androidx.compose.foundation.clickable
5 import androidx.compose.foundation.interaction.MutableInteractionSource
6 import androidx.compose.foundation.layout.Box
7 import androidx.compose.foundation.layout.Column
8 import androidx.compose.foundation.layout.Row
9 import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.offset
11 import androidx.compose.foundation.layout.padding
12 import androidx.compose.foundation.layout.widthIn
13 import androidx.compose.foundation.layout.wrapContentHeight
14 import androidx.compose.material.icons.Icons
15 import androidx.compose.material.icons.filled.Lock
16 import androidx.compose.material.icons.filled.LockOpen
17 import androidx.compose.material.icons.filled.MoreVert
18 import androidx.compose.foundation.layout.size
19 import androidx.compose.material3.Card
20 import androidx.compose.material3.CardDefaults
21 import androidx.compose.material3.CircularProgressIndicator
22 import androidx.compose.material3.DropdownMenu
23 import androidx.compose.material3.DropdownMenuItem
24 import androidx.compose.material3.HorizontalDivider
25 import androidx.compose.material3.Icon
26 import androidx.compose.material3.IconButton
27 import androidx.compose.material3.LocalTextStyle
28 import androidx.compose.material3.MaterialTheme
29 import androidx.compose.material3.Text
30 import androidx.compose.material3.ripple
31 import androidx.compose.runtime.Composable
32 import androidx.compose.runtime.LaunchedEffect
33 import androidx.compose.runtime.getValue
34 import androidx.compose.runtime.mutableStateOf
35 import androidx.compose.runtime.remember
36 import androidx.compose.runtime.setValue
37 import androidx.compose.ui.Alignment
38 import androidx.compose.ui.Modifier
39 import androidx.compose.ui.draw.clip
40 import androidx.compose.ui.res.stringResource
41 import androidx.compose.ui.text.PlatformTextStyle
42 import androidx.compose.ui.text.style.TextAlign
```

```

43 import androidx.compose.ui.unit.dp
44 import androidx.compose.ui.unit.sp
45 import androidx.compose.ui.zIndex
46 import androidx.lifecycle.compose.collectAsStateWithLifecycle
47 import com.github.nullptroma.wallenc.domain.datatypes.StorageMetaLoadState
48 import com.github.nullptroma.wallenc.domain.datatypes.Tree
49 import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
50 import com.github.nullptroma.wallenc.ui.R
51 import com.github.nullptroma.wallenc.ui.utils.debounceLambda
52 import java.util.UUID
53
54 @Composable
55 fun StorageTree(
56     modifier: Modifier,
57     tree: Tree<IStorageInfo>,
58     isUuidBusy: (UUID) -> Boolean,
59     onClick: (Tree<IStorageInfo>) -> Unit,
60     onRename: (Tree<IStorageInfo>, String) -> Unit,
61     onRemove: (Tree<IStorageInfo>) -> Unit,
62     onEncrypt: (Tree<IStorageInfo>, String, Boolean, Boolean) -> Unit,
63     onOpenEncrypted: (Tree<IStorageInfo>, String, Boolean) -> Unit,
64     onCloseEncrypted: (Tree<IStorageInfo>) -> Unit,
65     onDisableEncryption: (Tree<IStorageInfo>) -> Unit,
66     getStatusTextRes: (Tree<IStorageInfo>) -> Int,
67     isEncryptionOpened: (Tree<IStorageInfo>) -> Boolean,
68     isStorageSyncLockHeld: suspend (IStorageInfo) -> Boolean,
69     onClearStorageSyncLock: (IStorageInfo) -> Unit,
70 ) {
71     val cur = tree.value
72     val rowBusy = isUuidBusy(cur.uuid)
73     val numOfFiles by cur.numberOfFiles.collectAsStateWithLifecycle()
74     val size by cur.size.collectAsStateWithLifecycle()
75     val metaInfo by cur.metaInfo.collectAsStateWithLifecycle()
76     val metaLoadState by cur.metaLoadState.collectAsStateWithLifecycle()
77     val isAvailable by cur.isAvailable.collectAsStateWithLifecycle()
78     val metaUnavailable = metaLoadState == StorageMetaLoadState.Unavailable
79     val rowEnabled = isAvailable && !rowBusy && !metaUnavailable
80     val isEncrypted = metaInfo.encInfo != null
81     val isOpened = isEncryptionOpened(tree)
82     val borderColor =
83         if (cur.isVirtualStorage) MaterialTheme.colorScheme.secondary else
84         MaterialTheme.colorScheme.primary
85     val yesWord = stringResource(R.string.storage_value_yes)
86     val noWord = stringResource(R.string.storage_value_no)
87     val unavailableHint = stringResource(R.string.storage_unavailable_hint)

```



```

87     val metaUnavailableHint = stringResource(R.string.storage_meta_unavailable_hint)
88     Column(modifier) {
89         Box(
90             modifier = Modifier
91                 .fillMaxWidth()
92                 .wrapContentHeight()
93                 .zIndex(100f),
94         ) {
95             val interactionSource = remember { MutableInteractionSource() }
96             Box(
97                 modifier = Modifier
98                     .matchParentSize()
99                     .padding(end = 16.dp)
100                     .clip(CardDefaults.shape)
101                     .background(borderColor)
102                     .clickable(
103                         interactionSource = interactionSource,
104                         indication = ripple(),
105                         enabled = false,
106                         onClick = { },
107                     ),
108             )
109             Card(
110                 interactionSource = interactionSource,
111                 modifier = Modifier
112                     .padding(start = 8.dp)
113                     .fillMaxWidth()
114                     .wrapContentHeight(),
115                 elevation = CardDefaults.cardElevation(
116                     defaultElevation = 4.dp,
117                 ),
118                 enabled = rowEnabled,
119                 onClick = debouncedLambda(debounceMs = 500) {
120                     if (rowEnabled) {
121                         onClick(tree)
122                     }
123                 },
124             ) {
125                 Row(
126                     modifier = Modifier
127                         .fillMaxWidth()
128                         .wrapContentHeight(),
129                 ) {
130                     Column(

```

```

131         modifier = Modifier
132             .weight(1f)
133             .padding(8.dp),
134     ) {
135         Text(metaInfo.name ?: stringResource(R.string.no_name))
136         Text(
137             text = stringResource(
138                 R.string.storage_field_available,
139                 if (isAvailable) yesWord else noWord,
140             ),
141             style = MaterialTheme.typography.bodySmall,
142         )
143         Text(
144             text = stringResource(
145                 R.string.storage_field_files,
146                 numOfFiles?.toString() ?: "-",
147             ),
148             style = MaterialTheme.typography.bodySmall,
149         )
150         Text(
151             text = stringResource(
152                 R.string.storage_field_size,
153                 size?.toString() ?: "-",
154             ),
155             style = MaterialTheme.typography.bodySmall,
156         )
157         Text(
158             text = stringResource(
159                 R.string.storage_field_virtual,
160                 if (cur.isVirtualStorage) yesWord else noWord,
161             ),
162             style = MaterialTheme.typography.bodySmall,
163         )
164         if (metaUnavailable) {
165             Text(
166                 text = metaUnavailableHint,
167                 style = MaterialTheme.typography.bodySmall,
168                 color = MaterialTheme.colorScheme.error,
169             )
170         } else if (!isAvailable) {
171             Text(
172                 text = unavailableHint,
173                 style = MaterialTheme.typography.bodySmall,
174                 color = MaterialTheme.colorScheme.error,
175             )
176         }
177     }

```

```

176         }
177     }
178     Column(
179         modifier = Modifier
180             .widthIn(min = 112.dp)
181             .padding(end = 4.dp),
182         horizontalAlignment = Alignment.End,
183     ) {
184         var expanded by remember { mutableStateOf(false) }
185         var syncLockHeld by remember { mutableStateOf<Boolean?>(null) }
186         LaunchedEffect(expanded, cur.uuid) {
187             if (expanded) {
188                 syncLockHeld = null
189                 syncLockHeld = isStorageSyncLockHeld(cur)
190             } else {
191                 syncLockHeld = null
192             }
193         }
194         var showRenameDialog by remember { mutableStateOf(false) }
195         var showRemoveConfirmDialog by remember { mutableStateOf(false) }
196         var showLockDialog by remember { mutableStateOf(false) }
197         var showSetupEncryptionDialog by remember { mutableStateOf(false) }
198         var showOpenEncryptionDialog by remember { mutableStateOf(false) }
199         Box(modifier = Modifier.padding(0.dp, 0.dp, 0.dp, 0.dp)) {
200             Row(
201                 verticalAlignment = Alignment.CenterVertically,
202             ) {
203                 if (rowBusy) {
204                     CircularProgressIndicator(
205                         modifier = Modifier
206                             .padding(end = 4.dp)
207                             .size(22.dp),
208                         strokeWidth = 2.dp,
209                     )
210                 }
211                 IconButton(
212                     onClick = { expanded = !expanded },
213                     enabled = rowEnabled,
214                 ) {
215                     Icon(
216                         Icons.Default.MoreVert,
217                         contentDescription = stringResource(
218                             if (rowBusy) {
219                                 R.string.storage_row_task_running_cd

```

```

220         } else {
221             R.string.show_storage_item_menu
222         },
223     ),
224 )
225 }
226 }
227 DropdownMenu(
228     expanded = expanded,
229     onDismissRequest = { expanded = false },
230 ) {
231     DropdownMenuItem(
232         enabled = rowEnabled,
233         onClick = {
234             expanded = false
235             if (rowEnabled) showRenameDialog = true
236         },
237         text = {
238             Text(
239                 when {
240                     !isAvailable -> stringResource(
241                         R.string.storage_menu_unavailable,
242                         stringResource(R.string.rename),
243                     )
244                     rowBusy ->
stringResource(R.string.storage_menu_busy, stringResource(R.string.rename))
245                     else -> stringResource(R.string.rename)
246                 },
247             )
248         },
249     )
250     HorizontalDivider()
251     DropdownMenuItem(
252         enabled = rowEnabled,
253         onClick = {
254             expanded = false
255             if (rowEnabled) showRemoveConfirmDialog = true
256         },
257         text = {
258             Text(
259                 when {
260                     !isAvailable -> stringResource(
261                         R.string.storage_menu_unavailable,
262                         stringResource(R.string.remove),
263                     )

```

```

264         rowBusy ->
stringResource(R.string.storage_menu_busy, stringResource(R.string.remove))
265     else -> stringResource(R.string.remove)
266 },
267 )
268 },
269 )
270 if (!isEncrypted) {
271     HorizontalDivider()
272     DropdownMenuItem(
273         enabled = rowEnabled,
274         onClick = {
275             expanded = false
276             if (rowEnabled) showSetupEncryptionDialog = true
277         },
278         text = {
279             Text(
280                 when {
281                     !isAvailable -> stringResource(
282                         R.string.storage_menu_unavailable,
283                         stringResource(R.string.encrypt),
284                     )
285                     rowBusy ->
stringResource(R.string.storage_menu_busy, stringResource(R.string.encrypt))
286                     else -> stringResource(R.string.encrypt)
287                 },
288             )
289         },
290     )
291 }
292 HorizontalDivider()
293 DropdownMenuItem(
294     enabled = syncLockHeld == true && !rowBusy,
295     onClick = {
296         expanded = false
297         if (syncLockHeld == true && !rowBusy) {
298             onClearStorageSyncLock(cur)
299         }
300     },
301     text = {
302         Text(
303             when (syncLockHeld) {
304                 null ->
stringResource(R.string.storage_sync_lock_checking)
305                 true ->
stringResource(R.string.storage_sync_unlock_action)

```

```

306         stringResource(R.string.storage_sync_not_locked) false ->
307     },
308 )
309 },
310 )
311 }
312
313 if (showRenameDialog) {
314     TextEditCancelOkDialog(
315         onDismiss = { showRenameDialog = false },
316         onConfirmation = { newName ->
317             showRenameDialog = false
318             onRename(tree, newName)
319         },
320         title = stringResource(R.string.new_name_title),
321         startString = metaInfo.name ?: "",
322     )
323 }
324
325 if (showRemoveConfirmDialog) {
326     ConfirmationCancelOkDialog(
327         onDismiss = { showRemoveConfirmDialog = false },
328         title = stringResource(
329             R.string.remove_confirmation_dialog,
330             metaInfo.name ?: stringResource(R.string.no_name),
331         ),
332         onConfirmation = {
333             showRemoveConfirmDialog = false
334             onRemove(tree)
335         },
336     )
337 }
338
339 if (showLockDialog) {
340     StorageEncryptionActionsDialog(
341         onDismiss = { showLockDialog = false },
342         title = metaInfo.name ?:
stringResource(R.string.no_name),
343         isOpened = isOpened,
344         onOpen = {
345             showLockDialog = false
346             showOpenEncryptionDialog = true
347         },
348         onClose = {
349             showLockDialog = false

```

```

350         onCloseEncrypted(tree)
351     },
352     onDisable = {
353         showLockDialog = false
354         onDisableEncryption(tree)
355     },
356 )
357 }
358
359 if (showSetupEncryptionDialog) {
360     EncryptionSetupDialog(
361         onDismiss = { showSetupEncryptionDialog = false },
362         onConfirmation = { password, encryptPath,
rememberPassword ->
363             showSetupEncryptionDialog = false
364             onEncrypt(tree, password, encryptPath,
rememberPassword)
365         },
366     )
367 }
368
369 if (showOpenEncryptionDialog) {
370     OpenEncryptedStorageDialog(
371         onDismiss = { showOpenEncryptionDialog = false },
372         onConfirmation = { password, rememberPassword ->
373             showOpenEncryptionDialog = false
374             onOpenEncrypted(tree, password, rememberPassword)
375         },
376     )
377 }
378 }
379 if (isEncrypted) {
380     IconButton(
381         onClick = { showLockDialog = true },
382         enabled = rowEnabled,
383     ) {
384         Icon(
385             Icons.Default.Lock,
386             contentDescription =
stringResource(R.string.storage_lock_actions),
387         )
388     }
389 }
390 Text(
391     modifier = Modifier

```

```

392         .padding(top = 4.dp, end = 8.dp),
393         text = stringResource(getStatusTextRes(tree)),
394         textAlign = TextAlign.End,
395         fontSize = 11.sp,
396     )
397     Text(
398         modifier = Modifier
399             .padding(end = 8.dp, bottom = 8.dp),
400         text = cur.uuid.toString(),
401         textAlign = TextAlign.End,
402         fontSize = 8.sp,
403         style = LocalTextStyle.current.copy(
404             platformStyle = PlatformTextStyle(
405                 includeFontPadding = true,
406             ),
407         ),
408     )
409 }
410 }
411 }
412 }
413 for (i in tree.children ?: listOf()) {
414     StorageTree(
415         Modifier
416             .padding(16.dp, 0.dp, 0.dp, 0.dp)
417             .offset(y = (-4).dp),
418         tree = i,
419         isUuidBusy = isUuidBusy,
420         onClick = onClick,
421         onRename,
422         onRemove,
423         onEncrypt,
424         onOpenEncrypted,
425         onCloseEncrypted,
426         onDisableEncryption,
427         getStatusTextRes,
428         isEncryptionOpened,
429         isStorageSyncLockHeld,
430         onClearStorageSyncLock,
431     )
432 }
433 }
434 }
435

```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/elements/
WallencScreenScaffold.kt

```
1 package com.github.nullptroma.wallenc.ui.elements
2
3 import androidx.compose.foundation.layout.Box
4 import androidx.compose.foundation.layout.PaddingValues
5 import androidx.compose.foundation.layout.WindowInsets
6 import androidx.compose.foundation.layout.padding
7 import androidx.compose.material3.FabPosition
8 import androidx.compose.material3.Scaffold
9 import androidx.compose.material3.SnackbarHost
10 import androidx.compose.material3.SnackbarHostState
11 import androidx.compose.runtime.Composable
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.unit.dp
14
15 @Composable
16 fun WallencScreenScaffold(
17     modifier: Modifier = Modifier,
18     snackbarHostState: SnackbarHostState? = null,
19     floatingActionButton: @Composable () -> Unit = {},
20     content: @Composable (PaddingValues) -> Unit,
21 ) {
22     Scaffold(
23         modifier = modifier,
24         contentWindowInsets = WindowInsets(0.dp),
25         snackbarHost = {
26             if (snackbarHostState != null) {
27                 SnackbarHost(snackbarHostState)
28             }
29         },
30         floatingActionButton = floatingActionButton,
31         floatingActionButtonPosition = FabPosition.End,
32         content = content,
33     )
34 }
35
36 @Composable
37 fun WallencScreenContentPadding(
38     innerPadding: PaddingValues,
39     modifier: Modifier = Modifier,
40     content: @Composable () -> Unit,
41 ) {
42     Box(
```

```
43         modifier = modifier
44         .padding(innerPadding)
45         .padding(horizontal = 16.dp, vertical = 12.dp),
46     ) {
47         content()
48     }
49 }
50
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/extensions/
StorageInfo.kt

```
1 package com.github.nullptroma.wallenc.ui.extensions
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
4
5 fun IStorageInfo.toPrintable(): String {
6     return "{ uuid: $uuid, enc: ${metaInfo.value.encInfo} }"
7 }
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/locale/
AppLocaleController.kt

```
1 package com.github.nullptroma.wallenc.ui.locale
2
3 import kotlinx.coroutines.flow.Flow
4
5 enum class AppLanguage {
6     System,
7     English,
8     Russian,
9 }
10
11 interface AppLocaleController {
12     val language: Flow<AppLanguage>
13
14     suspend fun setLanguage(language: AppLanguage)
15
16     /** Applies persisted locale; call from [android.app.Application.onCreate]. */
17     suspend fun applyStoredLocale()
18 }
19
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/navigation/
MainNavRoutes.kt

```
1 package com.github.nullptroma.wallenc.ui.navigation
2
3 import com.github.nullptroma.wallenc.ui.screens.main.screens.remotes.RemoteVaultsRoute
4 import com.github.nullptroma.wallenc.ui.screens.main.screens.vault.LocalVaultRoute
5 import com.github.nullptroma.wallenc.ui.screens.shared.TextEditRoute
6
7 private val mainTopLevelRoutePrefixes: Set<String> = setOf(
8     LocalVaultRoute::class.qualifiedName!!,
9     RemoteVaultsRoute::class.qualifiedName!!,
10 )
11
12 fun isMainTopLevelRoute(route: String?): Boolean {
13     if (route == null) return true
14     return mainTopLevelRoutePrefixes.any { route.startsWith(it) }
15 }
16
17 fun isTextEditDestination(route: String?): Boolean {
18     val qualified = TextEditRoute::class.qualifiedName ?: return false
19     return route?.startsWith(qualified) == true
20 }
21
22 fun shouldShowMainFloatingBack(route: String?): Boolean {
23     if (route == null) return false
24     return !isMainTopLevelRoute(route)
25 }
26
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/navigation/
NavBarItemData.kt

```
1 package com.github.nullptroma.wallenc.ui.navigation
2
3 import androidx.compose.ui.graphics.vector.ImageVector
4
5 data class NavBarItemData(
6     val nameStringResourceId: Int,
7     val screenRouteClass: String,
8     val icon: ImageVector?,
9     val iconContentDescriptionResourceId: Int = nameStringResourceId,
10 )
11
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/navigation/ NavigationState.kt

```
1 package com.github.nullptroma.wallenc.ui.navigation
2
3 import androidx.compose.runtime.Composable
4 import androidx.compose.runtime.remember
5 import androidx.navigation.NavGraph.Companion.findStartDestination
6 import androidx.navigation.NavHostController
7 import androidx.navigation.compose.rememberNavController
8 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
9
10
11 class NavigationState(
12     val navHostController: NavHostController
13 ) {
14     fun changeTop(route: ScreenRoute) {
15         navHostController.navigate(route) {
16             popUpTo(navHostController.graph.findStartDestination().id)
17             launchSingleTop = true
18             restoreState = true
19         }
20     }
21
22     fun push(route: ScreenRoute) {
23         navHostController.navigate(route) {
24             restoreState = true
25         }
26     }
27
28     fun pop(): Boolean = navHostController.popBackStack()
29
30     fun canPop(): Boolean = navHostController.previousBackStackEntry != null
31 }
32
33 @Composable
34 fun rememberNavigationState(
35     navHostController: NavHostController? = null
36 ): NavigationState {
37     val controller = navHostController ?: rememberNavController()
38     return remember(controller) { NavigationState(controller) }
39 }
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/navigation/ WallencDeepLinks.kt

```
1 package com.github.nullptroma.wallenc.ui.navigation
2
3 import android.content.Intent
4
5 /**
6  * URI для входа извне приложения (уведомления, виджеты, шорткаты, другие Activity).
7  *
8  * Должны совпадать с `<data android:scheme=... android:host=.../></intent-
9  filter>` в манифесте
10  * и с [androidx.navigation.navDeepLink] на соответствующих `composable` в
11  [androidx.navigation.compose.NavHost].
12  */
13 object WallencDeepLinks {
14     const val SCHEME = "wallenc"
15
16     object Host {
17         const val MAIN = "main"
18         const val TASKS = "tasks"
19         const val SYNC = "sync"
20         const val SETTINGS = "settings"
21     }
22
23     const val MAIN_URI_PATTERN = "$SCHEME://${Host.MAIN}"
24     const val TASKS_URI_PATTERN = "$SCHEME://${Host.TASKS}"
25     const val SYNC_URI_PATTERN = "$SCHEME://${Host.SYNC}"
26     const val SETTINGS_URI_PATTERN = "$SCHEME://${Host.SETTINGS}"
27 }
28
29 /** `ACTION_VIEW` с URI нашей схемы — обрабатывается
30 [androidx.navigation.NavController.handleDeepLink]. */
31 fun Intent.matchesWallencDeepLink(): Boolean =
32     action == Intent.ACTION_VIEW && data?.scheme == WallencDeepLinks.SCHEME
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/ AppStrings.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.ui.R
5 import com.github.nullptroma.wallenc.usecases.AddStorageToSyncGroupResult
6 import com.github.nullptroma.wallenc.vault.contract.VaultLinkFailure
7
8 fun WallencException.toUserNotification(): UserNotification.TextRes = when (this) {
9     is WallencException.Feature.StorageNotFound ->
10         UserNotification.TextRes(R.string.error_storage_not_found)
11     is WallencException.Feature.NeedsDecryptedView ->
12         UserNotification.TextRes(R.string.error_storage_locked_view)
13     is WallencException.Feature.SecretNotFound ->
14         UserNotification.TextRes(R.string.error_secret_not_found)
15     is WallencException.Feature.StorageNotWritable ->
16         UserNotification.TextRes(R.string.error_storage_not_writable)
17
18     is WallencException.Storage.NotAvailable,
19     is WallencException.Storage.NotWritable,
20     ->
21         UserNotification.TextRes(R.string.error_storage_not_writable)
22     is WallencException.Storage.FileNotFound ->
23         UserNotification.TextRes(R.string.error_file_not_found)
24     is WallencException.Storage.IncorrectKey ->
25         UserNotification.TextRes(R.string.error_incorrect_password)
26     is WallencException.Storage.NotEncrypted ->
27         UserNotification.TextRes(R.string.error_storage_not_encrypted)
28     is WallencException.Storage.EncInfoMissing ->
29         UserNotification.TextRes(R.string.error_enc_info_missing)
30     is WallencException.Storage.DeleteRootForbidden ->
31         UserNotification.TextRes(R.string.error_delete_root_forbidden)
32     is WallencException.Storage.NotAFile ->
33         UserNotification.TextRes(R.string.error_not_a_file)
34     is WallencException.Storage.NotADirectory ->
35         UserNotification.TextRes(R.string.error_not_a_directory)
36     is WallencException.Storage.PathIsFile ->
37         UserNotification.TextRes(R.string.error_path_is_file)
38     is WallencException.Storage.CannotWriteOverDirectory ->
39         UserNotification.TextRes(R.string.error_cannot_write_over_directory)
40     is WallencException.Storage.UnexpectedState ->
41         UserNotification.TextRes(R.string.error_unexpected_state)
42     is WallencException.Storage.IoFailed ->
```

```

43         UserNotification.TextRes(R.string.error_network)
44
45         is WallencException.Auth.Failed,
46         is WallencException.Auth.TokenMissing,
47         ->
48             UserNotification.TextRes(R.string.vault_link_error_auth)
49         is WallencException.Network.ResourceLocked ->
50             UserNotification.TextRes(R.string.error_disk_resource_locked)
51         is WallencException.Network.OperationFailed,
52         is WallencException.Network.OperationTimedOut,
53         is WallencException.Network.HttpFailed,
54         is WallencException.Network.IoFailed,
55         ->
56             UserNotification.TextRes(R.string.error_network)
57
58         is WallencException.Unknown ->
59             UserNotification.TextRes(R.string.error_unknown)
60     }
61
62     fun VaultLinkFailure.toUserNotification(): UserNotification.TextRes = when (this) {
63         VaultLinkFailure.UnsupportedBrand ->
64             UserNotification.TextRes(R.string.vault_link_error_unsupported_brand)
65         VaultLinkFailure.NotRegistered ->
66             UserNotification.TextRes(R.string.vault_link_error_not_registered)
67         VaultLinkFailure.AuthError ->
68             UserNotification.TextRes(R.string.vault_link_error_auth)
69         VaultLinkFailure.Unknown ->
70             UserNotification.TextRes(R.string.vault_link_error_unknown)
71     }
72
73     fun AddStorageToSyncGroupResult.toUserNotification(groupId: String):
74     UserNotification.TextRes = when (this) {
75         AddStorageToSyncGroupResult.Added ->
76             UserNotification.TextRes(R.string.sync_msg_storage_added, listOf(groupId))
77         AddStorageToSyncGroupResult.GroupNotFound ->
78             UserNotification.TextRes(R.string.sync_error_group_not_found)
79         AddStorageToSyncGroupResult.AlreadyInGroup ->
80             UserNotification.TextRes(R.string.sync_msg_storage_already_added)
81         AddStorageToSyncGroupResult.EncryptedStorageNotAllowed ->
82             UserNotification.TextRes(R.string.sync_msg_only_plain_storage_allowed)
83         AddStorageToSyncGroupResult.MissingEncryptionSecret ->
84             UserNotification.TextRes(R.string.sync_msg_storage_encryption_key_required)
85         AddStorageToSyncGroupResult.IncompatibleEncryption ->
86             UserNotification.TextRes(R.string.sync_msg_storage_incompatible_encryption)
87     }

```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/ TaskLogKeys.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import com.github.nullptroma.wallenc.domain.tasks.StorageSyncTriggerReason
4 import com.github.nullptroma.wallenc.domain.tasks.TaskLogKey
5 import com.github.nullptroma.wallenc.ui.R
6
7 fun TaskLogKey.resolve(resolver: UiStringResolver): String = when (this) {
8     is TaskLogKey.SyncStarted -> resolver(
9         R.string.task_log_sync_started,
10        resolver.resolveSyncTriggerReason(reason),
11    )
12    is TaskLogKey.SyncFinished -> resolver(
13        R.string.task_log_sync_finished,
14        resolver.resolveSyncTriggerReason(reason),
15    )
16    is TaskLogKey.SyncFailed -> {
17        val notification = error.toUserNotification().resolve(resolver)
18        resolver(
19            R.string.task_log_sync_failed,
20            resolver.resolveSyncTriggerReason(reason),
21            notification,
22        )
23    }
24 }
25
26 fun UiStringResolver.storageSyncTaskTitle(reason: StorageSyncTriggerReason): String =
27     this(R.string.task_title_storage_sync, resolveSyncTriggerReason(reason))
28
29 private fun UiStringResolver.resolveSyncTriggerReason(reason: StorageSyncTriggerReason):
30 String =
31     when (reason) {
32         StorageSyncTriggerReason.Debounce -> this(R.string.task_sync_trigger_debounce)
33         StorageSyncTriggerReason.SyncTab -> this(R.string.task_sync_trigger_sync_tab)
34         StorageSyncTriggerReason.Background -> this(R.string.task_sync_trigger_background)
35     }
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/
TaskLogKeysCompose.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import androidx.compose.runtime.Composable
4 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLine
5
6 @Composable
7 fun TaskLogLine.displayText(): String {
8     val key = logKey
9     if (key != null) {
10         return key.resolve(composableUiStringResolver())
11     }
12     return message
13 }
14
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/
TaskPipelineTimeFormat.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import java.time.Instant
4 import java.time.ZoneId
5 import java.time.format.DateTimeFormatter
6
7 private val pipelineTimeFormatter: DateTimeFormatter =
8     DateTimeFormatter.ofPattern("HH:mm:ss")
9
10 fun formatTaskPipelineTime(timestampMs: Long): String =
11     Instant.ofEpochMilli(timestampMs)
12         .atZone(ZoneId.systemDefault())
13         .format(pipelineTimeFormatter)
14
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/ TaskProgressLabels.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
4 import com.github.nullptroma.wallenc.domain.tasks.VaultTaskStep
5 import com.github.nullptroma.wallenc.ui.R
6
7 fun TaskProgressLabel.resolve(resolver: UiStringResolver): String = when (this) {
8     TaskProgressLabel.SyncNoGroups -> resolver(R.string.sync_progress_no_groups)
9     TaskProgressLabel.SyncStarted -> resolver(R.string.sync_progress_started)
10    TaskProgressLabel.SyncCompleted -> resolver(R.string.sync_progress_completed)
11    is TaskProgressLabel.SyncPreparing ->
12        resolver.plurals(R.plurals.sync_progress_preparing, groupCount, groupCount)
13
14    is TaskProgressLabel.SyncGroupPreparing ->
15        resolver(R.string.sync_progress_group_preparing, groupId)
16    is TaskProgressLabel.SyncGroupNotFound ->
17        resolver(R.string.sync_progress_group_not_found, groupId)
18    is TaskProgressLabel.SyncGroupSkippedTooFewStorages ->
19        resolver(R.string.sync_progress_group_skipped_few_storages, groupId)
20    is TaskProgressLabel.SyncGroupSkippedIncompatibleEncryption ->
21        resolver(R.string.sync_progress_group_skipped_incompatible, groupId, count)
22    is TaskProgressLabel.SyncGroupAcquiringLocks ->
23        resolver(R.string.sync_progress_group_acquiring_locks, groupId)
24    is TaskProgressLabel.SyncGroupLockProgress ->
25        resolver(R.string.sync_progress_group_lock, groupId, current, total)
26    is TaskProgressLabel.SyncGroupLockFailed ->
27        resolver(R.string.sync_progress_group_lock_failed, groupId)
28    is TaskProgressLabel.SyncGroupReadingJournals ->
29        resolver(R.string.sync_progress_group_reading_journals, groupId)
30    is TaskProgressLabel.SyncGroupCancelled ->
31        resolver(R.string.sync_progress_group_cancelled, groupId)
32    is TaskProgressLabel.SyncGroupJournalProgress ->
33        resolver(R.string.sync_progress_group_journal, groupId, current, total)
34    is TaskProgressLabel.SyncGroupNoJournalEntries ->
35        resolver(R.string.sync_progress_group_no_entries, groupId)
36    is TaskProgressLabel.SyncGroupProcessingEntries ->
37        resolver.plurals(R.plurals.sync_progress_group_processing, count, groupId, count)
38    is TaskProgressLabel.SyncGroupEntryProgress ->
39        resolver(R.string.sync_progress_group_entry, groupId, current, total)
40    is TaskProgressLabel.SyncGroupCompleted ->
41        resolver(R.string.sync_progress_group_completed, groupId)
42    is TaskProgressLabel.SyncGroupEntriesFailed ->
```

```

41         resolver.plurals(R.plurals.sync_progress_group_entries_failed, failedCount, groupId,
failedCount)
42         is TaskProgressLabel.SyncGroupRenewingLocks ->
43             resolver(R.string.sync_progress_group_renewing_locks, groupId)
44         is TaskProgressLabel.SyncGroupLockRenewalFailed ->
45             resolver(R.string.sync_progress_group_lock_renewal_failed, groupId)
46
47         is TaskProgressLabel.ClearContentProgress ->
48             resolver(R.string.task_progress_clear_content, done, total)
49
50         is TaskProgressLabel.VaultTask -> when (step) {
51             VaultTaskStep.DumpStorageLog -> resolver(R.string.task_progress_dump_storage_log)
52             VaultTaskStep.CreateStorage -> resolver(R.string.task_progress_create_storage)
53             VaultTaskStep.EnableEncryption -> resolver(R.string.task_progress_enable_encryption)
54             VaultTaskStep.DecryptRunning -> resolver(R.string.task_progress_decrypt_running)
55             VaultTaskStep.CloseStorage -> resolver(R.string.task_progress_close_storage)
56             VaultTaskStep.DisableEncryption ->
resolver(R.string.task_progress_disable_encryption)
57             VaultTaskStep.RenameStorage -> resolver(R.string.task_progress_rename_storage)
58             VaultTaskStep.RemoveStorage -> resolver(R.string.task_progress_remove_storage)
59             VaultTaskStep.ClearSyncLock -> resolver(R.string.task_progress_clear_sync_lock)
60             VaultTaskStep.AddRemoteVault -> resolver(R.string.task_progress_add_remote_vault)
61             VaultTaskStep.RemoveRemoteVault ->
resolver(R.string.task_progress_remove_remote_vault)
62             VaultTaskStep.RetryRemoteVault ->
resolver(R.string.task_progress_retry_remote_vault)
63             VaultTaskStep.RescanVaultStorages ->
resolver(R.string.task_progress_rescan_vault_storages)
64             VaultTaskStep.Save2FaToken -> resolver(R.string.task_progress_save_2fa_token)
65             VaultTaskStep.Delete2FaToken -> resolver(R.string.task_progress_delete_2fa_token)
66             VaultTaskStep.SaveTextSecret -> resolver(R.string.task_progress_save_text_secret)
67             VaultTaskStep.DeleteTextSecret ->
resolver(R.string.task_progress_delete_text_secret)
68         }
69
70         is TaskProgressLabel.TestElapsed ->
71             resolver(R.string.task_pipeline_test_elapsed, elapsedSec, totalSec)
72     }
73

```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/
TaskProgressLabelsCompose.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import androidx.compose.runtime.Composable
4 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
5
6 @Composable
7 fun TaskProgressLabel.resolveText(): String = resolve(composableUiStringResolver())
8
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/
UiStringResolver.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import androidx.annotation.PluralsRes
4 import androidx.annotation.StringRes
5
6 /** Разрешение Android-строк для код-домена (ViewModel, без Compose). */
7 interface UiStringResolver {
8     operator fun invoke(@StringRes id: Int, vararg formatArgs: Any): String
9
10     fun plurals(@PluralsRes id: Int, quantity: Int, vararg formatArgs: Any): String
11 }
12
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/
UiStringResolverCompose.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import androidx.annotation.StringRes
4 import androidx.compose.runtime.Composable
5 import androidx.compose.runtime.remember
6 import androidx.compose.ui.platform.LocalResources
7 import androidx.compose.ui.res.stringResource
8
9 @Composable
10 fun composableUiStringResolver(): UiStringResolver {
11     val resources = LocalResources.current
12     return remember(resources) {
13         object : UiStringResolver {
14             override fun invoke(id: Int, vararg formatArgs: Any): String =
15                 if (formatArgs.isEmpty()) {
16                     resources.getString(id)
17                 } else {
18                     resources.getString(id, *formatArgs)
19                 }
20
21             override fun plurals(id: Int, quantity: Int, vararg formatArgs: Any): String =
22                 if (formatArgs.isEmpty()) {
23                     resources.getQuantityString(id, quantity)
24                 } else {
25                     resources.getQuantityString(id, quantity, *formatArgs)
26                 }
27         }
28     }
29 }
30
31 @Composable
32 fun resolveString(@StringRes id: Int, vararg formatArgs: Any): String =
33     if (formatArgs.isEmpty()) {
34         stringResource(id)
35     } else {
36         stringResource(id, *formatArgs)
37     }
38
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/
UserNotification.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import androidx.annotation.StringRes
4
5 sealed class UserNotification {
6     data class TextRes(@param:StringRes val id: Int, val formatArgs: List<Any> =
emptyList()) : UserNotification()
7     data class Plain(val message: String) : UserNotification()
8 }
9
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/resources/
UserNotificationCompose.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import androidx.compose.runtime.Composable
4 import androidx.compose.ui.res.stringResource
5
6 @Composable
7 fun UserNotification.resolveText(): String = when (this) {
8     is UserNotification.TextRes -> {
9         if (formatArgs.isEmpty()) {
10             stringResource(id)
11         } else {
12             stringResource(id, *formatArgs.toTypedArray())
13         }
14     }
15     is UserNotification.Plain -> message
16 }
17
18 fun UserNotification.resolve(resolver: UiStringResolver): String = when (this) {
19     is UserNotification.TextRes -> {
20         if (formatArgs.isEmpty()) {
21             resolver(id)
22         } else {
23             resolver(id, *formatArgs.toTypedArray())
24         }
25     }
26     is UserNotification.Plain -> message
27 }
28
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/
ScreenRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens
2
3 import android.os.Parcelable
4 import kotlinx.serialization.Serializable
5
6 @Serializable
7 abstract class ScreenRoute : Parcelable
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
MainRoute.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.main
2
3  import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4  import kotlinx.parcelize.Parcelize
5  import kotlinx.serialization.Serializable
6
7  @Serializable
8  @Parcelize
9  open class MainRoute: ScreenRoute()
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/ MainScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main
2
3 import androidx.compose.animation.core.tween
4 import androidx.compose.animation.fadeIn
5 import androidx.compose.animation.fadeOut
6 import androidx.compose.foundation.layout.Box
7 import androidx.compose.foundation.layout.WindowInsets
8 import androidx.compose.foundation.layout.fillMaxSize
9 import androidx.compose.foundation.layout.navigationBarsPadding
10 import androidx.compose.foundation.layout.padding
11 import androidx.compose.material.icons.Icons
12 import androidx.compose.material.icons.outlined.Cloud
13 import androidx.compose.material.icons.outlined.Folder
14 import androidx.compose.material3.ExperimentalMaterial3Api
15 import androidx.compose.material3.Scaffold
16 import androidx.compose.runtime.Composable
17 import androidx.compose.runtime.getValue
18 import androidx.compose.runtime.remember
19 import androidx.compose.ui.Alignment
20 import androidx.compose.ui.Modifier
21 import androidx.compose.ui.unit.dp
22 import androidx.compose.ui.zIndex
23 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
24 import androidx.lifecycle.compose.collectAsStateWithLifecycle
25 import androidx.navigation.compose.NavHost
26 import androidx.navigation.compose.composable
27 import androidx.navigation.compose.currentBackStackEntryAsState
28 import androidx.navigation.toRoute
29 import com.github.nullptroma.wallenc.ui.R
30 import com.github.nullptroma.wallenc.ui.elements.AnimatedFloatingBackButton
31 import com.github.nullptroma.wallenc.ui.elements.FloatingWallencNavigationBar
32 import com.github.nullptroma.wallenc.ui.elements.WallencNestedNavBarGap
33 import com.github.nullptroma.wallenc.ui.navigation.NavBarItemData
34 import com.github.nullptroma.wallenc.ui.navigation.NavigationState
35 import com.github.nullptroma.wallenc.ui.navigation.isMainTopLevelRoute
36 import com.github.nullptroma.wallenc.ui.navigation.isTextEditDestination
37 import com.github.nullptroma.wallenc.ui.navigation.shouldShowMainFloatingBack
38 import com.github.nullptroma.wallenc.ui.navigation.rememberNavigationState
39 import com.github.nullptroma.wallenc.ui.screens.main.screens.remotes.RemoteVaultsRoute
40 import com.github.nullptroma.wallenc.ui.screens.main.screens.remotes.RemoteVaultsScreen
41 import com.github.nullptroma.wallenc.ui.screens.main.screens.remotes.RemoteVaultsViewModel
42 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.StorageHomeRoute
```



```

43 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.StorageHomeScreen
44 import
45 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretDetailsRoute
46 import
47 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretDetailsScreen
48 import
49 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretEditRoute
50 import
51 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretEditScreen
52 import
53 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretsRoute
54 import
55 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretsScreen
56 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa.TwoFaTokensRoute
57 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa.TwoFaTokensScreen
58 import com.github.nullptroma.wallenc.ui.screens.main.screens.vault.LocalVaultRoute
59 import com.github.nullptroma.wallenc.ui.screens.main.screens.vault.LocalVaultScreen
60 import com.github.nullptroma.wallenc.ui.screens.main.screens.vault.LocalVaultViewModel
61 import com.github.nullptroma.wallenc.ui.screens.main.screens.vault.RemoteVaultViewModel
62 import com.github.nullptroma.wallenc.ui.screens.main.screens.vault.VaultBrowserRoute
63 import com.github.nullptroma.wallenc.ui.screens.main.screens.vault.VaultBrowserScreen
64 import com.github.nullptroma.wallenc.ui.screens.shared.TextEditRoute
65 import com.github.nullptroma.wallenc.ui.screens.shared.TextEditScreen
66
67 @OptIn(ExperimentalMaterial3Api::class)
68 @Composable
69 fun MainScreen(
70     modifier: Modifier = Modifier,
71     viewModel: MainViewModel = hiltViewModel(),
72     navState: NavigationState = rememberNavigationState(),
73 ) {
74     val routes = viewModel.routes
75     val mainUi by viewModel.state.collectAsStateWithLifecycle()
76     val localVaultViewModel: LocalVaultViewModel = hiltViewModel()
77     val remoteVaultsViewModel: RemoteVaultsViewModel = hiltViewModel()
78
79     val childBackStackEntry by navState.navHostController.currentBackStackEntryAsState()
80     val childRoute = childBackStackEntry?.destination?.route
81     val showWorkStatusBar = !isTextEditDestination(childRoute)
82     val showMainBottomNav = isMainTopLevelRoute(childRoute)
83     val showFloatingBack = shouldShowMainFloatingBack(childRoute) && navState.canPop()
84     val workStatus = mainUi.workStatus
85     val onBackPressed: () -> Unit = { navState.pop() }
86
87     val topLevelNavBarItems = remember {
88         mapOf(

```

```

83         LocalVaultRoute::class.qualifiedName!! to NavBarItemData(
84             nameStringResourceId = R.string.nav_label_local_vault,
85             screenRouteClass = LocalVaultRoute::class.qualifiedName!!,
86             icon = Icons.Outlined.Folder,
87             iconContentDescriptionResourceId = R.string.nav_cd_local_vault,
88         ),
89         RemoteVaultsRoute::class.qualifiedName!! to NavBarItemData(
90             nameStringResourceId = R.string.nav_label_remote_vaults,
91             screenRouteClass = RemoteVaultsRoute::class.qualifiedName!!,
92             icon = Icons.Outlined.Cloud,
93             iconContentDescriptionResourceId = R.string.nav_cd_remote_vaults,
94         ),
95     )
96 }
97
98 Scaffold(
99     modifier = modifier,
100     contentWindowInsets = WindowInsets(0.dp),
101     topBar = {
102         if (showWorkStatusBar) {
103             MainWorkStatusBar(status = workStatus)
104         }
105     },
106     bottomBar = {
107         if (showMainBottomNav) {
108             Box(
109                 modifier = Modifier
110                     .padding(horizontal = 16.dp)
111                     .padding(top = WallencNestedNavBarGap, bottom =
WallencNestedNavBarGap),
112                 ) {
113                 FloatingWallencNavigationBar(
114                     compact = true,
115                     items = topLevelNavBarItems,
116                     routes = routes,
117                     currentRoute = childRoute,
118                     onNavigate = { item ->
119                         val route = routes[item.screenRouteClass]
120                         ?: error("Route ${item.screenRouteClass} not found")
121                         navState.changeTop(route)
122                     },
123                 )
124             }
125         }
126     },

```

```

127     ) { innerPaddings ->
128         Box(
129             modifier = Modifier
130                 .fillMaxSize()
131                 .padding(innerPaddings),
132         ) {
133             NavHost(
134                 navController = navState.navHostController,
135                 startDestination = routes[LocalVaultRoute::class.qualifiedName]!!,
136                 modifier = Modifier.fillMaxSize(),
137             ) {
138                 composable<LocalVaultRoute>{
139                     enterTransition = { fadeIn(tween(200)) },
140                     exitTransition = { fadeOut(tween(200)) },
141                 } {
142                     LocalVaultScreen(
143                         viewModel = localVaultViewModel,
144                         onOpenStorageHome = { storageUuid ->
145                             navState.push(StorageHomeRoute(storageUuid = storageUuid))
146                         },
147                     )
148                 }
149                 composable<RemoteVaultsRoute>{
150                     enterTransition = { fadeIn(tween(200)) },
151                     exitTransition = { fadeOut(tween(200)) },
152                 } {
153                     RemoteVaultsScreen(
154                         viewModel = remoteVaultsViewModel,
155                         onOpenVault = { item ->
156                             navState.push(VaultBrowserRoute(item.uuid.toString()))
157                         },
158                     )
159                 }
160                 composable<VaultBrowserRoute>{
161                     enterTransition = { fadeIn(tween(200)) },
162                     exitTransition = { fadeOut(tween(200)) },
163                 } { entry ->
164                     val remoteVaultViewModel: RemoteVaultViewModel = hiltViewModel(entry)
165                     val route: VaultBrowserRoute = entry.toRoute()
166                     VaultBrowserScreen(
167                         viewModel = remoteVaultViewModel,
168                         onOpenStorageHome = { storageUuid ->
169                             navState.push(
170                                 StorageHomeRoute(
171                                     vaultUuid = route.vaultUuid,

```

```

172             storageUuid = storageUuid,
173         ),
174     )
175     },
176 )
177 }
178 composable<StorageHomeRoute>(
179     enterTransition = { fadeIn(tween(200)) },
180     exitTransition = { fadeOut(tween(200)) },
181 ) {
182     StorageHomeScreen(
183         onOpenTwoFa = { storageUuid ->
184             navState.push(TwoFaTokensRoute(storageUuid))
185         },
186         onOpenTextSecrets = { storageUuid ->
187             navState.push(TextSecretsRoute(storageUuid))
188         },
189     )
190 }
191 composable<TwoFaTokensRoute>(
192     enterTransition = { fadeIn(tween(200)) },
193     exitTransition = { fadeOut(tween(200)) },
194 ) {
195     TwoFaTokensScreen()
196 }
197 composable<TextSecretsRoute>(
198     enterTransition = { fadeIn(tween(200)) },
199     exitTransition = { fadeOut(tween(200)) },
200 ) { entry ->
201     val route: TextSecretsRoute = entry.toRoute()
202     TextSecretsScreen(
203         onOpenSecret = { secret ->
204             navState.push(
205                 TextSecretDetailsRoute(
206                     storageUuid = route.storageUuid,
207                     secretId = secret.id,
208                 ),
209             )
210         },
211         onCreateSecret = {
212             navState.push(
213                 TextSecretEditRoute(
214                     storageUuid = route.storageUuid,
215                     secretId = null,

```

```

216         ),
217     )
218 },
219 )
220 }
221 composable<TextSecretDetailsRoute>(
222     enterTransition = { fadeIn(tween(200)) },
223     exitTransition = { fadeOut(tween(200)) },
224 ) { entry ->
225     val route: TextSecretDetailsRoute = entry.toRoute()
226     TextSecretDetailsScreen(
227         onEdit = { secretId ->
228             navState.push(
229                 TextSecretEditRoute(
230                     storageUuid = route.storageUuid,
231                     secretId = secretId,
232                 ),
233             )
234         },
235         onDelete = { navState.pop() },
236     )
237 }
238 composable<TextSecretEditRoute>(
239     enterTransition = { fadeIn(tween(200)) },
240     exitTransition = { fadeOut(tween(200)) },
241 ) { entry ->
242     val route: TextSecretEditRoute = entry.toRoute()
243     TextSecretEditScreen(
244         onSave = { savedSecretId ->
245             val editingExisting = route.secretId != null
246             navState.pop()
247             if (!editingExisting) {
248                 navState.push(
249                     TextSecretDetailsRoute(
250                         storageUuid = route.storageUuid,
251                         secretId = savedSecretId,
252                     ),
253                 )
254             }
255         },
256     )
257 }
258 composable<TextEditRoute> {
259     val route: TextEditRoute = it.toRoute()
260     TextEditScreen(text = route.text)

```

```
261         }
262     }
263     AnimatedFloatingBackButton(
264         visible = showFloatingBack,
265         onClick = onBack,
266         modifier = Modifier
267             .zIndex(1f)
268             .align(Alignment.BottomStart)
269             .navigationBarsPadding()
270             .padding(start = 12.dp, bottom = 12.dp),
271     )
272 }
273 }
274 }
275
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
MainScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main
2
3 import androidx.compose.runtime.Immutable
4
5 @Immutable
6 data class MainScreenState(
7     val workStatus: MainWorkStatus = MainWorkStatus.Idle,
8 )
9
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/ MainViewModel.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.main
2
3  import androidx.compose.runtime.mutableStateOf
4  import androidx.lifecycle.SavedStateHandle
5  import androidx.lifecycle.viewModelScope
6  import androidx.lifecycle.viewmodel.compose.SavedStateHandleSaveableApi
7  import androidx.lifecycle.viewmodel.compose.saveable
8  import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
9  import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
10 import com.github.nullptroma.wallenc.domain.tasks.PipelineState
11 import com.github.nullptroma.wallenc.domain.tasks.TaskForegroundUiState
12 import com.github.nullptroma.wallenc.domain.tasks.TaskProgress
13 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
14 import com.github.nullptroma.wallenc.ui.R
15 import com.github.nullptroma.wallenc.ui.ViewModelBase
16 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
17 import com.github.nullptroma.wallenc.ui.resources.resolve
18 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
19 import com.github.nullptroma.wallenc.ui.screens.main.screens.remotes.RemoteVaultsRoute
20 import com.github.nullptroma.wallenc.ui.screens.main.screens.vault.LocalVaultRoute
21 import dagger.hilt.android.lifecycle.HiltViewModel
22 import kotlinx.coroutines.ExperimentalCoroutinesApi
23 import kotlinx.coroutines.flow.combine
24 import kotlinx.coroutines.flow.distinctUntilChanged
25 import kotlinx.coroutines.flow.flatMapLatest
26 import kotlinx.coroutines.flow.flowOf
27 import kotlinx.coroutines.launch
28 import javax.inject.Inject
29
30 @OptIn(ExperimentalCoroutinesApi::class)
31 @HiltViewModel
32 class MainViewModel @Inject constructor(
33     savedStateHandle: SavedStateHandle,
34     private val taskOrchestrator: ITaskOrchestrator,
35     private val vaultsManager: IVaultsManager,
36     private val uiStrings: UiStringResolver,
37 ) : ViewModelBase<MainScreenState>(MainScreenState()) {
38
39     @OptIn(SavedStateHandleSaveableApi::class)
40     var routes by savedStateHandle.saveable {
41         mutableStateOf(
42             mapOf<String, ScreenRoute>(
```



```

43         LocalVaultRoute::class.qualifiedName!! to LocalVaultRoute(),
44         RemoteVaultsRoute::class.qualifiedName!! to RemoteVaultsRoute(),
45     ),
46 )
47 }
48     private set
49
50     init {
51         viewModelScope.launch {
52             combine(
53                 taskOrchestrator.foregroundUi,
54                 taskOrchestrator.pipelineState,
55                 vaultsManager.vaults.flatMapLatest { vaults ->
56                     if (vaults.isEmpty()) {
57                         flowOf(false)
58                     } else {
59                         combine(vaults.map { it.storagesScanInProgress }) { flags ->
60                             flags.any { it }
61                         }
62                     }
63                 },
64             ) { fg, pipe, anyVaultScanning ->
65                 mapWorkStatus(fg, pipe, anyVaultScanning)
66             }
67                 .distinctUntilChanged()
68                 .collect { status ->
69                     updateState(state.value.copy(workStatus = status))
70                 }
71         }
72     }
73
74     private fun mapWorkStatus(
75         fg: TaskForegroundUiState,
76         pipe: PipelineState,
77         anyVaultScanning: Boolean,
78     ): MainWorkStatus {
79         when (fg) {
80             is TaskForegroundUiState.Visible -> {
81                 if (fg.tasks.isEmpty()) {
82                     return mapBackgroundWork(pipe, anyVaultScanning)
83                 }
84                 val head = fg.tasks.first()
85                 return activeWorkStatusFromProgress(head.title, head.progress)
86             }
87             TaskForegroundUiState.Hidden -> return mapBackgroundWork(pipe, anyVaultScanning)

```

```

88         }
89     }
90
91     private fun mapBackgroundWork(
92         pipe: PipelineState,
93         anyVaultScanning: Boolean,
94     ): MainWorkStatus {
95         val fromPipeline = mapPipelineRunningOnly(pipe)
96         if (fromPipeline != null) return fromPipeline
97         if (anyVaultScanning) {
98             return MainWorkStatus.Active(
99                 line = uiStrings(R.string.main_status_vault_scanning_storages),
100                 progressFraction = null,
101                 indeterminate = true,
102             )
103         }
104         return MainWorkStatus.Idle
105     }
106
107     private fun mapPipelineRunningOnly(pipe: PipelineState): MainWorkStatus? {
108         val running = pipe.tasks.filter { it.id in pipe.runningTaskIds }
109         if (running.isEmpty()) return null
110         if (running.size == 1) {
111             val t = running.first()
112             val progress = (t.state as? TaskRunState.Running)?.progress
113             return activeWorkStatusFromProgress(t.title, progress)
114         }
115         return MainWorkStatus.Active(
116             line = uiStrings(R.string.main_status_multiple_tasks, running.size),
117             progressFraction = null,
118             indeterminate = true,
119         )
120     }
121
122     private fun activeWorkStatusFromProgress(title: String, progress: TaskProgress?):
MainWorkStatus.Active {
123         val frac = progress?.fraction
124         val indeterminate = progress == null || frac == null
125         val label = progress?.label?.resolve(uiStrings)
126         val line = if (label != null) "$title$TITLE_LABEL_SEPARATOR$label" else title
127         return MainWorkStatus.Active(
128             line = line,
129             progressFraction = frac,
130             indeterminate = indeterminate,
131         )

```

```
132     }
133
134     private companion object {
135         private const val TITLE_LABEL_SEPARATOR = " — "
136     }
137 }
138
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
MainWorkStatus.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main
2
3 /** Состояние полосы «текущая работа» на Main. */
4 sealed class MainWorkStatus {
5     data object Idle : MainWorkStatus()
6
7     /** [line] – строка для отображения (уже локализованная, из оркестратора или фолбэк). */
8     data class Active(
9         val line: String,
10        val progressFraction: Float?,
11        val indeterminate: Boolean,
12    ) : MainWorkStatus()
13 }
14
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/ MainWorkStatusBar.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Row
6 import androidx.compose.foundation.layout.fillMaxWidth
7 import androidx.compose.foundation.layout.height
8 import androidx.compose.foundation.layout.padding
9 import androidx.compose.material3.LinearProgressIndicator
10 import androidx.compose.material3.MaterialTheme
11 import androidx.compose.material3.Surface
12 import androidx.compose.material3.Text
13 import androidx.compose.runtime.Composable
14 import androidx.compose.ui.Alignment
15 import androidx.compose.ui.Modifier
16 import androidx.compose.ui.res.stringResource
17 import androidx.compose.ui.text.style.TextAlign
18 import androidx.compose.ui.unit.dp
19 import com.github.nullptroma.wallenc.ui.R
20
21 private val progressTrackHeight = 2.dp
22
23 /**
24  * Полоса статуса работы на Main: всегда видна (пока экран не скрывает её целиком).
25  * Слева подпись «Статус:», справа – описание текущей задачи или пусто; внизу – тонкий
26  * индикатор прогресса.
27  */
28 @Composable
29 fun MainWorkStatusBar(
30     status: MainWorkStatus,
31     modifier: Modifier = Modifier,
32 ) {
33     val taskLine = when (status) {
34         MainWorkStatus.Idle -> ""
35         is MainWorkStatus.Active -> status.line
36     }
37     Surface(
38         modifier = modifier.fillMaxWidth(),
39         color = MaterialTheme.colorScheme.surfaceVariant,
40         tonalElevation = 1.dp,
41     ) {
42         Column(
```

```

42         modifier = Modifier
43             .fillMaxWidth()
44             .padding(horizontal = 12.dp, vertical = 6.dp),
45     ) {
46         Row(
47             modifier = Modifier.fillMaxWidth(),
48             verticalAlignment = Alignment.CenterVertically,
49             horizontalArrangement = Arrangement.spacedBy(8.dp),
50         ) {
51             Text(
52                 text = stringResource(R.string.main_work_status_label),
53                 style = MaterialTheme.typography.labelLarge,
54                 color = MaterialTheme.colorScheme.onSurfaceVariant,
55             )
56             Text(
57                 text = taskLine,
58                 modifier = Modifier.weight(1f),
59                 style = MaterialTheme.typography.bodySmall,
60                 color = MaterialTheme.colorScheme.onSurfaceVariant,
61                 maxLines = 2,
62                 textAlign = TextAlign.End,
63             )
64         }
65         when (status) {
66             MainWorkStatus.Idle -> {
67                 LinearProgressIndicator(
68                     progress = { 0f },
69                     modifier = Modifier
70                         .fillMaxWidth()
71                         .height(progressTrackHeight)
72                         .padding(top = 6.dp),
73                 )
74             }
75             is MainWorkStatus.Active -> {
76                 if (status.indeterminate) {
77                     LinearProgressIndicator(
78                         modifier = Modifier
79                             .fillMaxWidth()
80                             .height(progressTrackHeight)
81                             .padding(top = 6.dp),
82                     )
83                 } else {
84                     val frac = status.progressFraction ?: 0f
85                     LinearProgressIndicator(
86                         progress = { frac },

```

```
87         modifier = Modifier
88             .fillMaxWidth()
89             .height(progressTrackHeight)
90             .padding(top = 6.dp),
91     )
92 }
93 }
94 }
95 }
96 }
97 }
98
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/remotes/RemoteVaultsRoute.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.main.screens.remotes
2
3  import com.github.nullptroma.wallenc.ui.screens.main.MainRoute
4  import kotlinx.parcelize.Parcelize
5  import kotlinx.serialization.Serializable
6
7  @Serializable
8  @Parcelize
9  class RemoteVaultsRoute : MainRoute()
10
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/remotes/RemoteVaultsScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.remotes
2
3 import android.widget.Toast
4 import androidx.compose.foundation.clickable
5 import androidx.compose.foundation.interaction.MutableInteractionSource
6 import androidx.compose.foundation.layout.Arrangement
7 import androidx.compose.foundation.layout.Box
8 import androidx.compose.foundation.layout.Column
9 import androidx.compose.foundation.layout.PaddingValues
10 import androidx.compose.foundation.layout.Row
11 import androidx.compose.foundation.layout.Spacer
12 import androidx.compose.foundation.layout.WindowInsets
13 import androidx.compose.foundation.layout.fillMaxSize
14 import androidx.compose.foundation.layout.fillMaxWidth
15 import androidx.compose.foundation.layout.height
16 import androidx.compose.foundation.layout.padding
17 import androidx.compose.foundation.lazy.LazyColumn
18 import androidx.compose.foundation.lazy.items
19 import androidx.compose.foundation.shape.RoundedCornerShape
20 import androidx.compose.material.icons.Icons
21 import androidx.compose.material.icons.filled.Add
22 import androidx.compose.material.icons.filled.Delete
23 import androidx.compose.material3.AlertDialog
24 import androidx.compose.material3.CardDefaults
25 import androidx.compose.material3.CircularProgressIndicator
26 import androidx.compose.material3.ElevatedCard
27 import androidx.compose.material3.FilledTonalButton
28 import androidx.compose.material3.FloatingActionButton
29 import androidx.compose.material3.Icon
30 import androidx.compose.material3.IconButton
31 import androidx.compose.material3.MaterialTheme
32 import androidx.compose.material3.Scaffold
33 import androidx.compose.material3.Surface
34 import androidx.compose.material3.Text
35 import androidx.compose.material3.TextButton
36 import androidx.compose.runtime.Composable
37 import androidx.compose.runtime.getValue
38 import androidx.compose.runtime.remember
39 import androidx.compose.ui.Alignment
40 import androidx.compose.ui.draw.alpha
41 import androidx.compose.ui.Modifier
42 import androidx.compose.ui.platform.LocalContext
```

```

43 import androidx.compose.ui.platform.LocalResources
44 import androidx.compose.ui.res.stringResource
45 import androidx.compose.ui.unit.dp
46 import androidx.compose.ui.window.Dialog
47 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
48 import androidx.lifecycle.compose.collectAsStateWithLifecycle
49 import com.github.nullptroma.wallenc.ui.R
50 import com.github.nullptroma.wallenc.ui.resources.toUserNotification
51 import com.github.nullptroma.wallenc.vault.contract.CloudBrand
52 import com.github.nullptroma.wallenc.vault.contract.VaultLinkOutcome
53
54 @Composable
55 fun RemoteVaultsScreen(
56     modifier: Modifier = Modifier,
57     viewModel: RemoteVaultsViewModel = hiltViewModel(),
58     onOpenVault: (RemoteVaultListItem) -> Unit,
59 ) {
60     val uiState by viewModel.uiState.collectAsStateWithLifecycle()
61     val context = LocalContext.current
62     val resources = LocalResources.current
63
64     Box {
65         Scaffold(
66             modifier = modifier,
67             contentWindowInsets = WindowInsets(0.dp),
68             floatingActionButton = {
69                 FloatingActionButton(
70                     onClick = {
71                         if (!uiState.isBusy) viewModel.setAddChoiceVisible(true)
72                     },
73                     modifier = Modifier.alpha(if (uiState.isBusy) 0.38f else 1f),
74                 ) {
75                     Icon(
76                         Icons.Filled.Add,
77                         contentDescription = stringResource(R.string.remote_vaults_add_cd),
78                     )
79                 }
80             },
81         ) { innerPadding ->
82             if (uiState.vaults.isEmpty()) {
83                 Text(
84                     text = stringResource(R.string.remote_vaults_empty_hint),
85                     modifier = Modifier
86                         .padding(innerPadding)
87                         .padding(24.dp),

```

```

88         style = MaterialTheme.typography.bodyLarge,
89         color = MaterialTheme.colorScheme.onSurfaceVariant,
90     )
91 } else {
92     LazyColumn(
93         modifier = Modifier
94             .fillMaxSize()
95             .padding(innerPadding),
96         contentPadding = PaddingValues(16.dp),
97         verticalArrangement = Arrangement.spacedBy(12.dp),
98     ) {
99         items(uiState.vaults, key = { it.uuid }) { item ->
100             ElevatedCard(
101                 modifier = Modifier
102                     .fillMaxWidth()
103                     .clickable(
104                         enabled = !uiState.isBusy,
105                         interactionSource = remember
106                             { MutableInteractionSource() },
107                         indication = null,
108                     ) { onOpenVault(item) },
109                 shape = RoundedCornerShape(16.dp),
110                 elevation = CardDefaults.elevatedCardElevation(defaultElevation
111                     = 2.dp),
112                 colors = CardDefaults.elevatedCardColors(
113                     containerColor =
114                     MaterialTheme.colorScheme.surfaceContainerHigh,
115                 ),
116             ) {
117                 Row(
118                     modifier = Modifier
119                         .fillMaxWidth()
120                         .padding(16.dp),
121                     verticalAlignment = Alignment.CenterVertically,
122                     horizontalArrangement = Arrangement.SpaceBetween,
123                 ) {
124                     Column(modifier = Modifier.weight(1f)) {
125                         Text(
126                             text = item.label,
127                             style = MaterialTheme.typography.titleMedium,
128                             color = MaterialTheme.colorScheme.onSurface,
129                         )
130                         Spacer(modifier = Modifier.height(4.dp))
131                         Text(
132                             text = when (item.brand) {

```

```

130                                     CloudBrand.YANDEX ->
131     stringResource(R.string.remote_vault_type_yandex)
132                                     },
133                                     style = MaterialTheme.typography.bodyMedium,
134                                     color = MaterialTheme.colorScheme.onSurfaceVariant,
135                                     )
136     Spacer(modifier = Modifier.height(6.dp))
137     if (!item.isAvailable) {
138         Text(
139             text =
140             stringResource(R.string.remote_vault_unavailable),
141             style = MaterialTheme.typography.bodySmall,
142             color = MaterialTheme.colorScheme.error,
143         )
144         Spacer(modifier = Modifier.height(6.dp))
145         FilledTonalButton(
146             onClick = { viewModel.retryVault(item.uuid) },
147             enabled = !uiState.isBusy,
148         ) {
149             Text(
150                 if (item.isRefreshing) {
151                     stringResource(R.string.remote_vault_retrying)
152                 } else {
153                     stringResource(R.string.remote_vault_retry_action)
154                 },
155             )
156         }
157     }
158     IconButton(
159         onClick = { viewModel.requestDeleteVault(item) },
160         enabled = !uiState.isBusy,
161     ) {
162         Icon(
163             Icons.Filled.Delete,
164             contentDescription =
165             stringResource(R.string.remote_vault_delete_cd),
166             tint = MaterialTheme.colorScheme.error,
167         )
168     }
169 }
170 }
171 }
172 }

```

```

173     }
174
175     if (uiState.isBusy) {
176         CircularProgressIndicator(
177             modifier = Modifier
178                 .align(Alignment.Center)
179                 .padding(24.dp),
180         )
181     }
182 }
183
184 if (uiState.addChoiceVisible) {
185     Dialog(
186         onDismissRequest = { if (!uiState.isBusy)
viewModel.setAddChoiceVisible(false) },
187     ) {
188         Surface(
189             shape = RoundedCornerShape(28.dp),
190             color = MaterialTheme.colorScheme.surfaceContainerHigh,
191             tonalElevation = 3.dp,
192         ) {
193             Column(
194                 modifier = Modifier
195                     .padding(24.dp)
196                     .fillMaxWidth(),
197             ) {
198                 Text(
199                     text = stringResource(R.string.remote_vaults_add_title),
200                     style = MaterialTheme.typography.headlineSmall,
201                     color = MaterialTheme.colorScheme.onSurface,
202                 )
203                 Spacer(modifier = Modifier.height(12.dp))
204                 Text(
205                     text = stringResource(R.string.remote_vaults_add_pick_provider),
206                     style = MaterialTheme.typography.bodyLarge,
207                     color = MaterialTheme.colorScheme.onSurfaceVariant,
208                 )
209                 Spacer(modifier = Modifier.height(16.dp))
210                 FilledTonalButton(
211                     onClick = {
212                         viewModel.setAddChoiceVisible(false)
213                         viewModel.remoteAuthenticator.beginLink(CloudBrand.YANDEX)
214
215                     when (outcome) {
216                         is VaultLinkOutcome.Success ->

```

```

216         viewModel.onLinkSucceeded(outcome.registration)
217         is VaultLinkOutcome.Failed -> {
218             val notification =
outcome.reason.toUserNotification()
219             val text = if (notification.formatArgs.isEmpty()) {
220                 resources.getString(notification.id)
221             } else {
222                 resources.getString(
223                     notification.id,
224                     *notification.formatArgs.toTypedArray(),
225                 )
226             }
227             Toast.makeText(context, text,
Toast.LENGTH_LONG).show()
228         }
229         VaultLinkOutcome.Cancelled -> { }
230     }
231 }
232 },
233 modifier = Modifier.fillMaxWidth(),
234 enabled = !uiState.isBusy,
235 ) {
236     Text(stringResource(R.string.remote_vaults_provider_yandex))
237 }
238 Spacer(modifier = Modifier.height(16.dp))
239 Row(
240     modifier = Modifier.fillMaxWidth(),
241     horizontalArrangement = Arrangement.End,
242 ) {
243     Text(
244         text = stringResource(R.string.remote_vaults_add_cancel),
245         style = MaterialTheme.typography.labelLarge,
246         color = MaterialTheme.colorScheme.primary,
247         modifier = Modifier
248             .clickable(
249                 enabled = !uiState.isBusy,
250                 interactionSource = remember
{ MutableInteractionSource() },
251                 indication = null,
252             ) { viewModel.setAddChoiceVisible(false) }
253             .padding(vertical = 8.dp, horizontal = 4.dp),
254     )
255 }
256 }
257 }
258 }

```

```

259     }
260
261     uiState.vaultPendingDelete?.let { pending ->
262         AlertDialog(
263             onDismissRequest = { if (!uiState.isBusy) viewModel.dismissDeleteVault() },
264             title = {
265                 Text(stringResource(R.string.remote_vault_remove_title))
266             },
267             text = {
268                 Text(stringResource(R.string.remote_vault_remove_message, pending.label))
269             },
270             confirmButton = {
271                 TextButton(
272                     onClick = { viewModel.confirmDeleteVault() },
273                     enabled = !uiState.isBusy,
274                 ) {
275                     Text(stringResource(R.string.remove))
276                 }
277             },
278             dismissButton = {
279                 TextButton(
280                     onClick = { viewModel.dismissDeleteVault() },
281                     enabled = !uiState.isBusy,
282                 ) {
283                     Text(stringResource(R.string.remote_vaults_add_cancel))
284                 }
285             },
286         )
287     }
288 }
289

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/remotes/RemoteVaultsScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.remotes
2
3 import com.github.nullptroma.wallenc.vault.contract.CloudBrand
4 import java.util.UUID
5
6 data class RemoteVaultListItem(
7     val uuid: UUID,
8     val brand: CloudBrand,
9     val label: String,
10    val isAvailable: Boolean,
11    val isRefreshing: Boolean,
12 )
13
14 data class RemoteVaultsScreenState(
15     val vaults: List<RemoteVaultListItem> = emptyList(),
16     val isBusy: Boolean = false,
17     val addChoiceVisible: Boolean = false,
18     val vaultPendingDelete: RemoteVaultListItem? = null,
19 )
20
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/remotes/RemoteVaultsViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.remotes
2
3 import androidx.lifecycle.ViewModelScope
4 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
5 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
6 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLevel
7 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
8 import com.github.nullptroma.wallenc.domain.tasks.VaultTaskStep
9 import com.github.nullptroma.wallenc.ui.R
10 import com.github.nullptroma.wallenc.ui.ViewModelBase
11 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
12 import com.github.nullptroma.wallenc.vault.contract.DescribedVault
13 import com.github.nullptroma.wallenc.vault.contract.RemoteVaultAuthenticator
14 import com.github.nullptroma.wallenc.vault.contract.VaultDescriptor
15 import com.github.nullptroma.wallenc.vault.contract.VaultRegistrar
16 import com.github.nullptroma.wallenc.vault.contract.VaultRegistration
17 import com.github.nullptroma.wallenc.vault.contract.described
18 import com.github.nullptroma.wallenc.vault.contract.remotes
19 import dagger.hilt.android.lifecycle.HiltViewModel
20 import kotlinx.coroutines.Dispatchers
21 import kotlinx.coroutines.ExperimentalCoroutinesApi
22 import kotlinx.coroutines.flow.SharingStarted
23 import kotlinx.coroutines.flow.combine
24 import kotlinx.coroutines.flow.flatMapLatest
25 import kotlinx.coroutines.flow.flowOf
26 import kotlinx.coroutines.flow.stateIn
27 import kotlinx.coroutines.withContext
28 import javax.inject.Inject
29
30 @HiltViewModel
31 @OptIn(ExperimentalCoroutinesApi::class)
32 class RemoteVaultsViewModel @Inject constructor(
33     private val vaultsManager: IVaultsManager,
34     private val vaultRegistrar: VaultRegistrar,
35     val remoteAuthenticator: RemoteVaultAuthenticator,
36     private val taskOrchestrator: ITaskOrchestrator,
37     private val uiStrings: UiStringResolver,
38 ) : ViewModelBase<RemoteVaultsScreenState>(RemoteVaultsScreenState()) {
39
40     private val remoteItems = vaultsManager.vaults
41         .flatMapLatest { all ->
42             val remotes = all.described().remotes
```

```

43         if (remotes.isEmpty()) {
44             flowOf(emptyList())
45         } else {
46             combine(remotes.map { vault ->
47                 combine(vault.isAvailable, vault.storagesScanInProgress) { available,
scanInProgress ->
48                     toRemoteItem(vault, available, scanInProgress)
49                 }
50             }) { arr -> arr.toList().filterNotNull() }
51         }
52     }
53
54     val uiState = combine(
55         remoteItems,
56         state,
57     ) { items, base ->
58         base.copy(
59             vaults = items,
60         )
61     }.stateIn(
62         viewModelScope,
63         SharingStarted.Eagerly,
64         RemoteVaultsScreenState(),
65     )
66
67     fun setAddChoiceVisible(visible: Boolean) {
68         updateState(state.value.copy(addChoiceVisible = visible))
69     }
70
71     fun setBusy(busy: Boolean) {
72         updateState(state.value.copy(isBusy = busy))
73     }
74
75     fun onLinkSucceeded(registration: VaultRegistration) {
76         setBusy(true)
77         taskOrchestrator.enqueue(
78             title = uiStrings(R.string.task_title_add_remote_vault),
79             dispatcher = Dispatchers.IO,
80             work = { ctx ->
81                 try {
82                     ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.AddRemoteVault))
83                     ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_adding_vault))
84                     vaultRegistrar.register(registration)
85                     ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_vault_added))

```

```

86         } catch (e: Exception) {
87             ctx.log(TaskLogLevel.Error,
uiStrings(R.string.task_log_add_vault_failed))
88         } finally {
89             withContext(Dispatchers.Main.immediate) {
90                 setBusy(false)
91                 setAddChoiceVisible(false)
92             }
93         }
94     },
95 )
96 }
97
98 fun requestDeleteVault(item: RemoteVaultListItem) {
99     updateState(state.value.copy(vaultPendingDelete = item))
100 }
101
102 fun dismissDeleteVault() {
103     updateState(state.value.copy(vaultPendingDelete = null))
104 }
105
106 fun confirmDeleteVault() {
107     val pending = state.value.vaultPendingDelete ?: return
108     val uuid = pending.uuid
109     setBusy(true)
110     taskOrchestrator.enqueue(
111         title = uiStrings(R.string.task_title_remove_remote_vault),
112         dispatcher = Dispatchers.IO,
113         work = { ctx ->
114             try {
115                 ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.RemoveRemoteVault))
116                 ctx.log(TaskLogLevel.Info,
uiStrings(R.string.task_log_removing_remote_vault))
117                 vaultRegistrar.unregister(uuid)
118                 ctx.log(TaskLogLevel.Info,
uiStrings(R.string.task_log_remote_vault_removed))
119             } catch (e: Exception) {
120                 ctx.log(TaskLogLevel.Error,
uiStrings(R.string.task_log_remove_vault_failed))
121             } finally {
122                 withContext(Dispatchers.Main.immediate) {
123                     setBusy(false)
124                     dismissDeleteVault()
125                 }
126             }
127         },

```

```

128         )
129     }
130
131     fun retryVault(vaultUuid: java.util.UUID) {
132         setBusy(true)
133         taskOrchestrator.enqueue(
134             title = uiStrings(R.string.task_title_retry_remote_vault),
135             dispatcher = Dispatchers.IO,
136             work = { ctx ->
137                 try {
138                     ctx.reportProgress(null,
139 TaskProgressLabel.VaultTask(VaultTaskStep.RetryRemoteVault))
140                     ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_retrying_vault))
141                     vaultRegistrar.retry(vaultUuid)
142                     ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_retry_requested))
143                 } catch (e: Exception) {
144                     ctx.log(TaskLogLevel.Error,
145 uiStrings(R.string.task_log_retry_vault_failed))
146                 } finally {
147                     withContext(Dispatchers.Main.immediate) {
148                         setBusy(false)
149                     }
150                 }
151             },
152         )
153     }
154
155     private fun toRemoteItem(
156         vault: DescribedVault,
157         isAvailable: Boolean,
158         isRefreshing: Boolean,
159     ): RemoteVaultListItem? {
160         val descriptor = vault.descriptor as? VaultDescriptor.LinkedRemote ?: return null
161         return RemoteVaultListItem(
162             uuid = descriptor.uuid,
163             brand = descriptor.brand,
164             label = descriptor.accountDisplayName,
165             isAvailable = isAvailable,
166             isRefreshing = isRefreshing,
167         )
168     }
169 }

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/StorageHomeRoute.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.main.screens.storage
2
3  import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4  import kotlinx.parcelize.Parcelize
5  import kotlinx.serialization.Serializable
6
7  @Serializable
8  @Parcelize
9  data class StorageHomeRoute(
10      val vaultUuid: String? = null,
11      val storageUuid: String,
12  ) : ScreenRoute()
13
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/StorageHomeScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Box
5 import androidx.compose.foundation.layout.Column
6 import androidx.compose.foundation.layout.Row
7 import androidx.compose.foundation.layout.size
8 import androidx.compose.foundation.layout.fillMaxSize
9 import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.padding
11 import androidx.compose.material.icons.Icons
12 import androidx.compose.material.icons.automirrored.outlined.Notes
13 import androidx.compose.material.icons.outlined.Lock
14 import androidx.compose.material3.Card
15 import androidx.compose.material3.CardDefaults
16 import androidx.compose.material3.CircularProgressIndicator
17 import androidx.compose.material3.Icon
18 import androidx.compose.material3.MaterialTheme
19 import androidx.compose.material3.Text
20 import androidx.compose.runtime.Composable
21 import androidx.compose.runtime.getValue
22 import androidx.compose.ui.Alignment
23 import androidx.compose.ui.Modifier
24 import androidx.compose.ui.graphics.vector.ImageVector
25 import androidx.compose.ui.res.stringResource
26 import androidx.compose.ui.unit.dp
27 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
28 import androidx.lifecycle.compose.collectAsStateWithLifecycle
29 import com.github.nullptroma.wallenc.ui.R
30 import com.github.nullptroma.wallenc.ui.elements.WallencScreenContentPadding
31 import com.github.nullptroma.wallenc.ui.elements.WallencScreenScaffold
32 import com.github.nullptroma.wallenc.ui.resources.resolveText
33
34 @Composable
35 fun StorageHomeScreen(
36     modifier: Modifier = Modifier,
37     viewModel: StorageHomeViewModel = hiltViewModel(),
38     onOpenTwoFa: (String) -> Unit,
39     onOpenTextSecrets: (String) -> Unit,
40 ) {
41     val uiState by viewModel.state.collectAsStateWithLifecycle()
42 }
```

```

43 WallencScreenScaffold(modifier = modifier) { innerPadding ->
44     WallencScreenContentPadding(innerPadding) {
45         Column(
46             modifier = Modifier.fillMaxSize(),
47             verticalArrangement = Arrangement.spacedBy(12.dp),
48         ) {
49             if (uiState.isLoading) {
50                 Box(
51                     modifier = Modifier.fillMaxSize(),
52                     contentAlignment = Alignment.Center,
53                 ) {
54                     CircularProgressIndicator()
55                 }
56                 return@Column
57             }
58
59             Text(
60                 text = uiState.storageName.ifBlank {
61                     stringResource(R.string.storage_home_unnamed_storage)
62                 },
63                 style = MaterialTheme.typography.headlineSmall,
64             )
65             Text(
66                 text = uiState.storageUuid,
67                 style = MaterialTheme.typography.bodySmall,
68                 color = MaterialTheme.colorScheme.onSurfaceVariant,
69             )
70             Text(
71                 text = stringResource(
72                     R.string.storage_home_status_line,
73                     if (uiState.isAvailable) {
74                         stringResource(R.string.storage_home_status_available)
75                     } else {
76                         stringResource(R.string.storage_home_status_unavailable)
77                     },
78                     if (uiState.isEncrypted) {
79                         stringResource(R.string.storage_home_status_encrypted)
80                     } else {
81                         stringResource(R.string.storage_home_status_not_encrypted)
82                     },
83                 ),
84             )
85
86             uiState.errorNotification?.let { notification ->
87                 Text(

```

```

88             text = notification.resolveText(),
89             color = MaterialTheme.colorScheme.error,
90         )
91     }
92
93     StorageSectionCard(
94         title = stringResource(R.string.storage_home_two_fa_title,
95 uiState.twoFaCount),
96         description = stringResource(R.string.storage_home_two_fa_subtitle),
97         icon = Icons.Outlined.Lock,
98         enabled = uiState.canManageDomainData,
99         onClick = { onOpenTwoFa(uiState.storageUuid) },
100     )
101
102     StorageSectionCard(
103         title = stringResource(R.string.storage_home_text_secrets_title,
104 uiState.textSecretsCount),
105         description = stringResource(R.string.storage_home_text_secrets_subtitle),
106         icon = Icons.AutoMirrored.Outlined.Notes,
107         enabled = uiState.canManageDomainData,
108         onClick = { onOpenTextSecrets(uiState.storageUuid) },
109     )
110
111     Text(
112         text = stringResource(R.string.storage_home_future_sections),
113         style = MaterialTheme.typography.bodySmall,
114         color = MaterialTheme.colorScheme.onSurfaceVariant,
115     )
116 }
117
118
119 @Composable
120 private fun StorageSectionCard(
121     title: String,
122     description: String,
123     icon: ImageVector,
124     enabled: Boolean,
125     onClick: () -> Unit,
126 ) {
127     Card(
128         modifier = Modifier.fillMaxWidth(),
129         colors = CardDefaults.elevatedCardColors(
130             containerColor = MaterialTheme.colorScheme.surfaceContainerHigh,

```



```

131         ),
132         onClick = onClick,
133         enabled = enabled,
134     ) {
135         Row(
136             modifier = Modifier
137                 .fillMaxWidth()
138                 .padding(16.dp),
139             horizontalArrangement = Arrangement.spacedBy(14.dp),
140         ) {
141             Icon(
142                 imageVector = icon,
143                 contentDescription = null,
144                 modifier = Modifier.size(28.dp),
145                 tint = MaterialTheme.colorScheme.primary,
146             )
147             Column(
148                 verticalArrangement = Arrangement.spacedBy(4.dp),
149             ) {
150                 Text(
151                     text = title,
152                     style = MaterialTheme.typography.titleMedium,
153                 )
154                 Text(
155                     text = description,
156                     style = MaterialTheme.typography.bodySmall,
157                     color = MaterialTheme.colorScheme.onSurfaceVariant,
158                 )
159             }
160         }
161     }
162 }
163

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/StorageHomeScreenState.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.main.screens.storage
2
3  import androidx.compose.runtime.Immutable
4  import com.github.nullptroma.wallenc.ui.resources.UserNotification
5
6  @Immutable
7  data class StorageHomeScreenState(
8      val storageUuid: String = "",
9      val storageName: String = "",
10     val isLoading: Boolean = true,
11     val isAvailable: Boolean = false,
12     val isEncrypted: Boolean = false,
13     val isVirtualStorage: Boolean = false,
14     val twoFaCount: Int = 0,
15     val textSecretsCount: Int = 0,
16     val canManageDomainData: Boolean = false,
17     val errorNotification: UserNotification.TextRes? = null,
18 )
19
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/StorageHomeViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage
2
3 import androidx.lifecycle.SavedStateHandle
4 import androidx.lifecycle.ViewModelScope
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageMetaLoadState
6 import com.github.nullptroma.wallenc.domain.errors.WallencException
7 import com.github.nullptroma.wallenc.ui.R
8 import com.github.nullptroma.wallenc.ui.ViewModelBase
9 import com.github.nullptroma.wallenc.ui.resources.UserNotification
10 import com.github.nullptroma.wallenc.ui.resources.toUserNotification
11 import com.github.nullptroma.wallenc.usecases.FindStorageUseCase
12 import com.github.nullptroma.wallenc.usecases.ManageTextSecretsUseCase
13 import com.github.nullptroma.wallenc.usecases.ManageTwoFaTokensUseCase
14 import dagger.hilt.android.lifecycle.HiltViewModel
15 import kotlinx.coroutines.flow.combine
16 import kotlinx.coroutines.launch
17 import javax.inject.Inject
18
19 @HiltViewModel
20 class StorageHomeViewModel @Inject constructor(
21     savedStateHandle: SavedStateHandle,
22     private val findStorageUseCase: FindStorageUseCase,
23     private val manageTwoFaTokensUseCase: ManageTwoFaTokensUseCase,
24     private val manageTextSecretsUseCase: ManageTextSecretsUseCase,
25 ) : ViewModelBase<StorageHomeScreenState>(StorageHomeScreenState()) {
26
27     private val storageUuid = savedStateHandle.requireStorageUuid()
28
29     init {
30         observeStorageState()
31     }
32
33     private fun observeStorageState() {
34         viewModelScope.launch {
35             val storage = findStorageUseCase.find(storageUuid)
36             if (storage == null) {
37                 updateState(
38                     state.value.copy(
39                         isLoading = false,
40                         storageUuid = storageUuid.toString(),
41                         errorNotification =
42                             WallencException.Feature.StorageNotFound().toUserNotification(),

```

```

42         ),
43     )
44     return@launch
45 }
46 combine(
47     storage.isAvailable,
48     storage.metaInfo,
49     storage.metaLoadState,
50     manageTwoFaTokensUseCase.observe(storage),
51     manageTextSecretsUseCase.observe(storage),
52 ) { available, meta, metaState, twoFa, secrets ->
53     val metaUnavailable = metaState == StorageMetaLoadState.Unavailable
54     val isRawEncrypted = meta.encInfo != null && !storage.isVirtualStorage
55     val canManageDomainData = available && !isRawEncrypted && !metaUnavailable
56     state.value.copy(
57         isLoading = false,
58         storageUuid = storage.uuid.toString(),
59         storageName = meta.name.orEmpty(),
60         isAvailable = available,
61         isEncrypted = meta.encInfo != null,
62         isVirtualStorage = storage.isVirtualStorage,
63         twoFaCount = twoFa.size,
64         textSecretsCount = secrets.size,
65         canManageDomainData = canManageDomainData,
66         errorNotification = when {
67             metaUnavailable ->
68                 UserNotification.TextRes(R.string.storage_home_meta_unavailable)
69             isRawEncrypted ->
70                 WallencException.Feature.NeedsDecryptedView().toUserNotification()
71             else -> null
72         },
73     )
74     }.collect { ui ->
75         updateState(ui)
76     }
77 }
78

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/StorageRouteArgs.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage
2
3 import androidx.lifecycle.SavedStateHandle
4 import java.util.UUID
5
6 internal fun SavedStateHandle.requireStorageUuid(): UUID {
7     val raw = get<String>("storageUuid") ?: error("Missing storage UUID in navigation
8     arguments")
9     return UUID.fromString(raw)
10 }
11
12 internal fun SavedStateHandle.optionalSecretId(): String? = get<String>("secretId")
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretDetailsRoute.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3  import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4  import kotlinx.parcelize.Parcelize
5  import kotlinx.serialization.Serializable
6
7  @Serializable
8  @Parcelize
9  data class TextSecretDetailsRoute(
10     val storageUuid: String,
11     val secretId: String,
12 ) : ScreenRoute()
13
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretDetailsScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import android.content.ClipData
4 import androidx.compose.foundation.layout.Arrangement
5 import androidx.compose.foundation.layout.Column
6 import androidx.compose.foundation.layout.Row
7 import androidx.compose.foundation.layout.fillMaxSize
8 import androidx.compose.foundation.layout.fillMaxWidth
9 import androidx.compose.foundation.layout.padding
10 import androidx.compose.foundation.lazy.LazyColumn
11 import androidx.compose.foundation.lazy.items
12 import androidx.compose.material.icons.Icons
13 import androidx.compose.material.icons.filled.ContentCopy
14 import androidx.compose.material3.Button
15 import androidx.compose.material3.Card
16 import androidx.compose.material3.CardDefaults
17 import androidx.compose.material3.Icon
18 import androidx.compose.material3.IconButton
19 import androidx.compose.material3.LinearProgressIndicator
20 import androidx.compose.material3.MaterialTheme
21 import androidx.compose.material3.Text
22 import androidx.compose.material3.TextButton
23 import androidx.compose.runtime.Composable
24 import androidx.compose.runtime.getValue
25 import androidx.compose.runtime.rememberCoroutineScope
26 import androidx.compose.ui.Modifier
27 import androidx.compose.ui.platform.LocalClipboard
28 import androidx.compose.ui.platform.toClipEntry
29 import androidx.compose.ui.res.stringResource
30 import androidx.compose.ui.unit.dp
31 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
32 import androidx.lifecycle.compose.collectAsStateWithLifecycle
33 import com.github.nullptroma.wallenc.ui.R
34 import com.github.nullptroma.wallenc.ui.elements.WallencScreenContentPadding
35 import com.github.nullptroma.wallenc.ui.elements.WallencScreenScaffold
36 import com.github.nullptroma.wallenc.ui.resources.resolveText
37 import kotlinx.coroutines.launch
38
39 @Composable
40 fun TextSecretDetailsScreen(
41     modifier: Modifier = Modifier,
42     viewModel: TextSecretDetailsViewModel = hiltViewModel(),
```

```

43         onEdit: (String) -> Unit,
44         onDelete: () -> Unit,
45     ) {
46         val uiState by viewModel.state.collectAsStateWithLifecycle()
47         val secret = uiState.secret
48         val clipboard = LocalClipboard.current
49         val scope = rememberCoroutineScope()
50
51         WallencScreenScaffold(modifier = modifier) { innerPadding ->
52             WallencScreenContentPadding(innerPadding) {
53                 Column(
54                     modifier = Modifier.fillMaxSize(),
55                     verticalArrangement = Arrangement.spacedBy(10.dp),
56                 ) {
57                     uiState.errorNotification?.let { notification ->
58                         Text(
59                             text = notification.resolveText(),
60                             color = MaterialTheme.colorScheme.error,
61                         )
62                     }
63                     if (uiState.isMutating) {
64                         LinearProgressIndicator(modifier = Modifier.fillMaxWidth())
65                     }
66                     if (secret == null) return@Column
67
68                     Row(
69                         modifier = Modifier.fillMaxWidth(),
70                         horizontalArrangement = Arrangement.SpaceBetween,
71                     ) {
72                         Text(
73                             text = secret.title,
74                             style = MaterialTheme.typography.headlineSmall,
75                         )
76                         TextButton(
77                             onClick = { onEdit(secret.id) },
78                             enabled = uiState.isAvailable && !uiState.isMutating,
79                         ) {
80                             Text(stringResource(R.string.edit))
81                         }
82                     }
83
84                     LazyColumn(verticalArrangement = Arrangement.spacedBy(8.dp)) {
85                         items(secret.items) { item ->
86                             val clipboardFallbackLabel =
stringResource(R.string.text_secret_clipboard_fallback_label)

```



```

87         Card(
88             modifier = Modifier.fillMaxWidth(),
89             colors = CardDefaults.elevatedCardColors(
90                 containerColor = MaterialTheme.colorScheme.surfaceContainerHigh,
91             ),
92         ) {
93             Column(
94                 modifier = Modifier
95                     .fillMaxWidth()
96                     .padding(12.dp),
97                 verticalArrangement = Arrangement.spacedBy(4.dp),
98             ) {
99                 Text(
100                     text = item.label ?:
stringResource(R.string.text_secret_item_without_label),
101                     style = MaterialTheme.typography.labelMedium,
102                     color = MaterialTheme.colorScheme.onSurfaceVariant,
103                 )
104                 Row(
105                     modifier = Modifier.fillMaxWidth(),
106                     horizontalArrangement = Arrangement.spacedBy(8.dp),
107                 ) {
108                     Text(
109                         text = item.value,
110                         style = MaterialTheme.typography.bodyLarge,
111                         modifier = Modifier.weight(1f),
112                     )
113                     IconButton(
114                         onClick = {
115                             scope.launch {
116                                 val clipData = ClipData.newPlainText(
117                                     item.label ?: clipboardFallbackLabel,
118                                     item.value,
119                                 )
120                                 clipboard.setClipEntry(clipData.toClipEntry())
121                             }
122                         },
123                     ) {
124                         Icon(
125                             imageVector = Icons.Default.ContentCopy,
126                             contentDescription =
stringResource(R.string.text_secret_copy_value),
127                         )
128                     }
129                 }

```

```
130         }
131     }
132 }
133 }
134
135 Button(
136     onClick = { viewModel.delete(onDeleted) },
137     enabled = uiState.isAvailable && !uiState.isMutating,
138 ) {
139     Text(stringResource(R.string.remove))
140 }
141 }
142 }
143 }
144 }
145
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretDetailsScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.compose.runtime.Immutable
4 import com.github.nullptroma.wallenc.ui.resources.UserNotification
5 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretRecord
6
7 @Immutable
8 data class TextSecretDetailsScreenState(
9     val isLoading: Boolean = true,
10    val isAvailable: Boolean = false,
11    val isMutating: Boolean = false,
12    val secret: TextSecretRecord? = null,
13    val errorNotification: UserNotification.TextRes? = null,
14 )
15
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretDetailsViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.lifecycle.SavedStateHandle
4 import androidx.lifecycle.ViewModelScope
5 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
6 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
7 import com.github.nullptroma.wallenc.domain.tasks.VaultTaskStep
8 import com.github.nullptroma.wallenc.domain.tasks.TaskId
9 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
10 import com.github.nullptroma.wallenc.ui.R
11 import com.github.nullptroma.wallenc.domain.errors.WallencException
12 import com.github.nullptroma.wallenc.ui.ViewModelBase
13 import com.github.nullptroma.wallenc.ui.resources.toUserNotification
14 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
15 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.requireStorageUuid
16 import com.github.nullptroma.wallenc.usecases.FindStorageUseCase
17 import com.github.nullptroma.wallenc.usecases.ManageTextSecretsUseCase
18 import dagger.hilt.android.lifecycle.HiltViewModel
19 import kotlinx.coroutines.Dispatchers
20 import kotlinx.coroutines.flow.combine
21 import kotlinx.coroutines.flow.filterNotNull
22 import kotlinx.coroutines.flow.first
23 import kotlinx.coroutines.flow.map
24 import kotlinx.coroutines.launch
25 import javax.inject.Inject
26
27 @HiltViewModel
28 class TextSecretDetailsViewModel @Inject constructor(
29     savedStateHandle: SavedStateHandle,
30     private val findStorageUseCase: FindStorageUseCase,
31     private val manageTextSecretsUseCase: ManageTextSecretsUseCase,
32     private val taskOrchestrator: ITaskOrchestrator,
33     private val uiStrings: UiStringResolver,
34 ) : ViewModelBase<TextSecretDetailsScreenState>(TextSecretDetailsScreenState()) {
35
36     private val storageUuid = savedStateHandle.requireStorageUuid()
37     private val secretId: String = savedStateHandle.get<String>("secretId")
38         ?: error("Missing secret id")
39
40     init {
41         observeSecret()
42     }
```

```

43
44     private fun observeSecret() {
45         viewModelScope.launch {
46             val storage = findStorageUseCase.find(storageUuid)
47             if (storage == null) {
48                 updateState(
49                     state.value.copy(
50                         isLoading = false,
51                         errorNotification =
WallencException.Feature.StorageNotFound().toUserNotification(),
52                     ),
53                 )
54                 return@launch
55             }
56             if (storage.metaInfo.value.encInfo != null && !storage.isVirtualStorage) {
57                 updateState(
58                     state.value.copy(
59                         isLoading = false,
60                         isAvailable = false,
61                         errorNotification =
WallencException.Feature.NeedsDecryptedView().toUserNotification(),
62                     ),
63                 )
64                 return@launch
65             }
66             combine(
67                 storage.isAvailable,
68                 manageTextSecretsUseCase.observe(storage).map { list ->
69                     list.firstOrNull { it.id == secretId }
70                 },
71                 taskOrchestrator.pipelineState.map { pipe ->
72                     pipe.tasks.any { t ->
73                         t.busyStorageUuid == storage.uuid && isTaskActive(t.state)
74                     }
75                 },
76             ) { available, secret, isMutating ->
77                 state.value.copy(
78                     isLoading = false,
79                     isAvailable = available,
80                     isMutating = isMutating,
81                     secret = secret,
82                     errorNotification = if (secret == null) {
83                         WallencException.Feature.SecretNotFound().toUserNotification()
84                     } else {
85                         null

```

```

86         },
87     )
88     }.collect { ui ->
89         updateState(ui)
90     }
91 }
92 }
93
94 fun delete(onDeleted: () -> Unit) {
95     viewModelScope.launch {
96         val storage = findStorageUseCase.find(storageUuid) ?: return@launch
97         if (storage.metaInfo.value.encInfo != null && !storage.isVirtualStorage) {
98             updateState(
99                 state.value.copy(
100                     errorNotification =
WallencException.Feature.NeedsDecryptedView().toUserNotification(),
101                 ),
102             )
103             return@launch
104         }
105         val taskId = taskOrchestrator.enqueue(
106             title = uiStrings(R.string.task_title_delete_text_secret),
107             dispatcher = Dispatchers.IO,
108             busyStorageUuid = storage.uuid,
109             work = { ctx ->
110                 ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.DeleteTextSecret))
111                 manageTextSecretsUseCase.delete(storage, secretId)
112             },
113         )
114         if (awaitTaskTerminalState(taskId) is TaskRunState.Completed) {
115             onDeleted()
116         }
117     }
118 }
119
120 private fun isActive(state: TaskRunState): Boolean =
121     state is TaskRunState.Queued || state is TaskRunState.Running
122
123 private suspend fun awaitTaskTerminalState(taskId: TaskId): TaskRunState {
124     return taskOrchestrator.pipelineState
125         .map { pipe -> pipe.tasks.firstOrNull { it.id == taskId }?.state }
126         .filterNotNull()
127         .first { state ->
128             state is TaskRunState.Completed ||
129             state is TaskRunState.Cancelled ||

```

```
130             state is TaskRunState.Failed
131         }
132     }
133 }
134
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretEditRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4 import kotlinx.parcelize.Parcelize
5 import kotlinx.serialization.Serializable
6
7 @Serializable
8 @Parcelize
9 data class TextSecretEditRoute(
10     val storageUuid: String,
11     val secretId: String? = null,
12 ) : ScreenRoute()
13
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretEditScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Row
6 import androidx.compose.foundation.layout.fillMaxSize
7 import androidx.compose.foundation.layout.fillMaxWidth
8 import androidx.compose.foundation.layout.padding
9 import androidx.compose.foundation.lazy.LazyColumn
10 import androidx.compose.foundation.lazy.itemsIndexed
11 import androidx.compose.material.icons.Icons
12 import androidx.compose.material.icons.filled.Add
13 import androidx.compose.material.icons.filled.Delete
14 import androidx.compose.material3.Icon
15 import androidx.compose.material3.IconButton
16 import androidx.compose.material3.LinearProgressIndicator
17 import androidx.compose.material3.OutlinedTextField
18 import androidx.compose.material3.Text
19 import androidx.compose.material3.TextButton
20 import androidx.compose.runtime.Composable
21 import androidx.compose.runtime.getValue
22 import androidx.compose.runtime.mutableStateListOf
23 import androidx.compose.runtime.mutableStateOf
24 import androidx.compose.runtime.remember
25 import androidx.compose.runtime.rememberUpdatedState
26 import androidx.compose.runtime.setValue
27 import androidx.compose.ui.Modifier
28 import androidx.compose.ui.res.stringResource
29 import androidx.compose.ui.unit.dp
30 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
31 import androidx.lifecycle.compose.collectAsStateWithLifecycle
32 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretEntryRecord
33 import com.github.nullptroma.wallenc.ui.R
34 import com.github.nullptroma.wallenc.ui.elements.WallencScreenContentPadding
35 import com.github.nullptroma.wallenc.ui.elements.WallencScreenScaffold
36 import com.github.nullptroma.wallenc.ui.resources.resolveText
37
38 @Composable
39 fun TextSecretEditScreen(
40     modifier: Modifier = Modifier,
41     viewModel: TextSecretEditViewModel = hiltViewModel(),
42     onSave: (String) -> Unit,
```

```

43     ) {
44         val uiState by viewModel.state.collectAsStateWithLifecycle()
45         val currentOnSaved by rememberUpdatedState(onSaved)
46         val inputEnabled = uiState.isAvailable && !uiState.isMutating &&
47         uiState.errorNotification == null
48
49         var title by remember(uiState.initialSecret) {
50             mutableStateOf(uiState.initialSecret?.title.orEmpty())
51         }
52         val items = remember(uiState.initialSecret) {
53             mutableStateListOf<TextSecretEntryRecord>().apply {
54                 addAll(
55                     uiState.initialSecret?.items.orEmpty().ifEmpty {
56                         listOf(TextSecretEntryRecord(label = null, value = ""))
57                     },
58                 )
59             }
60
61         WallencScreenScaffold(modifier = modifier) { innerPadding ->
62             WallencScreenContentPadding(innerPadding) {
63                 Column(
64                     modifier = Modifier.fillMaxSize(),
65                     verticalArrangement = Arrangement.spacedBy(10.dp),
66                 ) {
67                     Text(
68                         if (uiState.initialSecret == null) {
69                             stringResource(R.string.text_secret_create)
70                         } else {
71                             stringResource(R.string.text_secret_edit)
72                         },
73                     )
74                     uiState.errorNotification?.let { notification ->
75                         Text(text = notification.resolveText())
76                     }
77                     if (uiState.isMutating) {
78                         LinearProgressIndicator(modifier = Modifier.fillMaxWidth())
79                     }
80                     OutlinedTextField(
81                         value = title,
82                         onValueChange = { title = it },
83                         enabled = inputEnabled,
84                         modifier = Modifier.fillMaxWidth(),
85                         label = { Text(stringResource(R.string.text_secret_title)) },
86                 )

```

```

87
88         LazyColumn(
89             modifier = Modifier.weight(1f),
90             verticalArrangement = Arrangement.spacedBy(8.dp),
91         ) {
92             itemsIndexed(items) { index, item ->
93                 Row(
94                     modifier = Modifier.fillMaxWidth(),
95                     horizontalArrangement = Arrangement.spacedBy(8.dp),
96                 ) {
97                     OutlinedTextField(
98                         value = item.label.orEmpty(),
99                         onChange = { newLabel ->
100                             items[index] = item.copy(label = newLabel.ifBlank { null })
101                         },
102                         enabled = inputEnabled,
103                         modifier = Modifier.weight(0.45f),
104                         label =
105 { Text(stringResource(R.string.text_secret_item_label_optional)) },
106                     )
107                     OutlinedTextField(
108                         value = item.value,
109                         onChange = { newValue ->
110                             items[index] = item.copy(value = newValue)
111                         },
112                         enabled = inputEnabled,
113                         modifier = Modifier.weight(0.55f),
114                         label =
115 { Text(stringResource(R.string.text_secret_item_value)) },
116                     )
117                     IconButton(
118                         enabled = inputEnabled,
119                         onClick = {
120                             if (items.size > 1) {
121                                 items.removeAt(index)
122                             } else {
123                                 items[index] = TextSecretEntryRecord(label = null, value
124 = "")
125                             }
126                         },
127                     ) {
128                         Icon(Icons.Default.Delete, contentDescription =
stringResource(R.string.remove))
129                     }
130                 }
131             }
132         }

```

```

129         }
130
131         Row(horizontalArrangement = Arrangement.spacedBy(8.dp)) {
132             TextButton(
133                 onClick = { items.add(TextSecretEntryRecord(label = null, value =
134                 enabled = inputEnabled,
135             ) {
136                 Icon(Icons.Default.Add, contentDescription = null)
137                 Text(stringResource(R.string.text_secret_add_item))
138             }
139             TextButton(
140                 onClick = {
141                     viewModel.save(
142                         title = title,
143                         items = items.toList(),
144                         onSave = currentOnSaved,
145                     )
146                 },
147                 enabled = title.isNotBlank() && inputEnabled,
148             ) {
149                 Text(stringResource(R.string.save))
150             }
151         }
152     }
153 }
154 }
155 }
156

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretEditScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.compose.runtime.Immutable
4 import com.github.nullptroma.wallenc.ui.resources.UserNotification
5 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretRecord
6
7 @Immutable
8 data class TextSecretEditScreenState(
9     val isLoading: Boolean = true,
10    val isAvailable: Boolean = false,
11    val isMutating: Boolean = false,
12    val initialSecret: TextSecretRecord? = null,
13    val errorNotification: UserNotification.TextRes? = null,
14 )
15
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretEditViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.lifecycle.SavedStateHandle
4 import androidx.lifecycle.ViewModelScope
5 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretEntryRecord
6 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretRecord
7 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
8 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
9 import com.github.nullptroma.wallenc.domain.tasks.VaultTaskStep
10 import com.github.nullptroma.wallenc.domain.tasks.TaskId
11 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
12 import com.github.nullptroma.wallenc.ui.R
13 import com.github.nullptroma.wallenc.domain.errors.WallencException
14 import com.github.nullptroma.wallenc.ui.ViewModelBase
15 import com.github.nullptroma.wallenc.ui.resources.toUserNotification
16 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
17 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.optionalSecretId
18 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.requireStorageUuid
19 import com.github.nullptroma.wallenc.usecases.FindStorageUseCase
20 import com.github.nullptroma.wallenc.usecases.ManageTextSecretsUseCase
21 import dagger.hilt.android.lifecycle.HiltViewModel
22 import kotlinx.coroutines.Dispatchers
23 import kotlinx.coroutines.flow.combine
24 import kotlinx.coroutines.flow.filterNotNull
25 import kotlinx.coroutines.flow.first
26 import kotlinx.coroutines.flow.flowOf
27 import kotlinx.coroutines.flow.map
28 import kotlinx.coroutines.launch
29 import java.util.UUID
30 import javax.inject.Inject
31
32 @HiltViewModel
33 class TextSecretEditViewModel @Inject constructor(
34     savedStateHandle: SavedStateHandle,
35     private val findStorageUseCase: FindStorageUseCase,
36     private val manageTextSecretsUseCase: ManageTextSecretsUseCase,
37     private val taskOrchestrator: ITaskOrchestrator,
38     private val uiStrings: UiStringResolver,
39 ) : ViewModelBase<TextSecretEditScreenState>(TextSecretEditScreenState()) {
40
41     private val storageUuid = savedStateHandle.requireStorageUuid()
42     private val secretId: String? = savedStateHandle.optionalSecretId()
```

```

43
44     init {
45         load()
46     }
47
48     private fun load() {
49         viewModelScope.launch {
50             val storage = findStorageUseCase.find(storageUuid)
51             if (storage == null) {
52                 updateState(
53                     state.value.copy(
54                         isLoading = false,
55                         errorNotification =
WallencException.Feature.StorageNotFound().toUserNotification(),
56                     ),
57                 )
58                 return@launch
59             }
60             if (storage.metaInfo.value.encInfo != null && !storage.isVirtualStorage) {
61                 updateState(
62                     state.value.copy(
63                         isLoading = false,
64                         isAvailable = false,
65                         errorNotification =
WallencException.Feature.NeedsDecryptedView().toUserNotification(),
66                     ),
67                 )
68                 return@launch
69             }
70             val initial = secretId?.let { id -> manageTextSecretsUseCase.get(storage, id) }
71             combine(
72                 storage.isAvailable,
73                 taskOrchestrator.pipelineState.map { pipe ->
74                     pipe.tasks.any { t ->
75                         t.busyStorageUuid == storage.uuid && isTaskActive(t.state)
76                     }
77                 },
78                 flowOf(initial),
79             ) { available, isMutating, currentSecret ->
80                 state.value.copy(
81                     isLoading = false,
82                     isAvailable = available,
83                     isMutating = isMutating,
84                     initialSecret = currentSecret,
85                     errorNotification = null,

```

```

86         )
87         }.collect { ui ->
88             updateState(ui)
89         }
90     }
91 }
92
93 fun save(
94     title: String,
95     items: List<TextSecretEntryRecord>,
96     onSave: (secretId: String) -> Unit,
97 ) {
98     viewModelScope.launch {
99         val storage = findStorageUseCase.find(storageUuid) ?: return@launch
100         if (storage.metaInfo.value.encInfo != null && !storage.isVirtualStorage) {
101             updateState(
102                 state.value.copy(
103                     errorNotification =
WallencException.Feature.NeedsDecryptedView().toUserNotification(),
104                 ),
105             )
106             return@launch
107         }
108         val existingId = secretId
109         val targetSecretId = existingId ?: UUID.randomUUID().toString()
110         val taskId = taskOrchestrator.enqueue(
111             title = uiStrings(R.string.task_title_save_text_secret),
112             dispatcher = Dispatchers.IO,
113             busyStorageUuid = storage.uuid,
114             work = { ctx ->
115                 ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.SaveTextSecret))
116                 if (existingId == null) {
117                     manageTextSecretsUseCase.create(
118                         storageInfo = storage,
119                         title = title,
120                         items = items,
121                         id = targetSecretId,
122                     )
123                 } else {
124                     manageTextSecretsUseCase.update(
125                         storageInfo = storage,
126                         secret = TextSecretRecord(
127                             id = targetSecretId,
128                             title = title,
129                             items = items,

```



```

130         ),
131     )
132 }
133 },
134 )
135 if (awaitTaskTerminalState(taskId) is TaskRunState.Completed) {
136     onSave(targetSecretId)
137 }
138 }
139 }
140
141 private fun isTaskActive(state: TaskRunState): Boolean =
142     state is TaskRunState.Queued || state is TaskRunState.Running
143
144 private suspend fun awaitTaskTerminalState(taskId: TaskId): TaskRunState {
145     return taskOrchestrator.pipelineState
146         .map { pipe -> pipe.tasks.firstOrNull { it.id == taskId }?.state }
147         .filterNotNull()
148         .first { state ->
149             state is TaskRunState.Completed ||
150             state is TaskRunState.Cancelled ||
151             state is TaskRunState.Failed
152         }
153 }
154 }
155

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretsRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4 import kotlinx.parcelize.Parcelize
5 import kotlinx.serialization.Serializable
6
7 @Serializable
8 @Parcelize
9 data class TextSecretsRoute(
10     val storageUuid: String,
11 ) : ScreenRoute()
12
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretsScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.fillMaxSize
5 import androidx.compose.foundation.layout.padding
6 import androidx.compose.foundation.lazy.LazyColumn
7 import androidx.compose.foundation.lazy.items
8 import androidx.compose.material.icons.Icons
9 import androidx.compose.material.icons.filled.Add
10 import androidx.compose.material3.FloatingActionButton
11 import androidx.compose.material3.Icon
12 import androidx.compose.runtime.Composable
13 import androidx.compose.runtime.getValue
14 import androidx.compose.ui.Modifier
15 import androidx.compose.ui.draw.alpha
16 import androidx.compose.ui.res.stringResource
17 import androidx.compose.ui.unit.dp
18 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
19 import androidx.lifecycle.compose.collectAsStateWithLifecycle
20 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretRecord
21 import com.github.nullptroma.wallenc.ui.R
22 import com.github.nullptroma.wallenc.ui.elements.WallencScreenContentPadding
23 import com.github.nullptroma.wallenc.ui.elements.WallencScreenScaffold
24
25 @Composable
26 fun TextSecretsScreen(
27     modifier: Modifier = Modifier,
28     viewModel: TextSecretsViewModel = hiltViewModel(),
29     onOpenSecret: (TextSecretRecord) -> Unit,
30     onCreateSecret: () -> Unit,
31 ) {
32     val uiState by viewModel.state.collectAsStateWithLifecycle()
33
34     WallencScreenScaffold(
35         modifier = modifier,
36         floatingActionButton = {
37             FloatingActionButton(
38                 onClick = {
39                     if (uiState.isAvailable) {
40                         onCreateSecret()
41                     }
42                 },
```

```

43         modifier = Modifier.alpha(if (uiState.isAvailable) 1f else 0.5f),
44     ) {
45         Icon(Icons.Default.Add, contentDescription =
stringResource(R.string.text_secret_create))
46     },
47 },
48 ) { innerPadding ->
49     WallencScreenContentPadding(innerPadding) {
50         TextSecretsScreenContent(
51             uiState = uiState,
52             modifier = Modifier.fillMaxSize(),
53         ) {
54             LazyColumn(verticalArrangement = Arrangement.spacedBy(10.dp)) {
55                 items(uiState.items) { secret ->
56                     TextSecretListCard(
57                         secret = secret,
58                         onClick = { onOpenSecret(secret) },
59                         enabled = uiState.isAvailable,
60                     )
61                 }
62             }
63         }
64     }
65 }
66 }
67

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretsScreenContent.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Row
6 import androidx.compose.foundation.layout.fillMaxSize
7 import androidx.compose.foundation.layout.fillMaxWidth
8 import androidx.compose.foundation.layout.padding
9 import androidx.compose.foundation.layout.size
10 import androidx.compose.material.icons.Icons
11 import androidx.compose.material.icons.automirrored.filled.KeyboardArrowRight
12 import androidx.compose.material.icons.automirrored.outlined.Notes
13 import androidx.compose.material3.Card
14 import androidx.compose.material3.CardDefaults
15 import androidx.compose.material3.Icon
16 import androidx.compose.material3.MaterialTheme
17 import androidx.compose.material3.Text
18 import androidx.compose.runtime.Composable
19 import androidx.compose.ui.Alignment
20 import androidx.compose.ui.Modifier
21 import androidx.compose.ui.res.stringResource
22 import androidx.compose.ui.unit.dp
23 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretRecord
24 import com.github.nullptroma.wallenc.ui.R
25 import com.github.nullptroma.wallenc.ui.resources.resolveText
26
27 @Composable
28 fun TextSecretsScreenContent(
29     uiState: TextSecretsScreenState,
30     modifier: Modifier = Modifier,
31     secretList: @Composable () -> Unit = {},
32 ) {
33     Column(
34         modifier = modifier
35             .fillMaxSize()
36             .padding(16.dp),
37         verticalArrangement = Arrangement.spacedBy(12.dp),
38     ) {
39         uiState.errorNotification?.let { notification ->
40             Text(
41                 text = notification.resolveText(),
42                 color = MaterialTheme.colorScheme.error,
```

```

43         )
44     }
45
46     if (uiState.items.isEmpty()) {
47         Text(stringResource(R.string.text_secret_empty_state))
48     } else {
49         secretList()
50     }
51 }
52 }
53
54 @Composable
55 fun TextSecretListCard(
56     secret: TextSecretRecord,
57     onClick: () -> Unit,
58     enabled: Boolean,
59     modifier: Modifier = Modifier,
60 ) {
61     Card(
62         modifier = modifier.fillMaxWidth(),
63         onClick = onClick,
64         enabled = enabled,
65         colors = CardDefaults.elevatedCardColors(
66             containerColor = MaterialTheme.colorScheme.surfaceContainerHigh,
67         ),
68     ) {
69         Row(
70             modifier = Modifier
71                 .fillMaxWidth()
72                 .padding(14.dp),
73             horizontalArrangement = Arrangement.SpaceBetween,
74             verticalAlignment = Alignment.Top,
75         ) {
76             Row(
77                 modifier = Modifier.weight(1f),
78                 horizontalArrangement = Arrangement.spacedBy(12.dp),
79             ) {
80                 Icon(
81                     imageVector = Icons.AutoMirrored.Outlined.Notes,
82                     contentDescription = null,
83                     modifier = Modifier
84                         .size(22.dp)
85                         .padding(top = 2.dp),
86                     tint = MaterialTheme.colorScheme.primary,
87                 )

```

```

88         Column(modifier = Modifier.padding(end = 12.dp)) {
89             Text(
90                 text = secret.title,
91                 style = MaterialTheme.typography.titleSmall,
92             )
93             Text(
94                 text = stringResource(R.string.text_secret_items_count,
secret.items.size),
95                 style = MaterialTheme.typography.bodySmall,
96                 color = MaterialTheme.colorScheme.onSurfaceVariant,
97             )
98             val fieldNames = secret.items
99                 .map { item ->
100                     item.label
101                         ?.trim()
102                         .takeUnless { it.isNullOrBlank() }
103                         ?: stringResource(R.string.text_secret_item_without_label)
104                 }
105             if (fieldNames.isNotEmpty()) {
106                 fieldNames.take(3).forEach { fieldName ->
107                     Text(
108                         text = "\u2022 $fieldName",
109                         style = MaterialTheme.typography.bodySmall,
110                         color = MaterialTheme.colorScheme.onSurfaceVariant,
111                     )
112                 }
113                 val hiddenCount = fieldNames.size - 3
114                 if (hiddenCount > 0) {
115                     Text(
116                         text = stringResource(
117                             R.string.text_secret_more_fields,
118                             hiddenCount,
119                         ),
120                         style = MaterialTheme.typography.bodySmall,
121                         color = MaterialTheme.colorScheme.onSurfaceVariant,
122                     )
123                 }
124             }
125         }
126     }
127     Icon(
128         imageVector = Icons.AutoMirrored.Filled.KeyboardArrowRight,
129         contentDescription = null,
130     )
131 }

```

132 }
133 }
134

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretsScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.compose.runtime.Immutable
4 import com.github.nullptroma.wallenc.ui.resources.UserNotification
5 import com.github.nullptroma.wallenc.domain.datatypes.TextSecretRecord
6
7 @Immutable
8 data class TextSecretsScreenState(
9     val isLoading: Boolean = true,
10    val isAvailable: Boolean = false,
11    val items: List<TextSecretRecord> = emptyList(),
12    val errorNotification: UserNotification.TextRes? = null,
13 )
14
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/secrets/TextSecretsViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets
2
3 import androidx.lifecycle.SavedStateHandle
4 import androidx.lifecycle.ViewModelScope
5 import com.github.nullptroma.wallenc.domain.errors.WallencException
6 import com.github.nullptroma.wallenc.ui.ViewModelBase
7 import com.github.nullptroma.wallenc.ui.resources.toUserNotification
8 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.requireStorageUuid
9 import com.github.nullptroma.wallenc.usecases.FindStorageUseCase
10 import com.github.nullptroma.wallenc.usecases.ManageTextSecretsUseCase
11 import dagger.hilt.android.lifecycle.HiltViewModel
12 import kotlinx.coroutines.flow.combine
13 import kotlinx.coroutines.launch
14 import javax.inject.Inject
15
16 @HiltViewModel
17 class TextSecretsViewModel @Inject constructor(
18     savedStateHandle: SavedStateHandle,
19     private val findStorageUseCase: FindStorageUseCase,
20     private val manageTextSecretsUseCase: ManageTextSecretsUseCase,
21 ) : ViewModelBase<TextSecretsScreenState>(TextSecretsScreenState()) {
22
23     private val storageUuid = savedStateHandle.requireStorageUuid()
24
25     init {
26         observeSecrets()
27     }
28
29     private fun observeSecrets() {
30         viewModelScope.launch {
31             val storage = findStorageUseCase.find(storageUuid)
32             if (storage == null) {
33                 updateState(
34                     state.value.copy(
35                         isLoading = false,
36                         errorNotification =
37                         WallencException.Feature.StorageNotFound().toUserNotification(),
38                     ),
39                     return@launch
40                 )
41             if (storage.metaInfo.value.encInfo != null && !storage.isVirtualStorage) {
```

```

42         updateState(
43             state.value.copy(
44                 isLoading = false,
45                 isAvailable = false,
46                 errorNotification =
WallencException.Feature.NeedsDecryptedView().toUserNotification(),
47             ),
48         )
49         return@launch
50     }
51     combine(
52         storage.isAvailable,
53         manageTextSecretsUseCase.observe(storage),
54     ) { available, items ->
55         state.value.copy(
56             isLoading = false,
57             isAvailable = available,
58             items = items,
59             errorNotification = null,
60         )
61     }.collect { ui ->
62         updateState(ui)
63     }
64 }
65 }
66 }
67

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/twofa/OtpAuthUriParser.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa
2
3 import androidx.core.net.toUri
4
5 data class ParsedOtpAuthToken(
6     val issuer: String,
7     val account: String,
8     val secret: String,
9     val digits: Int,
10    val periodSeconds: Int,
11    val algorithm: String,
12 )
13
14 fun parseOtpAuthTotpUri(raw: String): ParsedOtpAuthToken? {
15     val uri = runCatching { raw.trim().toUri() }.getOrNull() ?: return null
16     if (!uri.scheme.equals("otpauth", ignoreCase = true)) return null
17     if (!uri.host.equals("totp", ignoreCase = true)) return null
18
19     val secret = uri.getQueryParameter("secret")?.trim().orEmpty()
20     if (secret.isBlank()) return null
21
22     val label = uri.path.orEmpty().trim('/').trim()
23     val queryIssuer = uri.getQueryParameter("issuer")?.trim().orEmpty()
24     val (issuerFromLabel, accountFromLabel) = splitLabel(label)
25     val issuer = queryIssuer.ifBlank { issuerFromLabel }.ifBlank { "TOTP" }
26     val account = accountFromLabel.ifBlank { label.ifBlank { "account" } }
27     val digits = uri.getQueryParameter("digits")?.toIntOrNull()?.coerceIn(6, 8) ?: 6
28     val periodSeconds = uri.getQueryParameter("period")?.toIntOrNull()?.coerceAtLeast(1) ?:
30
29     val algorithm = normalizeAlgorithm(uri.getQueryParameter("algorithm") ?: "SHA1")
30
31     return ParsedOtpAuthToken(
32         issuer = issuer,
33         account = account,
34         secret = secret,
35         digits = digits,
36         periodSeconds = periodSeconds,
37         algorithm = algorithm,
38     )
39 }
40
41 private fun splitLabel(label: String): Pair<String, String> {
```

```

42     val idx = label.indexOf(':')
43     return if (idx >= 0) {
44         label.substring(0, idx).trim() to label.substring(idx + 1).trim()
45     } else {
46         "" to label.trim()
47     }
48 }
49
50 private fun normalizeAlgorithm(input: String): String {
51     val normalized = input
52         .trim()
53         .uppercase()
54         .replace("HMAC", "")
55         .replace("-", "")
56         .replace("_", "")
57     return when (normalized) {
58         "SHA1", "SHA256", "SHA512" -> normalized
59         else -> "SHA1"
60     }
61 }
62

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/twofa/TwoFaTokensRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa
2
3 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4 import kotlinx.parcelize.Parcelize
5 import kotlinx.serialization.Serializable
6
7 @Serializable
8 @Parcelize
9 data class TwoFaTokensRoute(
10     val storageUuid: String,
11 ) : ScreenRoute()
12
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/twofa/TwoFaTokensScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa
2
3 import android.Manifest
4 import android.content.ClipData
5 import android.content.pm.PackageManager
6 import androidx.activity.compose.rememberLauncherForActivityResult
7 import androidx.activity.result.contract.ActivityResultContracts
8 import androidx.compose.foundation.background
9 import androidx.compose.foundation.layout.Arrangement
10 import androidx.compose.foundation.layout.Box
11 import androidx.compose.foundation.layout.Column
12 import androidx.compose.foundation.layout.Row
13 import androidx.compose.foundation.layout.WindowInsets
14 import androidx.compose.foundation.layout.fillMaxHeight
15 import androidx.compose.foundation.layout.fillMaxSize
16 import androidx.compose.foundation.layout.fillMaxWidth
17 import androidx.compose.foundation.layout.heightIn
18 import androidx.compose.foundation.layout.offset
19 import androidx.compose.foundation.layout.padding
20 import androidx.compose.foundation.layout.size
21 import androidx.compose.foundation.lazy.LazyColumn
22 import androidx.compose.foundation.lazy.items
23 import androidx.compose.foundation.rememberScrollState
24 import androidx.compose.foundation.verticalScroll
25 import androidx.compose.material.icons.Icons
26 import androidx.compose.material.icons.filled.Add
27 import androidx.compose.material.icons.filled.ContentCopy
28 import androidx.compose.material.icons.filled.Delete
29 import androidx.compose.material.icons.filled.Edit
30 import androidx.compose.material.icons.filled.QrCodeScanner
31 import androidx.compose.material3.AlertDialog
32 import androidx.compose.material3.Card
33 import androidx.compose.material3.CardDefaults
34 import androidx.compose.material3.DropdownMenu
35 import androidx.compose.material3.DropdownMenuItem
36 import androidx.compose.material3.ExperimentalMaterial3Api
37 import androidx.compose.material3.ExposedDropdownMenuAnchorType
38 import androidx.compose.material3.ExposedDropdownMenuBox
39 import androidx.compose.material3.ExposedDropdownMenuDefaults
40 import androidx.compose.material3.FloatingActionButton
41 import androidx.compose.material3.Icon
42 import androidx.compose.material3.IconButton
```

```

43     import androidx.compose.material3.LinearProgressIndicator
44     import androidx.compose.material3.MaterialTheme
45     import androidx.compose.material3.OutlinedTextField
46     import androidx.compose.material3.Scaffold
47     import androidx.compose.material3.Slider
48     import androidx.compose.material3.Text
49     import androidx.compose.material3.TextButton
50     import androidx.compose.runtime.Composable
51     import androidx.compose.runtime.LaunchedEffect
52     import androidx.compose.runtime.getValue
53     import androidx.compose.runtime.mutableFloatStateOf
54     import androidx.compose.runtime.mutableIntStateOf
55     import androidx.compose.runtime.mutableStateOf
56     import androidx.compose.runtime.produceState
57     import androidx.compose.runtime.remember
58     import androidx.compose.runtime.withFrameMillis
59     import androidx.compose.runtime.rememberCoroutineScope
60     import androidx.compose.runtime.setValue
61     import androidx.compose.runtime.snapshotFlow
62     import androidx.compose.ui.Alignment
63     import androidx.compose.ui.Modifier
64     import androidx.compose.ui.draw.alpha
65     import androidx.compose.ui.graphics.Color
66     import androidx.compose.ui.layout.onSizeChanged
67     import androidx.compose.ui.platform.LocalClipboard
68     import androidx.compose.ui.platform.LocalContext
69     import androidx.compose.ui.platform.LocalDensity
70     import androidx.compose.ui.platform.toClipEntry
71     import androidx.compose.ui.res.stringResource
72     import androidx.compose.ui.text.font.FontFamily
73     import androidx.compose.ui.text.style.TextAlign
74     import androidx.compose.ui.unit.IntOffset
75     import androidx.compose.ui.unit.dp
76     import androidx.compose.ui.unit.sp
77     import androidx.core.content.ContextCompat
78     import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
79     import androidx.lifecycle.compose.collectAsStateWithLifecycle
80     import com.github.nullptroma.wallenc.domain.datatypes.TwoFaTokenRecord
81     import com.github.nullptroma.wallenc.ui.R
82     import com.github.nullptroma.wallenc.ui.elements.QrScannerDialog
83     import com.github.nullptroma.wallenc.ui.elements.WallencScreenContentPadding
84     import com.github.nullptroma.wallenc.ui.elements.WallencScreenScaffold
85     import com.github.nullptroma.wallenc.usecases.TwoFaCodeState
86     import com.github.nullptroma.wallenc.usecases.buildTwoFaCodeState

```



```

87 import com.github.nullptroma.wallenc.usecases.totpPeriodProgress
88 import com.github.nullptroma.wallenc.usecases.totpSecondsUntilRefresh
89 import kotlinx.coroutines.flow.collectLatest
90 import kotlinx.coroutines.launch
91 import kotlin.math.roundToInt
92
93 @Composable
94 fun TwoFaTokensScreen(
95     modifier: Modifier = Modifier,
96     viewModel: TwoFaTokensViewModel = hiltViewModel(),
97 ) {
98     val uiState by viewModel.state.collectAsStateWithLifecycle()
99     val clipboard = LocalClipboard.current
100    val scope = rememberCoroutineScope()
101    val nowMillis by produceState(initialValue = System.currentTimeMillis()) {
102        while (true) {
103            withFrameMillis {
104                value = System.currentTimeMillis()
105            }
106        }
107    }
108    var editingToken by remember { mutableStateOf<TwoFaTokenRecord?>(null) }
109    var creating by remember { mutableStateOf(false) }
110
111    WallencScreenScaffold(
112        modifier = modifier,
113        floatingActionButton = {
114            FloatingActionButton(
115                onClick = {
116                    if (uiState.isAvailable && !uiState.isMutating) {
117                        creating = true
118                    }
119                },
120                modifier = Modifier.alpha(if (uiState.isAvailable && !uiState.isMutating) 1f
else 0.5f),
121            ) {
122                Icon(Icons.Default.Add, contentDescription =
stringResource(R.string.two_fa_add_token))
123            }
124        },
125    ) { innerPadding ->
126        WallencScreenContentPadding(innerPadding) {
127            TwoFaTokensScreenContent(
128                uiState = uiState,
129                modifier = Modifier.fillMaxSize(),
130            ) {

```

```

131         LazyColumn(verticalArrangement = Arrangement.spacedBy(10.dp)) {
132             items(uiState.items) { item ->
133                 Card(
134                     modifier = Modifier.fillMaxWidth(),
135                     colors = CardDefaults.elevatedCardColors(
136                         containerColor =
MaterialTheme.colorScheme.surfaceContainerHigh,
137                     ),
138                 ) {
139                     val codeState = rememberTwoFaCode(item, nowMillis)
140                     Row(
141                         modifier = Modifier
142                             .fillMaxWidth()
143                             .padding(14.dp),
144                         horizontalArrangement = Arrangement.SpaceBetween,
145                     ) {
146                         TwoFaTokenListHeader(
147                             issuer = item.issuer,
148                             account = item.account,
149                             modifier = Modifier.weight(1f),
150                         )
151                         Row {
152                             IconButton(
153                                 onClick = { editingToken = item },
154                                 enabled = uiState.isAvailable && !
uiState.isMutating,
155                             ) {
156                                 Icon(Icons.Default.Edit, contentDescription =
stringResource(R.string.edit))
157                             }
158                             IconButton(
159                                 onClick = { viewModel.deleteToken(item.id) },
160                                 enabled = uiState.isAvailable && !
uiState.isMutating,
161                             ) {
162                                 Icon(Icons.Default.Delete, contentDescription =
stringResource(R.string.remove))
163                             }
164                         }
165                     }
166                     val codeProgress = if (codeState == null) {
167                         0f
168                     } else {
169                         totpPeriodProgress(nowMillis, item.periodSeconds)
170                     }
171                     val secondsUntilRefresh = codeState?.let {

```

```

172         totpSecondsUntilRefresh(nowMillis, item.periodSeconds)
173     }
174     Row(
175         modifier = Modifier
176             .fillMaxWidth()
177             .padding(start = 14.dp, end = 14.dp, bottom = 10.dp),
178         horizontalArrangement = Arrangement.SpaceBetween,
179         verticalAlignment = Alignment.Top,
180     ) {
181         Column(
182             verticalArrangement = Arrangement.spacedBy(2.dp),
183         ) {
184             Text(
185                 text =
stringResource(R.string.two_fa_code_refresh_label),
186                 style = MaterialTheme.typography.bodySmall,
187                 color = MaterialTheme.colorScheme.onSurfaceVariant,
188             )
189             Text(
190                 text = if (secondsUntilRefresh != null) {
191                     stringResource(
192                         R.string.two_fa_code_refresh_seconds,
193                         secondsUntilRefresh,
194                     )
195                 } else {
196                     stringResource(R.string.two_fa_code_invalid_secret)
197                 },
198                 style =
MaterialTheme.typography.bodySmall.copy(fontSize = 13.sp),
199                 color = if (codeState != null) {
200                     MaterialTheme.colorScheme.onSurfaceVariant
201                 } else {
202                     Color.Red
203                 },
204             )
205         }
206         Card(
207             onClick = {
208                 codeState?.let {
209                     scope.launch {
210                         val clipData =
ClipData.newPlainText(it.code, it.code)
211                         clipboard.setClipEntry(clipData.toClipEntry())
212                     }
213                 }

```

```

214         },
215         enabled = codeState != null,
216         colors = CardDefaults.cardColors(
217             MaterialTheme.colorScheme.surfaceContainer,
218             containerColor =
219         ),
220     ) {
221         Column(
222             modifier = Modifier
223                 .padding(horizontal = 16.dp, vertical = 10.dp),
224             verticalArrangement = Arrangement.spacedBy(4.dp),
225         ) {
226             Row(
227                 horizontalArrangement =
228                 Arrangement.spacedBy(8.dp),
229                 verticalAlignment = Alignment.CenterVertically,
230             ) {
231                 Icon(
232                     imageVector = Icons.Default.ContentCopy,
233                     contentDescription = null,
234                     tint =
235                     MaterialTheme.colorScheme.onSurfaceVariant,
236                 )
237                 Text(
238                     text = codeState?.code ?:
239                     stringResource(R.string.two_fa_code_unavailable),
240                     style = MaterialTheme.typography.titleLarge,
241                     fontFamily = FontFamily.Monospace,
242                     textAlign = TextAlign.End,
243                 )
244             }
245         }
246     }
247 }
248
249 LinearProgressIndicator(
250     progress = { codeProgress },
251     modifier = Modifier.fillMaxWidth(),
252 )
253 }
254
255 if (creating) {

```

```

256         TwoFaTokenEditDialog(
257             startValue = null,
258             isBusy = uiState.isMutating,
259             onDismiss = { creating = false },
260             onSave = { issuer, account, secret, notes, digits, periodSeconds, algorithm ->
261                 viewModel.saveToken(
262                     existingId = null,
263                     issuer = issuer,
264                     account = account,
265                     secret = secret,
266                     notes = notes,
267                     digits = digits,
268                     periodSeconds = periodSeconds,
269                     algorithm = algorithm,
270                 )
271                 creating = false
272             },
273         )
274     }
275
276     editingToken?.let { token ->
277         TwoFaTokenEditDialog(
278             startValue = token,
279             isBusy = uiState.isMutating,
280             onDismiss = { editingToken = null },
281             onSave = { issuer, account, secret, notes, digits, periodSeconds, algorithm ->
282                 viewModel.saveToken(
283                     existingId = token.id,
284                     issuer = issuer,
285                     account = account,
286                     secret = secret,
287                     notes = notes,
288                     digits = digits,
289                     periodSeconds = periodSeconds,
290                     algorithm = algorithm,
291                 )
292                 editingToken = null
293             },
294         )
295     }
296 }
297
298 @OptIn(ExperimentalMaterial3Api::class)
299 @Composable
300 private fun TwoFaTokenEditDialog(

```

```

301     startValue: TwoFaTokenRecord?,
302     isBusy: Boolean,
303     onDismiss: () -> Unit,
304     onSave: (String, String, String, String?, Int, Int, String) -> Unit,
305 ) {
306     val context = LocalContext.current
307     var issuer by remember(startValue) { mutableStateOf(startValue?.issuer.orEmpty()) }
308     var account by remember(startValue) { mutableStateOf(startValue?.account.orEmpty()) }
309     var secret by remember(startValue) { mutableStateOf(startValue?.secret.orEmpty()) }
310     var notes by remember(startValue) { mutableStateOf(startValue?.notes.orEmpty()) }
311     var digitsValue by remember(startValue) { mutableIntStateOf((startValue?.digits ?:
312 6).coerceIn(6, 8)) }
313     var periodSecondsValue by remember(startValue)
314 { mutableIntStateOf((startValue?.periodSeconds ?: 30).coerceIn(10, 120)) }
315     var algorithmValue by remember(startValue) { mutableStateOf((startValue?.algorithm ?:
316 "SHA1").uppercase()) }
317     var algorithmExpanded by remember { mutableStateOf(false) }
318     var showScanner by remember { mutableStateOf(false) }
319     var scanError by remember { mutableStateOf<String?>(null) }
320     var permissionDenied by remember { mutableStateOf(false) }
321     val dialogScrollState = rememberScrollState()
322     var scrollContainerHeightPx by remember { mutableIntStateOf(0) }
323     var scrollProgress by remember { mutableFloatStateOf(0f) }
324     val density = LocalDensity.current
325     val scanInvalidText = stringResource(R.string.two_fa_scan_qr_invalid)
326     val cameraPermissionLauncher = rememberLauncherForActivityResult(
327         contract = ActivityResultContracts.RequestPermission(),
328     ) { granted ->
329         if (granted) {
330             showScanner = true
331             permissionDenied = false
332         } else {
333             permissionDenied = true
334         }
335     }
336     LaunchedEffect(dialogScrollState) {
337         snapshotFlow { dialogScrollState.value to dialogScrollState.maxValue }
338         .collectLatest { (value, maxValue) ->
339             scrollProgress = if (maxValue == 0) 0f else value.toFloat() /
340             maxValue.toFloat()
341         }
342     }
343     AlertDialog(
344         onDismissRequest = { if (!isBusy) onDismiss() },
345         title = {

```

```

343         Text(
344             if (startValue == null) {
345                 stringResource(R.string.two_fa_create_title)
346             } else {
347                 stringResource(R.string.two_fa_edit_title)
348             },
349         )
350     },
351     text = {
352         Box(
353             modifier = Modifier
354                 .fillMaxWidth()
355                 .heightIn(max = 520.dp)
356                 .onSizeChanged { scrollContainerHeightPx = it.height },
357         ) {
358             Column(
359                 modifier = Modifier
360                     .fillMaxWidth()
361                     .padding(end = 10.dp)
362                     .verticalScroll(dialogScrollState),
363                 verticalArrangement = Arrangement.spacedBy(8.dp),
364             ) {
365                 if (isBusy) {
366                     LinearProgressIndicator(modifier = Modifier.fillMaxWidth())
367                 }
368                 OutlinedTextField(
369                     value = issuer,
370                     onValueChange = { issuer = it },
371                     enabled = !isBusy,
372                     label = { Text(stringResource(R.string.two_fa_field_issuer)) },
373                 )
374                 OutlinedTextField(
375                     value = account,
376                     onValueChange = { account = it },
377                     enabled = !isBusy,
378                     label = { Text(stringResource(R.string.two_fa_field_account)) },
379                 )
380                 OutlinedTextField(
381                     value = secret,
382                     onValueChange = { secret = it },
383                     enabled = !isBusy,
384                     label = { Text(stringResource(R.string.two_fa_field_secret)) },
385                 )
386                 Text(
387                     text = stringResource(R.string.two_fa_field_algorithm),

```

```

388         style = MaterialTheme.typography.labelMedium,
389     )
390     ExposedDropDownMenuBox(
391         expanded = algorithmExpanded,
392         onExpandedChange = { if (!isBusy) algorithmExpanded = !
algorithmExpanded },
393     ) {
394         val fillMaxWidth = Modifier
395             .fillMaxWidth()
396         OutlinedTextField(
397             value = algorithmValue,
398             onValueChange = {},
399             readOnly = true,
400             enabled = !isBusy,
401             modifier = fillMaxWidth.menuAnchor(
402                 type = ExposedDropDownMenuAnchorType.PrimaryNotEditable,
403                 enabled = !isBusy,
404             ),
405             trailingIcon = {
406                 ExposedDropDownMenuDefaults.TrailingIcon(expanded =
algorithmExpanded)
407             },
408         )
409         DropdownMenu(
410             expanded = algorithmExpanded,
411             onDismissRequest = { algorithmExpanded = false },
412         ) {
413             listOf("SHA1", "SHA256", "SHA512").forEach { algo ->
414                 DropdownMenuItem(
415                     text = { Text(algo) },
416                     onClick = {
417                         algorithmValue = algo
418                         algorithmExpanded = false
419                     },
420                 )
421             }
422         }
423     }
424     Text(
425         periodSecondsValue,
426         style = MaterialTheme.typography.labelMedium,
427     )
428     Slider(
429         value = periodSecondsValue.toFloat(),

```



```

430         120) },
431         valueRange = 10f..120f,
432         steps = 109,
433         enabled = !isBusy,
434     )
435     Text(
436         digitsValue),
437         style = MaterialTheme.typography.labelMedium,
438     )
439     Slider(
440         value = digitsValue.toFloat(),
441         onChange = { digitsValue = it.roundToInt().coerceIn(6, 8) },
442         valueRange = 6f..8f,
443         steps = 1,
444         enabled = !isBusy,
445     )
446     TextButton(
447         enabled = !isBusy,
448         onClick = {
449             val granted = ContextCompat.checkSelfPermission(
450                 context,
451                 Manifest.permission.CAMERA,
452             ) == PackageManager.PERMISSION_GRANTED
453             if (granted) {
454                 showScanner = true
455                 permissionDenied = false
456             } else {
457                 cameraPermissionLauncher.launch(Manifest.permission.CAMERA)
458             }
459         },
460     ) {
461         Icon(
462             imageVector = Icons.Default.QrCodeScanner,
463             contentDescription = null,
464             modifier = Modifier.padding(end = 8.dp),
465         )
466         Text(stringResource(R.string.two_fa_scan_qr_action))
467     }
468     if (permissionDenied) {
469         Text(
470             text =
stringResource(R.string.two_fa_camera_permission_required),
471             color = Color.Red,
472             style = MaterialTheme.typography.bodySmall,

```

```

473         )
474     }
475     scanError?.let { error ->
476         Text(
477             text = error,
478             color = Color.Red,
479             style = MaterialTheme.typography.bodySmall,
480         )
481     }
482     OutlinedTextField(
483         value = notes,
484         onValueChange = { notes = it },
485         enabled = !isBusy,
486         label =
487 { Text(stringResource(R.string.two_fa_field_notes_optional)) },
488     )
489     val trackColor = MaterialTheme.colorScheme.outlineVariant
490     val thumbColor = MaterialTheme.colorScheme.primary
491     Box(
492         modifier = Modifier
493             .align(Alignment.CenterEnd)
494             .fillMaxSize()
495             .padding(end = 2.dp, top = 4.dp, bottom = 4.dp),
496         contentAlignment = Alignment.TopEnd,
497     ) {
498         Box(
499             modifier = Modifier
500                 .fillMaxHeight()
501                 .size(width = 3.dp, height = 0.dp)
502                 .background(trackColor),
503         )
504         if (dialogScrollState.maxValue > 0 && scrollContainerHeightPx > 0) {
505             val viewport = scrollContainerHeightPx.toFloat()
506             val content = viewport + dialogScrollState.maxValue.toFloat()
507             val thumbHeightPx = (viewport * (viewport /
508 content)).coerceAtLeast(24f)
509             val maxTravel = (viewport - thumbHeightPx).coerceAtLeast(0f)
510             val offsetY = maxTravel * scrollProgress
511             Box(
512                 modifier = Modifier
513                     .offset { IntOffset(0, offsetY.roundToInt()) }
514                     .size(width = 3.dp, height = with(density)
515 { thumbHeightPx.toDp() })
516                     .background(thumbColor),

```

```

515         )
516     }
517 }
518 }
519 },
520 confirmButton = {
521     TextButton(
522         enabled = !isBusy && issuer.isNotBlank() && account.isNotBlank() &&
secret.isNotBlank(),
523         onClick = {
524             onSave(
525                 issuer,
526                 account,
527                 secret,
528                 notes.ifBlank { null },
529                 digitsValue,
530                 periodSecondsValue,
531                 algorithmValue,
532             )
533         },
534     ) {
535         Text(stringResource(R.string.save))
536     }
537 },
538 dismissButton = {
539     TextButton(onClick = onDismiss, enabled = !isBusy) {
540         Text(stringResource(R.string.cancel))
541     }
542 },
543 )
544
545 if (showScanner) {
546     QrScannerDialog(
547         onDismiss = { showScanner = false },
548         onScanned = { raw ->
549             val parsed = parseOtpAuthTotpUri(raw)
550             if (parsed == null) {
551                 scanError = scanInvalidText
552                 return@QrScannerDialog
553             }
554             issuer = parsed.issuer
555             account = parsed.account
556             secret = parsed.secret
557             digitsValue = parsed.digits.coerceIn(6, 8)
558             periodSecondsValue = parsed.periodSeconds.coerceIn(10, 120)

```

```

559         algorithmValue = parsed.algorithm.uppercase()
560         scanError = null
561         showScanner = false
562     },
563 )
564 }
565 }
566
567 @Composable
568 private fun rememberTwoFaCode(token: TwoFaTokenRecord, nowMillis: Long): TwoFaCodeState? {
569     val period = token.periodSeconds.coerceAtLeast(1)
570     val periodSlot = nowMillis / 1000L / period
571     return remember(token, periodSlot) {
572         buildTwoFaCodeState(token, nowMillis)
573     }
574 }
575

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/twoFa/TwoFaTokensScreenContent.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twoFa
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Row
6 import androidx.compose.foundation.layout.fillMaxSize
7 import androidx.compose.foundation.layout.padding
8 import androidx.compose.foundation.layout.size
9 import androidx.compose.material.icons.Icons
10 import androidx.compose.material.icons.outlined.Lock
11 import androidx.compose.material3.CircularProgressIndicator
12 import androidx.compose.material3.Icon
13 import androidx.compose.material3.MaterialTheme
14 import androidx.compose.material3.Text
15 import androidx.compose.runtime.Composable
16 import androidx.compose.ui.Alignment
17 import androidx.compose.ui.Modifier
18 import androidx.compose.ui.res.stringResource
19 import androidx.compose.ui.unit.dp
20 import com.github.nullptroma.wallenc.ui.R
21 import com.github.nullptroma.wallenc.ui.resources.resolveText
22
23 @Composable
24 fun TwoFaTokensScreenContent(
25     uiState: TwoFaTokensScreenState,
26     modifier: Modifier = Modifier,
27     tokenList: @Composable () -> Unit = {},
28 ) {
29     Column(
30         modifier = modifier.fillMaxSize(),
31         verticalArrangement = Arrangement.spacedBy(12.dp),
32     ) {
33         if (uiState.isLoading) {
34             CircularProgressIndicator()
35             return@Column
36         }
37
38         uiState.errorNotification?.let { notification ->
39             Text(
40                 text = notification.resolveText(),
41                 color = MaterialTheme.colorScheme.error,
42             )
43         }
44     }
45 }
```

```

43         }
44
45         if (uiState.items.isEmpty()) {
46             Text(stringResource(R.string.two_fa_empty_state))
47         } else {
48             tokenList()
49         }
50     }
51 }
52
53 @Composable
54 fun TwoFaTokenListHeader(
55     issuer: String,
56     account: String,
57     modifier: Modifier = Modifier,
58 ) {
59     Row(
60         modifier = modifier,
61         horizontalArrangement = Arrangement.spacedBy(12.dp),
62         verticalAlignment = Alignment.Top,
63     ) {
64         Icon(
65             imageVector = Icons.Outlined.Lock,
66             contentDescription = null,
67             modifier = Modifier
68                 .size(22.dp)
69                 .padding(top = 2.dp),
70             tint = MaterialTheme.colorScheme.primary,
71         )
72         Column {
73             Text(
74                 text = issuer,
75                 style = MaterialTheme.typography.titleSmall,
76             )
77             Text(
78                 text = account,
79                 style = MaterialTheme.typography.bodyMedium,
80                 color = MaterialTheme.colorScheme.onSurfaceVariant,
81             )
82         }
83     }
84 }
85

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/twoFa/TwoFaTokensScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twoFa
2
3 import androidx.compose.runtime.Immutable
4 import com.github.nullptroma.wallenc.domain.datatypes.TwoFaTokenRecord
5 import com.github.nullptroma.wallenc.ui.resources.UserNotification
6
7 @Immutable
8 data class TwoFaTokensScreenState(
9     val isLoading: Boolean = true,
10    val isAvailable: Boolean = false,
11    val isMutating: Boolean = false,
12    val items: List<TwoFaTokenRecord> = emptyList(),
13    val errorNotification: UserNotification.TextRes? = null,
14 )
15
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/twofa/TwoFaTokensViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa
2
3 import androidx.lifecycle.SavedStateHandle
4 import androidx.lifecycle.ViewModelScope
5 import com.github.nullptroma.wallenc.domain.datatypes.TwoFaTokenRecord
6 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
7 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
8 import com.github.nullptroma.wallenc.domain.tasks.VaultTaskStep
9 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
10 import com.github.nullptroma.wallenc.ui.R
11 import com.github.nullptroma.wallenc.domain.errors.WallencException
12 import com.github.nullptroma.wallenc.ui.ViewModelBase
13 import com.github.nullptroma.wallenc.ui.resources.toUserNotification
14 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
15 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.requireStorageUuid
16 import com.github.nullptroma.wallenc.usecases.FindStorageUseCase
17 import com.github.nullptroma.wallenc.usecases.ManageTwoFaTokensUseCase
18 import dagger.hilt.android.lifecycle.HiltViewModel
19 import kotlinx.coroutines.Dispatchers
20 import kotlinx.coroutines.flow.combine
21 import kotlinx.coroutines.flow.map
22 import kotlinx.coroutines.launch
23 import javax.inject.Inject
24
25 @HiltViewModel
26 class TwoFaTokensViewModel @Inject constructor(
27     savedStateHandle: SavedStateHandle,
28     private val findStorageUseCase: FindStorageUseCase,
29     private val manageTwoFaTokensUseCase: ManageTwoFaTokensUseCase,
30     private val taskOrchestrator: ITaskOrchestrator,
31     private val uiStrings: UiStringResolver,
32 ) : ViewModelBase<TwoFaTokensScreenState>(TwoFaTokensScreenState()) {
33
34     private val storageUuid = savedStateHandle.requireStorageUuid()
35
36     init {
37         observeStorageTokens()
38     }
39
40     private fun observeStorageTokens() {
41         viewModelScope.launch {
42             val storage = findStorageUseCase.find(storageUuid)
```



```

43         if (storage == null) {
44             updateState(
45                 state.value.copy(
46                     isLoading = false,
47                     errorNotification =
WallencException.Feature.StorageNotFound().toUserNotification(),
48                 ),
49             )
50             return@launch
51         }
52         if (storage.metaInfo.value.encInfo != null && !storage.isVirtualStorage) {
53             updateState(
54                 state.value.copy(
55                     isLoading = false,
56                     isAvailable = false,
57                     errorNotification =
WallencException.Feature.NeedsDecryptedView().toUserNotification(),
58                 ),
59             )
60             return@launch
61         }
62         combine(
63             storage.isAvailable,
64             manageTwoFaTokensUseCase.observe(storage),
65             taskOrchestrator.pipelineState.map { pipe ->
66                 pipe.tasks.any { t ->
67                     t.busyStorageUuid == storage.uuid && isTaskActive(t.state)
68                 }
69             },
70         ) { available, items, isMutating ->
71             state.value.copy(
72                 isLoading = false,
73                 isAvailable = available,
74                 isMutating = isMutating,
75                 items = items,
76                 errorNotification = null,
77             )
78         }.collect { ui ->
79             updateState(ui)
80         }
81     }
82 }
83
84 fun saveToken(
85     existingId: String?,

```

```

86         issuer: String,
87         account: String,
88         secret: String,
89         notes: String?,
90         digits: Int = 6,
91         periodSeconds: Int = 30,
92         algorithm: String = "SHA1",
93     ) {
94         viewModelScope.launch {
95             val storage = findStorageUseCase.find(storageUuid) ?: return@launch
96             if (storage.metaInfo.value.encInfo != null && !storage.isVirtualStorage) {
97                 updateState(
98                     state.value.copy(
99                         errorNotification =
WallencException.Feature.NeedsDecryptedView().toUserNotification(),
100                     ),
101                 )
102                 return@launch
103             }
104             taskOrchestrator.enqueue(
105                 title = uiStrings(R.string.task_title_save_2fa_token),
106                 dispatcher = Dispatchers.IO,
107                 busyStorageUuid = storage.uuid,
108                 work = { ctx ->
109                     ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.Save2FaToken))
110                     if (existingId == null) {
111                         manageTwoFaTokensUseCase.create(
112                             storageInfo = storage,
113                             issuer = issuer,
114                             account = account,
115                             secret = secret,
116                             notes = notes,
117                             digits = digits,
118                             periodSeconds = periodSeconds,
119                             algorithm = algorithm,
120                         )
121                     } else {
122                         manageTwoFaTokensUseCase.update(
123                             storageInfo = storage,
124                             token = TwoFaTokenRecord(
125                                 id = existingId,
126                                 issuer = issuer,
127                                 account = account,
128                                 secret = secret,
129                                 notes = notes,

```

```

130             digits = digits,
131             periodSeconds = periodSeconds,
132             algorithm = algorithm,
133         ),
134     )
135 }
136 },
137 )
138 }
139 }
140
141 fun deleteToken(id: String) {
142     viewModelScope.launch {
143         val storage = findStorageUseCase.find(storageUuid) ?: return@launch
144         if (storage.metaInfo.value.encInfo != null && !storage.isVirtualStorage) {
145             updateState(
146                 state.value.copy(
147                     errorNotification =
148                     WallencException.Feature.NeedsDecryptedView().toUserNotification(),
149                 ),
150             )
151             return@launch
152         }
153         taskOrchestrator.enqueue(
154             title = uiStrings(R.string.task_title_delete_2fa_token),
155             dispatcher = Dispatchers.IO,
156             busyStorageUuid = storage.uuid,
157             work = { ctx ->
158                 ctx.reportProgress(null,
159                     TaskProgressLabel.VaultTask(VaultTaskStep.Delete2FaToken))
160                 manageTwoFaTokensUseCase.delete(storage, id)
161             },
162         )
163     }
164 }
165
166 private fun isTaskActive(state: TaskRunState): Boolean =
167     state is TaskRunState.Queued || state is TaskRunState.Running
168 }

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/tasks/TaskPipelineRoute.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.main.screens.tasks
2
3  import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4  import kotlinx.parcelize.Parcelize
5  import kotlinx.serialization.Serializable
6
7  @Serializable
8  @Parcelize
9  class TaskPipelineRoute : ScreenRoute()
10
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/tasks/TaskPipelineScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.tasks
2
3 import androidx.compose.foundation.clickable
4 import androidx.compose.foundation.layout.Arrangement
5 import androidx.compose.foundation.layout.Column
6 import androidx.compose.foundation.layout.Row
7 import androidx.compose.foundation.layout.WindowInsets
8 import androidx.compose.foundation.layout.fillMaxSize
9 import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.padding
11 import androidx.compose.foundation.layout.widthIn
12 import androidx.compose.foundation.lazy.LazyColumn
13 import androidx.compose.foundation.lazy.items
14 import androidx.compose.material3.AlertDialog
15 import androidx.compose.material3.Button
16 import androidx.compose.material3.Checkbox
17 import androidx.compose.material3.ExperimentalMaterial3Api
18 import androidx.compose.material3.LinearProgressIndicator
19 import androidx.compose.material3.MaterialTheme
20 import androidx.compose.material3.Scaffold
21 import androidx.compose.material3.Slider
22 import androidx.compose.material3.Text
23 import androidx.compose.material3.TopAppBar
24 import androidx.compose.runtime.Composable
25 import androidx.compose.runtime.getValue
26 import androidx.compose.runtime.mutableFloatStateOf
27 import androidx.compose.runtime.mutableStateOf
28 import androidx.compose.runtime.remember
29 import androidx.compose.runtime.setValue
30 import androidx.compose.ui.Alignment
31 import androidx.compose.ui.Modifier
32 import androidx.compose.ui.res.stringResource
33 import androidx.compose.ui.unit.dp
34 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
35 import androidx.lifecycle.compose.collectAsStateWithLifecycle
36 import com.github.nullptroma.wallenc.domain.tasks.PipelineTask
37 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLevel
38 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
39 import com.github.nullptroma.wallenc.ui.R
40 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLine
41 import com.github.nullptroma.wallenc.ui.resources.displayText
42 import com.github.nullptroma.wallenc.ui.resources.formatTaskPipelineTime
```

```

43 import com.github.nullptroma.wallenc.ui.resources.resolveText
44 import com.github.nullptroma.wallenc.ui.resources.toUserNotification
45
46 @OptIn(ExperimentalMaterial3Api::class)
47 @Composable
48 fun TaskPipelineScreen(
49     modifier: Modifier = Modifier,
50     viewModel: TaskPipelineViewModel = hiltViewModel(),
51 ) {
52     val pipeline by viewModel.orchestrator.pipelineState.collectAsStateWithLifecycle()
53     val logs by viewModel.orchestrator.logLines.collectAsStateWithLifecycle()
54     val hasAnyTask = pipeline.tasks.isNotEmpty()
55     val runningTaskIds = pipeline.runningTaskIds
56     var showTestDialog by remember { mutableStateOf(false) }
57     var testDurationSec by remember { mutableFloatStateOf(10f) }
58     var testInfinity by remember { mutableStateOf(false) }
59
60     Scaffold(
61         modifier = modifier,
62         topBar = {
63             TopAppBar(
64                 windowInsets = WindowInsets(0.dp),
65                 title = { Text(stringResource(R.string.task_pipeline_title)) },
66             )
67         },
68     ) { inner ->
69         Column(
70             Modifier
71                 .padding(inner)
72                 .fillMaxSize()
73                 .padding(16.dp),
74             verticalArrangement = Arrangement.spacedBy(12.dp),
75         ) {
76             Text(
77                 stringResource(R.string.task_pipeline_jobs),
78                 style = MaterialTheme.typography.titleMedium,
79             )
80             LazyColumn(
81                 modifier = Modifier.weight(1f),
82                 verticalArrangement = Arrangement.spacedBy(8.dp),
83             ) {
84                 items(pipeline.tasks, key = { it.id.uuid }) { task ->
85                     TaskRow(task = task, isRunning = task.id in runningTaskIds)
86                 }
87             }

```

```

88         Text(
89             stringResource(R.string.task_pipeline_log),
90             style = MaterialTheme.typography.titleMedium,
91         )
92     LazyColumn(
93         modifier = Modifier.weight(1f),
94         verticalArrangement = Arrangement.spacedBy(4.dp),
95     ) {
96         items(logs.size) { i ->
97             PipelineLogRow(line = logs[i])
98         }
99     }
100     Button(
101         onClick = { showTestDialog = true },
102         modifier = Modifier.fillMaxWidth(),
103     ) {
104         Text(stringResource(R.string.task_pipeline_run_test))
105     }
106
107     Button(
108         onClick = { viewModel.orchestrator.cancelAll() },
109         enabled = hasAnyTask,
110         modifier = Modifier.fillMaxWidth(),
111     ) {
112         Text(stringResource(R.string.task_pipeline_cancel_all))
113     }
114 }
115
116
117 if (showTestDialog) {
118     AlertDialog(
119         onDismissRequest = { showTestDialog = false },
120         title = { Text(stringResource(R.string.task_pipeline_test_dialog_title)) },
121         text = {
122             Column(verticalArrangement = Arrangement.spacedBy(12.dp)) {
123                 Text(
124                     stringResource(
125                         R.string.task_pipeline_test_dialog_duration,
126                         testDurationSec.toInt(),
127                     )
128                 )
129                 Slider(
130                     value = testDurationSec,
131                     onValueChange = { testDurationSec = it },
132                     valueRange = 0f..60f,

```

```

133         )
134         Row(
135             verticalAlignment = Alignment.CenterVertically,
136             modifier = Modifier.fillMaxWidth(),
137         ) {
138             Checkbox(
139                 checked = testInfinity,
140                 onCheckedChange = { testInfinity = it },
141             )
142             Text(
143                 stringResource(R.string.task_pipeline_test_dialog_infinity),
144                 modifier = Modifier
145                     .clickable { testInfinity = !testInfinity }
146                     .weight(1f),
147             )
148         }
149     },
150 },
151 confirmButton = {
152     Button(
153         onClick = {
154             viewModel.startTestTask(
155                 testDurationSec.toInt(),
156                 testInfinity,
157             )
158             showTestDialog = false
159         },
160     ) {
161         Text(stringResource(R.string.task_pipeline_test_dialog_start))
162     }
163 },
164 dismissButton = {
165     Button(onClick = { showTestDialog = false }) {
166         Text(stringResource(R.string.task_pipeline_test_dialog_cancel))
167     }
168 },
169 )
170 }
171 }
172
173 @Composable
174 private fun PipelineLogRow(line: TaskLogLine) {
175     val prefix = when (line.level) {
176         TaskLogLevel.Debug -> "D"

```



```

177         TaskLogLevel.Info -> "I"
178         TaskLogLevel.Warn -> "W"
179         TaskLogLevel.Error -> "E"
180     }
181     Row(
182         modifier = Modifier.fillMaxWidth(),
183         horizontalArrangement = Arrangement.SpaceBetween,
184         verticalAlignment = Alignment.Top,
185     ) {
186         Text(
187             text = "[$prefix] ${line.displayText()}",
188             modifier = Modifier
189                 .weight(1f)
190                 .padding(end = 8.dp),
191             style = MaterialTheme.typography.bodySmall,
192             color = MaterialTheme.colorScheme.onSurfaceVariant,
193         )
194         Text(
195             text = formatTaskPipelineTime(line.timestampMs),
196             modifier = Modifier.widthIn(min = 56.dp),
197             style = MaterialTheme.typography.bodySmall,
198             color = MaterialTheme.colorScheme.onSurfaceVariant,
199         )
200     }
201 }
202
203 @Composable
204 private fun TaskRow(task: PipelineTask, isRunning: Boolean) {
205     Column(Modifier.fillMaxWidth()) {
206         Row(
207             modifier = Modifier.fillMaxWidth(),
208             horizontalArrangement = Arrangement.SpaceBetween,
209             verticalAlignment = Alignment.CenterVertically,
210         ) {
211             Text(
212                 text = task.title,
213                 modifier = Modifier
214                     .weight(1f)
215                     .padding(end = 8.dp),
216                 style = if (isRunning) MaterialTheme.typography.titleSmall
217                     else MaterialTheme.typography.bodyMedium,
218             )
219             Text(
220                 text = formatTaskPipelineTime(task.enqueuedAtMs),
221                 modifier = Modifier.widthIn(min = 56.dp),

```

```

222         style = MaterialTheme.typography.bodySmall,
223         color = MaterialTheme.colorScheme.onSurfaceVariant,
224     )
225 }
226 val runningProgress = (task.state as? TaskRunState.Running)?.progress
227 val progressLabel = runningProgress?.label?.resolveText()
228 val stateLabel = when (val s = task.state) {
229     TaskRunState.Queued -> stringResource(R.string.task_state_queued)
230     is TaskRunState.Running ->
231         progressLabel ?: stringResource(R.string.task_state_running)
232     TaskRunState.Completed -> stringResource(R.string.task_state_completed)
233     TaskRunState.Cancelled -> stringResource(R.string.task_state_cancelled)
234     is TaskRunState.Failed -> {
235         val notification = s.error.toUserNotification()
236         stringResource(
237             R.string.task_state_failed,
238             if (notification.formatArgs.isEmpty()) {
239                 stringResource(notification.id)
240             } else {
241                 stringResource(notification.id,
242 *notification.formatArgs.toTypedArray())
243             },
244         )
245     }
246 Text(stateLabel, style = MaterialTheme.typography.bodySmall)
247 if (task.state is TaskRunState.Running) {
248     val frac = runningProgress?.fraction
249     if (frac != null) {
250         LinearProgressIndicator(
251             progress = { frac },
252             modifier = Modifier
253                 .fillMaxWidth()
254                 .padding(top = 4.dp),
255         )
256     } else {
257         LinearProgressIndicator(
258             modifier = Modifier
259                 .fillMaxWidth()
260                 .padding(top = 4.dp),
261         )
262     }
263 }
264 }
265 }

```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/tasks/TaskPipelineViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.tasks
2
3 import androidx.lifecycle.ViewModel
4 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
5 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLevel
6 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
7 import com.github.nullptroma.wallenc.ui.R
8 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
9 import dagger.hilt.android.lifecycle.HiltViewModel
10 import kotlinx.coroutines.Dispatchers
11 import kotlinx.coroutines.delay
12 import javax.inject.Inject
13
14 @HiltViewModel
15 class TaskPipelineViewModel @Inject constructor(
16     val orchestrator: ITaskOrchestrator,
17     private val uiStrings: UiStringResolver,
18 ) : ViewModel() {
19
20     fun startTestTask(durationSec: Int, infinityIndeterminateProgress: Boolean) {
21         val safeDurationSec = durationSec.coerceIn(0, 60)
22         val title =
23             if (infinityIndeterminateProgress) {
24                 uiStrings(R.string.task_pipeline_test_running_infinity, safeDurationSec)
25             } else {
26                 uiStrings(R.string.task_pipeline_test_running, safeDurationSec)
27             }
28         orchestrator.enqueue(
29             title = title,
30             dispatcher = Dispatchers.Default,
31             work = { ctx ->
32                 val steps = if (safeDurationSec == 0) 1 else safeDurationSec * 10
33                 ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_test_started,
34                     safeDurationSec))
35                 for (step in 0..steps) {
36                     val fraction = step.toFloat() / steps.toFloat()
37                     val elapsedSec = ((fraction * safeDurationSec * 1000).toInt()) / 1000
38                     ctx.reportProgress(
39                         fraction = if (infinityIndeterminateProgress) null else fraction,
40                         label = TaskProgressLabel.TestElapsed(elapsedSec, safeDurationSec),
41                     )
42                     if (step < steps) delay(100)
43                 }
44             }
45         )
46     }
47 }
```

```
42         }
43         ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_test_finished))
44     },
45 )
46 }
47 }
48
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/AbstractVaultBrowserViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3 import androidx.annotation.StringRes
4 import androidx.lifecycle.viewModelScope
5 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
6 import com.github.nullptroma.wallenc.domain.datatypes.StorageMetaLoadState
7 import com.github.nullptroma.wallenc.domain.datatypes.Tree
8 import com.github.nullptroma.wallenc.domain.errors.WallencException
9 import com.github.nullptroma.wallenc.domain.errors.toWallencException
10 import java.io.IOException
11 import com.github.nullptroma.wallenc.domain.interfaces.ILogger
12 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
13 import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
14 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
15 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLevel
16 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
17 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
18 import com.github.nullptroma.wallenc.domain.tasks.VaultTaskStep
19 import com.github.nullptroma.wallenc.ui.R
20 import com.github.nullptroma.wallenc.ui.ViewModelBase
21 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
22 import com.github.nullptroma.wallenc.ui.resources.UserNotification
23 import com.github.nullptroma.wallenc.ui.resources.toUserNotification
24 import com.github.nullptroma.wallenc.usecases.GetOpenedStoragesUseCase
25 import com.github.nullptroma.wallenc.usecases.ManageStoragesEncryptionUseCase
26 import com.github.nullptroma.wallenc.usecases.ManageVaultUseCase
27 import com.github.nullptroma.wallenc.usecases.RemoveStorageUseCase
28 import com.github.nullptroma.wallenc.usecases.RenameStorageUseCase
29 import com.github.nullptroma.wallenc.usecases.StorageFileManagementUseCase
30 import kotlinx.coroutines.Dispatchers
31 import kotlinx.coroutines.flow.Flow
32 import kotlinx.coroutines.flow.MutableSharedFlow
33 import kotlinx.coroutines.flow.SharedFlow
34 import kotlinx.coroutines.flow.combine
35 import kotlinx.coroutines.flow.distinctUntilChanged
36 import kotlinx.coroutines.flow.flowOf
37 import kotlinx.coroutines.flow.map
38 import kotlinx.coroutines.launch
39 import java.util.UUID
40
41 /**
42  * Общая логика дерева storages для локального и удалённого vault (presentation).
```

```

43     */
44     abstract class AbstractVaultBrowserViewModel(
45         storagesFlow: Flow<List<IStorage>>,
46         private val storagesScanInProgressFlow: Flow<Boolean> = flowOf(false),
47         private val vaultHeaderFlow: Flow<VaultBrowserHeader?> = flowOf(null),
48         private val vaultAvailabilityFlow: Flow<Boolean>,
49         private val resolveCreateVaultUuid: () -> UUID?,
50         private val removeStorageUseCase: RemoveStorageUseCase,
51         private val getOpenedStoragesUseCase: GetOpenedStoragesUseCase,
52         private val storageFileManagementUseCase: StorageFileManagementUseCase,
53         private val manageStoragesEncryptionUseCase: ManageStoragesEncryptionUseCase,
54         private val renameStorageUseCase: RenameStorageUseCase,
55         private val manageVaultUseCase: ManageVaultUseCase,
56         private val taskOrchestrator: ITaskOrchestrator,
57         private val uiStrings: UiStringResolver,
58         private val logger: ILogger,
59     ) : ViewModelBase<VaultBrowserScreenState>(
60         VaultBrowserScreenState(
61             storagesList = emptyList(),
62             storagesRefreshing = true,
63             busyStorageUuids = emptySet(),
64             vaultListMutationActive = false,
65             addStorageFabEnabled = false,
66         ),
67     ) {
68
69         private val _userNotifications = MutableSharedFlow<UserNotification>(extraBufferCapacity
70             = 8)
71         val userNotifications: SharedFlow<UserNotification> = _userNotifications
72
73         /** Удалённый vault: показать кнопку повторного сканирования storages на Диске. */
74         open val supportsStorageRescan: Boolean = false
75
76         init {
77             collectStoragesFlow(storagesFlow)
78             collectVaultHeaderFlow()
79             collectPipelineBusyFlags()
80             viewModelScope.launch {
81                 vaultAvailabilityFlow
82                     .distinctUntilChanged()
83                     .collect { available ->
84                         updateState(state.value.copy(addStorageFabEnabled = available))
85                         logger.debug(TAG, "vault availability → add FAB enabled=$available")
86                     }
87             }
88         }
89     }

```

```

87     }
88
89     private fun isPipelineTaskActive(state: TaskRunState): Boolean =
90         when (state) {
91             is TaskRunState.Queued,
92             is TaskRunState.Running,
93             -> true
94             else -> false
95         }
96
97     private fun isStorageTaskActive(storageUuid: UUID): Boolean =
98         taskOrchestrator.pipelineState.value.tasks.any { t ->
99             t.busyStorageUuid == storageUuid && isPipelineTaskActive(t.state)
100         }
101
102     private fun isVaultListMutationActive(): Boolean =
103         taskOrchestrator.pipelineState.value.tasks.any { t ->
104             t.locksVaultStorageList && isPipelineTaskActive(t.state)
105         }
106
107     private fun collectVaultHeaderFlow() {
108         viewModelScope.launch {
109             vaultHeaderFlow.collect { header ->
110                 updateState(state.value.copy(header = header))
111             }
112         }
113     }
114
115     private fun collectStoragesFlow(storagesFlow: Flow<List<IStorage>>) {
116         viewModelScope.launch {
117             combine(
118                 storagesFlow,
119                 storagesScanInProgressFlow,
120                 getOpenedStoragesUseCase.openedStorages,
121             ) { storages, scanInProgress, opened ->
122                 Triple(storages, scanInProgress, opened)
123             }.collect { (storages, scanInProgress, opened) ->
124                 val list = mutableListOf<Tree<IStorageInfo>>()
125                 for (storage in storages) {
126                     val tree = Tree<IStorageInfo>(storage)
127                     list.add(tree)
128                     while (opened.containsKey(tree.value.uuid)) {
129                         val child = opened.getValue(tree.value.uuid)
130                         val nextTree = Tree(child)
131                         tree.children = listOf(nextTree)

```



```

132         tree = nextTree
133     }
134 }
135     updateState(
136         state.value.copy(
137             storagesList = list,
138             storagesRefreshing = scanInProgress,
139         ),
140     )
141 }
142 }
143 }
144
145 private fun collectPipelineBusyFlags() {
146     viewModelScope.launch {
147         taskOrchestrator.pipelineState
148             .map { pipe ->
149                 val activeTasks = pipe.tasks.filter { isPipelineTaskActive(it.state) }
150                 val busyUuids = activeTasks.mapNotNull { it.busyStorageUuid }.toSet()
151                 val listMut = activeTasks.any { it.locksVaultStorageList }
152                 busyUuids to listMut
153             }
154             .distinctUntilChanged()
155             .collect { (busyUuids, listMut) ->
156                 updateState(
157                     state.value.copy(
158                         busyStorageUuids = busyUuids,
159                         vaultListMutationActive = listMut,
160                     ),
161                 )
162             }
163     }
164 }
165
166 private fun notifyUser(@StringRes messageRes: Int) {
167     viewModelScope.launch {
168         _userNotifications.emit(UserNotification.TextRes(messageRes))
169     }
170 }
171
172 fun rescanStorages() {
173     if (!supportsStorageRescan) return
174     if (state.value.storagesRefreshing) {
175         notifyUser(R.string.vault_msg_rescan_already_in_progress)
176         return
177     }

```

```

177         }
178         if (isVaultListMutationActive()) {
179             notifyUser(R.string.vault_msg_vault_list_mutation_busy)
180             return
181         }
182         val vaultUuid = resolveCreateVaultUuid() ?: return
183         taskOrchestrator.enqueue(
184             title = uiStrings(R.string.task_title_rescan_vault_storages),
185             dispatcher = Dispatchers.IO,
186             locksVaultStorageList = true,
187             work = { ctx ->
188                 try {
189                     ctx.reportProgress(null,
190 TaskProgressLabel.VaultTask(VaultTaskStep.RescanVaultStorages))
191                     ctx.log(TaskLogLevel.Info,
192 uiStrings(R.string.task_log_rescanning_vault_storages))
193                     manageVaultUseCase.rescanStorages(vaultUuid)
194                     ctx.log(TaskLogLevel.Info,
195 uiStrings(R.string.task_log_rescan_vault_storages_done))
196                     val vault = manageVaultUseCase.find(vaultUuid)
197                     if (vault != null && !vault.isAvailable.value) {
198                         emitTaskError(WallencException.Network.IoFailed(IOException("Vault
199 unreachable")))
200                     }
201                 } catch (e: Exception) {
202                     logger.debug(TAG, "rescanStorages failed: ${e.stackTraceToString()}")
203                     ctx.log(TaskLogLevel.Error,
204 uiStrings(R.string.task_log_rescan_vault_storages_failed))
205                     emitTaskError(e)
206                 }
207             },
208         )
209     }
210
211     fun createStorage() {
212         if (!state.value.addStorageFabEnabled) {
213             logger.debug(TAG, "createStorage ignored (vault unavailable or FAB disabled)")
214             return
215         }
216         if (isVaultListMutationActive()) {
217             logger.debug(TAG, "createStorage ignored (vault list mutation already running)")
218             notifyUser(R.string.vault_msg_vault_list_mutation_busy)
219             return
220         }
221         logger.debug(TAG, "createStorage: enqueue task")
222         taskOrchestrator.enqueue(

```

```

218         title = uiStrings(R.string.task_title_create_storage),
219         dispatcher = Dispatchers.IO,
220         locksVaultStorageList = true,
221         work = { ctx ->
222             try {
223                 ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.CreateStorage))
224                 ctx.log(TaskLogLevel.Info,
uiStrings(R.string.task_log_creating_storage))
225                 val uuid = resolveCreateVaultUuid()
226                 ?: throw IllegalStateException("Vault is not available")
227                 logger.debug(TAG, "createStorage: vaultUuid=$uuid")
228                 val storage = manageVaultUseCase.createStorage(uuid)
229                 ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_storage_created))
230                 logger.debug(TAG, "createStorage: done storageUuid=${storage.uuid}")
231             } catch (e: Exception) {
232                 logger.debug(TAG, "createStorage failed: ${e.stackTraceToString()}")
233                 ctx.log(TaskLogLevel.Error,
uiStrings(R.string.task_log_add_vault_failed))
234                 emitTaskError(e)
235             }
236         },
237     )
238 }
239
240 fun enableEncryption(
241     storage: IStorageInfo,
242     password: String,
243     encryptPath: Boolean,
244     rememberPassword: Boolean,
245 ) {
246     val id = storage.uuid
247     if (isStorageTaskActive(id)) {
248         notifyUser(R.string.vault_msg_storage_pipeline_busy)
249         return
250     }
251     val key = EncryptKey(password)
252     taskOrchestrator.enqueue(
253         title = uiStrings(R.string.task_title_enable_encryption),
254         dispatcher = Dispatchers.IO,
255         busyStorageUuid = id,
256         work = { ctx ->
257             try {
258                 ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.EnableEncryption))
259                 ctx.log(TaskLogLevel.Info,
uiStrings(R.string.task_log_checking_storage))

```

```

260         when (manageStoragesEncryptionUseCase.canEncrypt(storage)) {
261             ManageStoragesEncryptionUseCase.CanEncryptResult.Allowed -> {
262                 ctx.log(TaskLogLevel.Info,
263                 uiStrings(R.string.task_log_encrypting))
264                 manageStoragesEncryptionUseCase.enableEncryption(storage, key,
265                 encryptPath)
266                 if (rememberPassword) {
267                     manageStoragesEncryptionUseCase.rememberStorageKey(storage,
268                     key)
269                 }
270                 try {
271                     manageStoragesEncryptionUseCase.openStorage(
272                     storage,
273                     key,
274                     rememberPassword = false,
275                     )
276                     ctx.log(TaskLogLevel.Info,
277                     uiStrings(R.string.task_log_encryption_enabled))
278                     _userNotifications.emit(UserNotification.TextRes(R.string.msg_encryption_enabled))
279                     } catch (openError: Exception) {
280                         logger.debug(TAG, "open after encrypt failed:
281                         ${openError.stackTraceToString()}")
282                         ctx.log(TaskLogLevel.Info,
283                         uiStrings(R.string.task_log_encryption_enabled))
284                         _userNotifications.emit(
285                         UserNotification.TextRes(R.string.msg_encryption_enabled_open_failed),
286                         )
287                     }
288                 }
289             }
290             ManageStoragesEncryptionUseCase.CanEncryptResult.AlreadyEncrypted ->
291             {
292                 ctx.log(TaskLogLevel.Info,
293                 uiStrings(R.string.task_log_already_encrypted))
294                 _userNotifications.emit(UserNotification.TextRes(R.string.msg_storage_already_encrypted))
295             }
296             ManageStoragesEncryptionUseCase.CanEncryptResult.StorageIsNotEmpty -
297             > {
298                 ctx.log(TaskLogLevel.Info,
299                 uiStrings(R.string.task_log_not_empty))
300                 _userNotifications.emit(UserNotification.TextRes(R.string.msg_storage_not_empty))
301             }
302             ManageStoragesEncryptionUseCase.CanEncryptResult.StorageStateUnknown
303             -> {
304                 ctx.log(TaskLogLevel.Info,
305                 uiStrings(R.string.task_log_empty_unknown))
306                 _userNotifications.emit(UserNotification.TextRes(R.string.msg_storage_empty_state_unknown))
307             }

```

```

3295 ManageStoragesEncryptionUseCase.CanEncryptResult.UnsupportedStorageType -> {
3296     ctx.log(TaskLogLevel.Info,
3297     uiStrings(R.string.task_log_unsupported_type))
3298     _userNotifications.emit(UserNotification.TextRes(R.string.msg_unsupported_storage_type))
3299     }
3300     }
3301     } catch (e: Exception) {
3302         ctx.log(TaskLogLevel.Error,
3303         uiStrings(R.string.task_log_enable_encryption_failed))
3304         emitTaskError(e)
3305     }
3306     },
3307 )
3308 }
3309
3310 fun openEncryptedStorage(storage: IStorageInfo, password: String, rememberPassword:
3311 Boolean) {
3312     val id = storage.uuid
3313     if (isStorageTaskActive(id)) {
3314         notifyUser(R.string.vault_msg_storage_pipeline_busy)
3315         return
3316     }
3317     val key = EncryptKey(password)
3318     taskOrchestrator.enqueue(
3319         title = uiStrings(R.string.task_title_open_encrypted_storage),
3320         dispatcher = Dispatchers.IO,
3321         busyStorageUuid = id,
3322         work = { ctx ->
3323             try {
3324                 ctx.reportProgress(null,
3325                 TaskProgressLabel.VaultTask(VaultTaskStep.DecryptRunning))
3326                 ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_opening_storage))
3327                 manageStoragesEncryptionUseCase.openStorage(storage, key,
3328                 rememberPassword)
3329                 ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_storage_opened))
3330             } catch (e: Exception) {
3331                 ctx.log(TaskLogLevel.Error,
3332                 uiStrings(R.string.task_log_open_storage_failed))
3333                 emitTaskError(e)
3334             }
3335         },
3336     )
3337 }
3338
3339 fun closeEncryptedStorage(storage: IStorageInfo) {
3340     val id = storage.uuid

```

```

335         if (isStorageTaskActive(id)) {
336             notifyUser(R.string.vault_msg_storage_pipeline_busy)
337             return
338         }
339         taskOrchestrator.enqueue(
340             title = uiStrings(R.string.task_title_close_encrypted_storage),
341             dispatcher = Dispatchers.IO,
342             busyStorageUuid = id,
343             work = { ctx ->
344                 try {
345                     ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.CloseStorage))
346                     ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_closing_storage))
347                     manageStoragesEncryptionUseCase.closeStorage(storage)
348                     ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_storage_closed))
349                 } catch (e: Exception) {
350                     ctx.log(TaskLogLevel.Error,
uiStrings(R.string.task_log_close_storage_failed))
351                     emitTaskError(e)
352                 }
353             },
354         )
355     }
356
357     fun disableEncryption(storage: IStorageInfo) {
358         val id = storage.uuid
359         if (isStorageTaskActive(id)) {
360             notifyUser(R.string.vault_msg_storage_pipeline_busy)
361             return
362         }
363         taskOrchestrator.enqueue(
364             title = uiStrings(R.string.task_title_disable_encryption),
365             dispatcher = Dispatchers.IO,
366             busyStorageUuid = id,
367             work = { ctx ->
368                 try {
369                     ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.DisableEncryption))
370                     ctx.log(TaskLogLevel.Info,
uiStrings(R.string.task_log_disabling_encryption))
371                     manageStoragesEncryptionUseCase.clearAndDisableEncryption(storage) { p -
>
372                         ctx.reportProgress(
373                             p.fraction,
374                             p.label ?:
TaskProgressLabel.VaultTask(VaultTaskStep.DisableEncryption),
375                         )

```

```

376         }
377         ctx.log(TaskLogLevel.Info,
uiStrings(R.string.task_log_encryption_disabled))
378     _userNotifications.emit(UserNotification.TextRes(R.string.msg_encryption_disabled))
379     } catch (e: Exception) {
380         ctx.log(TaskLogLevel.Error,
uiStrings(R.string.task_log_disable_encryption_failed))
381         emitTaskError(e)
382     }
383     },
384 )
385 }
386
387 fun rename(storage: IStorageInfo, newName: String) {
388     val id = storage.uuid
389     if (isStorageTaskActive(id)) {
390         notifyUser(R.string.vault_msg_storage_pipeline_busy)
391         return
392     }
393     taskOrchestrator.enqueue(
394         title = uiStrings(R.string.task_title_rename_storage),
395         dispatcher = Dispatchers.IO,
396         busyStorageUuid = id,
397         work = { ctx ->
398             try {
399                 ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.RenameStorage))
400                 ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_renaming))
401                 renameStorageUseCase.rename(storage, newName)
402                 ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_renamed))
403             } catch (e: Exception) {
404                 ctx.log(TaskLogLevel.Error, uiStrings(R.string.task_log_rename_failed))
405             }
406         },
407     )
408 }
409
410 fun remove(storage: IStorageInfo) {
411     val id = storage.uuid
412     if (isStorageTaskActive(id)) {
413         notifyUser(R.string.vault_msg_storage_pipeline_busy)
414         return
415     }
416     taskOrchestrator.enqueue(
417         title = uiStrings(R.string.task_title_remove_storage),

```

```

418         dispatcher = Dispatchers.IO,
419         busyStorageUuid = id,
420         locksVaultStorageList = true,
421         work = { ctx ->
422             try {
423                 ctx.reportProgress(null,
TaskProgressLabel.VaultTask(VaultTaskStep.RemoveStorage))
424                 ctx.log(TaskLogLevel.Info,
uiStrings(R.string.task_log_removing_storage))
425                 removeStorageUseCase.remove(storage)
426                 ctx.log(TaskLogLevel.Info, uiStrings(R.string.task_log_removed))
427             } catch (e: Exception) {
428                 ctx.log(TaskLogLevel.Error, uiStrings(R.string.task_log_remove_failed))
429             }
430         },
431     )
432 }
433
434 @StringRes
435 fun getStorageStatusRes(storage: IStorageInfo): Int {
436     if (storage.metaLoadState.value == StorageMetaLoadState.Unavailable) {
437         return R.string.storage_status_meta_unavailable
438     }
439     val encrypted = storage.metaInfo.value.encInfo != null
440     if (!encrypted) return R.string.storage_status_not_encrypted
441     val opened = isEncryptionSessionOpen(storage)
442     return if (opened) R.string.storage_status_encrypted_open else
R.string.storage_status_encrypted_closed
443 }
444
445 fun isEncryptionSessionOpen(storage: IStorageInfo): Boolean {
446     val openedMap = getOpenedStoragesUseCase.openedStorages.value
447     return openedMap.containsKey(storage.uuid)
448 }
449
450 suspend fun isStorageSyncLockHeld(storage: IStorageInfo): Boolean {
451     val s = storage as? IStorage ?: return false
452     return try {
453         s.accessor.readSyncLock() != null
454     } catch (_: Exception) {
455         false
456     }
457 }
458
459 fun clearStorageSyncLock(storage: IStorageInfo) {
460     val id = storage.uuid

```



```

461         if (isStorageTaskActive(id)) {
462             notifyUser(R.string.vault_msg_storage_pipeline_busy)
463             return
464         }
465         taskOrchestrator.enqueue(
466             title = uiStrings(R.string.task_title_clear_sync_lock),
467             dispatcher = Dispatchers.IO,
468             busyStorageUuid = id,
469             work = { ctx ->
470                 try {
471                     val s = storage as? IStorage
472                     if (s == null) {
473                         ctx.log(TaskLogLevel.Error,
474                             uiStrings(R.string.task_log_invalid_storage))
475                         _userNotifications.emit(UserNotification.TextRes(R.string.msg_invalid_storage_for_sync_lock))
476                         return@enqueue
477                     }
478                     ctx.reportProgress(null,
479                         TaskProgressLabel.VaultTask(VaultTaskStep.ClearSyncLock))
480                     ctx.log(TaskLogLevel.Info,
481                         uiStrings(R.string.task_log_clearing_sync_lock))
482                     s.accessor.forceClearSyncLock()
483                     ctx.log(TaskLogLevel.Info,
484                         uiStrings(R.string.task_log_sync_lock_cleared))
485                     _userNotifications.emit(UserNotification.TextRes(R.string.msg_sync_lock_cleared))
486                     } catch (e: Exception) {
487                         ctx.log(TaskLogLevel.Error,
488                             uiStrings(R.string.task_log_clear_sync_lock_failed))
489                         emitTaskError(e)
490                     }
491                 },
492             )
493         }
494
495         private suspend fun emitTaskError(e: Exception) {
496             _userNotifications.emit(e.toWallencException().toUserNotification())
497         }
498
499         private companion object {
500             private const val TAG = "VaultBrowser"
501         }
502     }

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/LocalVaultRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3 import com.github.nullptroma.wallenc.ui.screens.main.MainRoute
4 import kotlinx.parcelize.Parcelize
5 import kotlinx.serialization.Serializable
6
7 @Serializable
8 @Parcelize
9 class LocalVaultRoute : MainRoute()
10
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/LocalVaultScreen.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3  import androidx.compose.runtime.Composable
4  import androidx.compose.ui.Modifier
5  import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
6
7  @Composable
8  fun LocalVaultScreen(
9      modifier: Modifier = Modifier,
10     viewModel: LocalVaultViewModel = hiltViewModel(),
11     onOpenStorageHome: (String) -> Unit,
12 ) {
13     VaultBrowserScreen(
14         modifier = modifier,
15         viewModel = viewModel,
16         onOpenStorageHome = onOpenStorageHome,
17     )
18 }
19
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/LocalVaultViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3 import com.github.nullptroma.wallenc.domain.interfaces.ILogger
4 import com.github.nullptroma.wallenc.domain.interfaces.IVault
5 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
6 import com.github.nullptroma.wallenc.ui.R
7 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
8 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
9 import com.github.nullptroma.wallenc.usecases.GetOpenedStoragesUseCase
10 import com.github.nullptroma.wallenc.usecases.ManageStoragesEncryptionUseCase
11 import com.github.nullptroma.wallenc.usecases.ManageVaultUseCase
12 import com.github.nullptroma.wallenc.usecases.RemoveStorageUseCase
13 import com.github.nullptroma.wallenc.usecases.RenameStorageUseCase
14 import com.github.nullptroma.wallenc.usecases.StorageFileManagementUseCase
15 import com.github.nullptroma.wallenc.vault.contract.DescribedVault
16 import com.github.nullptroma.wallenc.vault.contract.VaultDescriptor
17 import com.github.nullptroma.wallenc.vault.contract.described
18 import com.github.nullptroma.wallenc.vault.contract.locals
19 import dagger.hilt.android.lifecycle.HiltViewModel
20 import kotlinx.coroutines.ExperimentalCoroutinesApi
21 import kotlinx.coroutines.flow.flatMapLatest
22 import kotlinx.coroutines.flow.flowOf
23 import kotlinx.coroutines.flow.map
24 import javax.inject.Inject
25
26 @OptIn(ExperimentalCoroutinesApi::class)
27 @HiltViewModel
28 class LocalVaultViewModel @Inject constructor(
29     vaultsManager: IVaultsManager,
30     manageVaultUseCase: ManageVaultUseCase,
31     removeStorageUseCase: RemoveStorageUseCase,
32     getOpenedStoragesUseCase: GetOpenedStoragesUseCase,
33     storageFileManagementUseCase: StorageFileManagementUseCase,
34     manageStoragesEncryptionUseCase: ManageStoragesEncryptionUseCase,
35     renameStorageUseCase: RenameStorageUseCase,
36     taskOrchestrator: ITaskOrchestrator,
37     uiStrings: UiStringResolver,
38     logger: ILogger,
39 ) : AbstractVaultBrowserViewModel(
40     storagesFlow = vaultsManager.vaults
41         .map { vaults -> vaults.described().locals.firstOrNull() }
42         .flatMapLatest { v -> v?.storages ?: flowOf(emptyList()) },
```

```

43     vaultHeaderFlow = vaultsManager.vaults
44         .map { vaults ->
45 vaults.described().locals.firstOrNull().toLocalVaultBrowserHeader() },
46     vaultAvailabilityFlow = vaultsManager.vaults
47         .map { vaults -> vaults.described().locals.firstOrNull() }
48         .flatMapLatest { v -> v?.isAvailable ? flowOf(false) },
49     resolveCreateVaultUuid =
50 { vaultsManager.vaults.value.described().locals.firstOrNull()?.uuid },
51     removeStorageUseCase = removeStorageUseCase,
52     getOpenedStoragesUseCase = getOpenedStoragesUseCase,
53     storageFileManagementUseCase = storageFileManagementUseCase,
54     manageStoragesEncryptionUseCase = manageStoragesEncryptionUseCase,
55     renameStorageUseCase = renameStorageUseCase,
56     manageVaultUseCase = manageVaultUseCase,
57     taskOrchestrator = taskOrchestrator,
58     uiStrings = uiStrings,
59     logger = logger,
60 )
61
62 private fun IVault?.toLocalVaultBrowserHeader(): VaultBrowserHeader? {
63     if ((this as? DescribedVault)?.descriptor !is VaultDescriptor.LocalDevice) return null
64     return VaultBrowserHeader(titleResId = R.string.screen_title_local_vault)
65 }

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/RemoteVaultViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3 import androidx.lifecycle.SavedStateHandle
4 import com.github.nullptroma.wallenc.domain.interfaces.ILogger
5 import com.github.nullptroma.wallenc.domain.interfaces.IVault
6 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
7 import com.github.nullptroma.wallenc.ui.R
8 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
9 import com.github.nullptroma.wallenc.vault.contract.CloudBrand
10 import com.github.nullptroma.wallenc.vault.contract.DescribedVault
11 import com.github.nullptroma.wallenc.vault.contract.VaultDescriptor
12 import com.github.nullptroma.wallenc.usecases.GetOpenedStoragesUseCase
13 import com.github.nullptroma.wallenc.usecases.ManageStoragesEncryptionUseCase
14 import com.github.nullptroma.wallenc.usecases.ManageVaultUseCase
15 import com.github.nullptroma.wallenc.usecases.RemoveStorageUseCase
16 import com.github.nullptroma.wallenc.usecases.RenameStorageUseCase
17 import com.github.nullptroma.wallenc.usecases.StorageFileManagementUseCase
18 import dagger.hilt.android.lifecycle.HiltViewModel
19 import kotlinx.coroutines.ExperimentalCoroutinesApi
20 import kotlinx.coroutines.flow.flatMapLatest
21 import kotlinx.coroutines.flow.flowOf
22 import kotlinx.coroutines.flow.map
23 import java.util.UUID
24 import javax.inject.Inject
25
26 @OptIn(ExperimentalCoroutinesApi::class)
27 @HiltViewModel
28 class RemoteVaultViewModel @Inject constructor(
29     savedStateHandle: SavedStateHandle,
30     manageVaultUseCase: ManageVaultUseCase,
31     removeStorageUseCase: RemoveStorageUseCase,
32     getOpenedStoragesUseCase: GetOpenedStoragesUseCase,
33     storageFileManagementUseCase: StorageFileManagementUseCase,
34     manageStoragesEncryptionUseCase: ManageStoragesEncryptionUseCase,
35     renameStorageUseCase: RenameStorageUseCase,
36     taskOrchestrator: ITaskOrchestrator,
37     uiStrings: UiStringResolver,
38     logger: ILogger,
39 ) : AbstractVaultBrowserViewModel(
40     storagesFlow = manageVaultUseCase.storagesOf(savedStateHandle.requireVaultUuid()),
41     storagesScanInProgressFlow = manageVaultUseCase.storagesScanInProgressOf(
42         savedStateHandle.requireVaultUuid(),
```

```

43     ),
44     vaultHeaderFlow = manageVaultUseCase.observe(savedStateHandle.requireVaultUuid())
45         .map { vault -> vault.toRemoteVaultBrowserHeader() },
46     vaultAvailabilityFlow = manageVaultUseCase.observe(savedStateHandle.requireVaultUuid())
47         .flatMapLatest { v -> v?.isAvailable ?: flowOf(false) },
48     resolveCreateVaultUuid = { savedStateHandle.requireVaultUuid() },
49     removeStorageUseCase = removeStorageUseCase,
50     getOpenedStoragesUseCase = getOpenedStoragesUseCase,
51     storageFileManagementUseCase = storageFileManagementUseCase,
52     manageStoragesEncryptionUseCase = manageStoragesEncryptionUseCase,
53     renameStorageUseCase = renameStorageUseCase,
54     manageVaultUseCase = manageVaultUseCase,
55     taskOrchestrator = taskOrchestrator,
56     uiStrings = uiStrings,
57     logger = logger,
58 ) {
59     override val supportsStorageRescan: Boolean = true
60 }
61
62 private fun SavedStateHandle.requireVaultUuid(): UUID {
63     val raw = get<String>("vaultUuid") ?: error("Missing vault UUID in navigation
arguments")
64     return UUID.fromString(raw)
65 }
66
67 private fun IVault?.toRemoteVaultBrowserHeader(): VaultBrowserHeader? {
68     val remote = (this as? DescribedVault)?.descriptor as? VaultDescriptor.LinkedRemote ?:
return null
69     val subtitle = when (remote.brand) {
70         CloudBrand.YANDEX -> remote.accountDisplayName
71     }
72     val titleResId = when (remote.brand) {
73         CloudBrand.YANDEX -> R.string.screen_title_yandex_vault
74     }
75     return VaultBrowserHeader(
76         titleResId = titleResId,
77         subtitle = subtitle,
78     )
79 }
80

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/VaultBrowserHeader.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3 import androidx.annotation.StringRes
4
5 data class VaultBrowserHeader(
6     @param:StringRes val titleResId: Int,
7     val subtitle: String? = null,
8 )
9
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/VaultBrowserRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4 import kotlinx.parcelize.Parcelize
5 import kotlinx.serialization.Serializable
6
7 @Serializable
8 @Parcelize
9 data class VaultBrowserRoute(val vaultUuid: String) : ScreenRoute()
10
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/VaultBrowserScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3 import android.widget.Toast
4 import androidx.compose.foundation.background
5 import androidx.compose.foundation.layout.Arrangement
6 import androidx.compose.foundation.layout.Box
7 import androidx.compose.foundation.layout.Column
8 import androidx.compose.foundation.layout.Row
9 import androidx.compose.foundation.layout.Spacer
10 import androidx.compose.foundation.layout.WindowInsets
11 import androidx.compose.foundation.layout.fillMaxSize
12 import androidx.compose.foundation.layout.fillMaxWidth
13 import androidx.compose.foundation.layout.height
14 import androidx.compose.foundation.layout.padding
15 import androidx.compose.foundation.layout.size
16 import androidx.compose.foundation.lazy.LazyColumn
17 import androidx.compose.foundation.lazy.items
18 import androidx.compose.material.icons.Icons
19 import androidx.compose.material.icons.filled.Add
20 import androidx.compose.material3.CircularProgressIndicator
21 import androidx.compose.material3.FilledTonalButton
22 import androidx.compose.material3.FloatingActionButton
23 import androidx.compose.material3.Icon
24 import androidx.compose.material3.MaterialTheme
25 import androidx.compose.material3.Scaffold
26 import androidx.compose.material3.Text
27 import androidx.compose.runtime.Composable
28 import androidx.compose.runtime.LaunchedEffect
29 import androidx.compose.runtime.getValue
30 import androidx.compose.runtime.mutableStateOf
31 import androidx.compose.runtime.remember
32 import androidx.compose.runtime.setValue
33 import androidx.compose.ui.Alignment
34 import androidx.compose.ui.Modifier
35 import androidx.compose.ui.draw.alpha
36 import androidx.compose.ui.platform.LocalContext
37 import androidx.compose.ui.res.stringResource
38 import androidx.compose.ui.text.style.TextAlign
39 import androidx.compose.ui.unit.dp
40 import androidx.lifecycle.compose.collectAsStateWithLifecycle
41 import com.github.nullptroma.wallenc.ui.R
42 import com.github.nullptroma.wallenc.ui.elements.StorageTree
```

```

43 import com.github.nullptroma.wallenc.ui.resources.UserNotification
44 import java.util.UUID
45
46 @Composable
47 fun VaultBrowserScreen(
48     modifier: Modifier = Modifier,
49     viewModel: AbstractVaultBrowserViewModel,
50     onOpenStorageHome: (String) -> Unit,
51 ) {
52     val uiState by viewModel.state.collectAsStateWithLifecycle()
53     val context = LocalContext.current
54     var pendingNotification by remember { mutableStateOf<UserNotification?>(null) }
55
56     LaunchedEffect(viewModel) {
57         viewModel.userNotifications.collect { notification ->
58             pendingNotification = notification
59         }
60     }
61     val notificationText = when (val notification = pendingNotification) {
62         is UserNotification.TextRes -> {
63             if (notification.formatArgs.isEmpty()) {
64                 stringResource(notification.id)
65             } else {
66                 stringResource(notification.id, *notification.formatArgs.toTypedArray())
67             }
68         }
69         is UserNotification.Plain -> notification.message
70         null -> null
71     }
72     LaunchedEffect(notificationText) {
73         val text = notificationText ?: return@LaunchedEffect
74         Toast.makeText(context, text, Toast.LENGTH_SHORT).show()
75         pendingNotification = null
76     }
77
78     val fabEnabled = uiState.addStorageFabEnabled
79     val fabBusy = uiState.vaultListMutationActive
80     val showFullscreenLoader = uiState.storagesList.isEmpty() && uiState.storagesRefreshing
81     val showEmptyState = uiState.storagesList.isEmpty() && !uiState.storagesRefreshing
82     val showRescan = viewModel.supportsStorageRescan
83     val rescanEnabled = showRescan &&
84         !uiState.vaultListMutationActive &&
85         !uiState.storagesRefreshing
86     val isUuidBusy: (UUID) -> Boolean = { uuid -> uuid in uiState.busyStorageUuids }
87

```

```

88     val addFab: @Composable () -> Unit = {
89         FloatingActionButton(
90             onClick = {
91                 if (fabEnabled && !fabBusy) {
92                     viewModel.createStorage()
93                 }
94             },
95             modifier = Modifier.alpha(if (fabEnabled && !fabBusy) 1f else 0.38f),
96         ) {
97             Icon(
98                 Icons.Filled.Add,
99                 contentDescription = stringResource(
100                     when {
101                         !fabEnabled -> R.string.vault_fab_add_storage_disabled_cd
102                         fabBusy -> R.string.vault_fab_add_storage_busy_cd
103                         else -> R.string.vault_fab_add_storage_cd
104                     },
105                 ),
106             )
107         }
108     }
109
110     val vaultContent: @Composable (androidx.compose.foundation.layout.PaddingValues) -> Unit
111     = { innerPadding ->
112         Column(
113             modifier = Modifier
114                 .padding(innerPadding)
115                 .fillMaxSize(),
116         ) {
117             uiState.header?.let { header ->
118                 Row(
119                     modifier = Modifier
120                         .fillMaxWidth()
121                         .padding(horizontal = 16.dp, vertical = 12.dp),
122                     horizontalArrangement = Arrangement.SpaceBetween,
123                     verticalAlignment = Alignment.CenterVertically,
124                 ) {
125                     Column(modifier = Modifier.weight(1f)) {
126                         Text(
127                             text = stringResource(header.titleResId),
128                             style = MaterialTheme.typography.headlineSmall,
129                             color = MaterialTheme.colorScheme.onSurface,
130                         )
131                         header.subtitle?.let { subtitle ->
132                             Spacer(modifier = Modifier.height(4.dp))

```

```

132         Text(
133             text = subtitle,
134             style = MaterialTheme.typography.bodyMedium,
135             color = MaterialTheme.colorScheme.onSurfaceVariant,
136         )
137     }
138 }
139 if (showRescan) {
140     FilledTonalButton(
141         onClick = { viewModel.rescanStorages() },
142         enabled = rescanEnabled,
143     ) {
144         Text(stringResource(R.string.vault_rescan_storages_action))
145     }
146 }
147 }
148 }
149 if (!fabEnabled) {
150     Text(
151         text = stringResource(R.string.vault_unavailable_banner),
152         style = MaterialTheme.typography.bodySmall,
153         color = MaterialTheme.colorScheme.onSurfaceVariant,
154         modifier = Modifier
155             .fillMaxWidth()
156             .padding(horizontal = 12.dp, vertical = 6.dp),
157     )
158 }
159 Box(
160     modifier = Modifier
161         .weight(1f)
162         .fillMaxWidth(),
163 ) {
164     when {
165         showEmptyState -> {
166             Text(
167                 text = stringResource(
168                     if (showRescan) {
169                         R.string.vault_empty_list_hint_remote
170                     } else {
171                         R.string.vault_empty_list_hint
172                     },
173                 ),
174                 style = MaterialTheme.typography.bodyLarge,
175                 color = MaterialTheme.colorScheme.onSurfaceVariant,

```

```

176             textAlign = TextAlign.Center,
177             modifier = Modifier
178                 .align(Alignment.Center)
179                 .padding(horizontal = 24.dp),
180         )
181     }
182     else -> {
183         LazyColumn(
184             modifier = Modifier.fillMaxSize(),
185         ) {
186             items(uiState.storagesList) { listItem ->
187                 StorageTree(
188                     modifier = Modifier.padding(8.dp, 8.dp, 8.dp, 0.dp),
189                     tree = listItem,
190                     isUuidBusy = isUuidBusy,
191                     onClick = { onOpenStorageHome(it.value.uuid.toString()) },
192                     onRename = { tree, newName ->
193                         viewModel.rename(tree.value, newName) },
194                     onRemove = { tree -> viewModel.remove(tree.value) },
195                     onEncrypt = { tree, password, encryptPath,
196                         rememberPassword ->
197                             viewModel.enableEncryption(
198                                 tree.value,
199                                 password,
200                                 encryptPath,
201                                 rememberPassword,
202                             )
203                         },
204                     onOpenEncrypted = { tree, password, remember ->
205                         viewModel.openEncryptedStorage(tree.value, password,
206                         remember)
207                     },
208                     onCloseEncrypted = { tree ->
209                         viewModel.closeEncryptedStorage(tree.value) },
210                     onDisableEncryption = { tree ->
211                         viewModel.disableEncryption(tree.value) },
212                     getStatusTextRes = { tree ->
213                         viewModel.getStorageStatusRes(tree.value) },
214                     isEncryptionOpened = { tree ->
215                         viewModel.isEncryptionSessionOpen(tree.value) },
216                     isStorageSyncLockHeld = { info ->
217                         viewModel.isStorageSyncLockHeld(info) },
218                     onClearStorageSyncLock = { info ->
219                         viewModel.clearStorageSyncLock(info) },
220                 )
221             }
222         }
223     }

```

```

214         Spacer(modifier = Modifier.height(8.dp))
215     }
216 }
217 }
218 }
219 }
220 }
221 }
222
223 Box(modifier = modifier) {
224     Scaffold(
225         contentWindowInsets = WindowInsets(0.dp),
226         floatingActionButton = addFab,
227         content = vaultContent,
228     )
229
230     if (showFullscreenLoader) {
231         Box(modifier = Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
232             Box(
233                 modifier = Modifier
234                     .fillMaxSize()
235                     .background(MaterialTheme.colorScheme.scrim),
236             )
237             Column(
238                 horizontalAlignment = Alignment.CenterHorizontally,
239                 verticalArrangement = Arrangement.spacedBy(12.dp),
240             ) {
241                 CircularProgressIndicator(
242                     modifier = Modifier.size(64.dp),
243                     color = MaterialTheme.colorScheme.primary,
244                     trackColor = MaterialTheme.colorScheme.surfaceVariant,
245                 )
246                 Text(
247                     text = stringResource(R.string.vault_loading_storages),
248                     style = MaterialTheme.typography.bodyLarge,
249                     color = MaterialTheme.colorScheme.onSurface,
250                     textAlign = TextAlign.Center,
251                     modifier = Modifier.padding(horizontal = 24.dp),
252                 )
253             }
254         }
255     }
256 }
257 }
258

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/vault/VaultBrowserScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.vault
2
3 import com.github.nullptroma.wallenc.domain.datatypes.Tree
4 import com.github.nullptroma.wallenc.domain.interfaces.IStorageInfo
5 import java.util.UUID
6
7 data class VaultBrowserScreenState(
8     val header: VaultBrowserHeader? = null,
9     val storagesList: List<Tree<IStorageInfo>>,
10    /** Первый снимок списка storages ещё не получен (удалённый vault). */
11    val storagesRefreshing: Boolean,
12    /** Storages с активной задачей в
13    [com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator]. */
14    val busyStorageUuids: Set<UUID>,
15    /** Активна задача, меняющая состав списка storages (создание и т.п.). */
16    val vaultListMutationActive: Boolean,
17    val addStorageFabEnabled: Boolean = false,
18 )
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/
settings/SettingsRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.settings
2
3 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4 import kotlinx.parcelize.Parcelize
5 import kotlinx.serialization.Serializable
6
7 @Serializable
8 @Parcelize
9 class SettingsRoute: ScreenRoute()
10
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/
settings/SettingsScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.settings
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Row
6 import androidx.compose.foundation.layout.fillMaxSize
7 import androidx.compose.foundation.layout.fillMaxWidth
8 import androidx.compose.foundation.layout.padding
9 import androidx.compose.foundation.selection.selectable
10 import androidx.compose.foundation.selection.selectableGroup
11 import androidx.compose.material3.MaterialTheme
12 import androidx.compose.material3.RadioButton
13 import androidx.compose.material3.Text
14 import androidx.compose.runtime.Composable
15 import androidx.compose.runtime.getValue
16 import androidx.compose.ui.Alignment
17 import androidx.compose.ui.Modifier
18 import androidx.compose.ui.res.stringResource
19 import androidx.compose.ui.semantics.Role
20 import androidx.compose.ui.unit.dp
21 import androidx.lifecycle.compose.collectAsStateWithLifecycle
22 import com.github.nullptroma.wallenc.ui.R
23 import com.github.nullptroma.wallenc.ui.locale.AppLanguage
24
25 @Composable
26 fun SettingsScreen(modifier: Modifier, viewModel: SettingsViewModel) {
27     val state by viewModel.state.collectAsStateWithLifecycle()
28
29     Column(
30         modifier = modifier
31             .fillMaxSize()
32             .padding(16.dp),
33         verticalArrangement = Arrangement.spacedBy(16.dp),
34     ) {
35         Text(
36             text = stringResource(id = R.string.settings_title),
37             style = MaterialTheme.typography.headlineSmall,
38         )
39         Text(
40             text = stringResource(R.string.settings_language_section),
41             style = MaterialTheme.typography.titleMedium,
42         )
43     }
44 }
```

```

43         Column(Modifier.selectableGroup()) {
44             LanguageOption(
45                 label = stringResource(R.string.settings_language_system),
46                 selected = state.selectedLanguage == AppLanguage.System,
47                 onClick = { viewModel.setLanguage(AppLanguage.System) },
48             )
49             LanguageOption(
50                 label = stringResource(R.string.settings_language_english),
51                 selected = state.selectedLanguage == AppLanguage.English,
52                 onClick = { viewModel.setLanguage(AppLanguage.English) },
53             )
54             LanguageOption(
55                 label = stringResource(R.string.settings_language_russian),
56                 selected = state.selectedLanguage == AppLanguage.Russian,
57                 onClick = { viewModel.setLanguage(AppLanguage.Russian) },
58             )
59         }
60     }
61 }
62
63 @Composable
64 private fun LanguageOption(
65     label: String,
66     selected: Boolean,
67     onClick: () -> Unit,
68 ) {
69     Row(
70         Modifier
71             .fillMaxWidth()
72             .selectable(
73                 selected = selected,
74                 onClick = onClick,
75                 role = Role.RadioButton,
76             )
77         .padding(vertical = 8.dp),
78         verticalAlignment = Alignment.CenterVertically,
79     ) {
80         RadioButton(selected = selected, onClick = onClick)
81         Text(
82             text = label,
83             style = MaterialTheme.typography.bodyLarge,
84             modifier = Modifier.padding(start = 8.dp),
85         )
86     }

```

87 }
88

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/
settings/SettingsScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.settings
2
3 import com.github.nullptroma.wallenc.ui.locale.AppLanguage
4
5 data class SettingsScreenState(
6     val selectedLanguage: AppLanguage = AppLanguage.System,
7 )
8
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/
settings/SettingsViewModel.kt

```
1  package com.github.nullptroma.wallenc.ui.screens.settings
2
3  import androidx.lifecycle.ViewModelScope
4  import com.github.nullptroma.wallenc.ui.ViewModelBase
5  import com.github.nullptroma.wallenc.ui.locale.AppLanguage
6  import com.github.nullptroma.wallenc.ui.locale.AppLocaleController
7  import dagger.hilt.android.lifecycle.HiltViewModel
8  import kotlinx.coroutines.launch
9  import javax.inject.Inject
10
11  @HiltViewModel
12  class SettingsViewModel @Inject constructor(
13      private val appLocaleController: AppLocaleController,
14  ) : ViewModelBase<SettingsScreenState>(SettingsScreenState()) {
15
16      init {
17          viewModelScope.launch {
18              appLocaleController.language.collect { language ->
19                  updateState(state.value.copy(selectedLanguage = language))
20              }
21          }
22      }
23
24      fun setLanguage(language: AppLanguage) {
25          viewModelScope.launch {
26              appLocaleController.setLanguage(language)
27          }
28      }
29  }
30
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/
shared/TextEditRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.shared
2
3 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4 import kotlinx.parcelize.Parcelize
5 import kotlinx.serialization.Serializable
6
7 @Serializable
8 @Parcelize
9 data class TextEditRoute(val text: String): ScreenRoute()
10
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/
shared/TextEditScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.shared
2
3 import androidx.compose.material3.MaterialTheme
4 import androidx.compose.material3.Text
5 import androidx.compose.runtime.Composable
6 import androidx.compose.ui.Modifier
7 import androidx.compose.ui.res.stringResource
8 import com.github.nullptroma.wallenc.ui.R
9 import com.github.nullptroma.wallenc.ui.elements.WallencScreenContentPadding
10 import com.github.nullptroma.wallenc.ui.elements.WallencScreenScaffold
11
12 @Composable
13 fun TextEditScreen(
14     text: String,
15     modifier: Modifier = Modifier,
16 ) {
17     WallencScreenScaffold(modifier = modifier) { innerPadding ->
18         WallencScreenContentPadding(innerPadding) {
19             Text(
20                 text = stringResource(R.string.text_edit_screen_placeholder, text),
21                 style = MaterialTheme.typography.bodyLarge,
22             )
23         }
24     }
25 }
26
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/sync/
StorageSyncRoute.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.sync
2
3 import com.github.nullptroma.wallenc.ui.screens.ScreenRoute
4 import kotlinx.parcelize.Parcelize
5 import kotlinx.serialization.Serializable
6
7 @Serializable
8 @Parcelize
9 class StorageSyncRoute : ScreenRoute()
10
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/sync/ StorageSyncScreen.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.sync
2
3 import androidx.compose.foundation.layout.Arrangement
4 import androidx.compose.foundation.layout.Box
5 import androidx.compose.foundation.layout.Column
6 import androidx.compose.foundation.layout.Row
7 import androidx.compose.foundation.layout.WindowInsets
8 import androidx.compose.foundation.layout.fillMaxSize
9 import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.padding
11 import androidx.compose.foundation.lazy.LazyColumn
12 import androidx.compose.foundation.lazy.items
13 import androidx.compose.foundation.clickable
14 import androidx.compose.material.icons.Icons
15 import androidx.compose.material.icons.automirrored.rounded.ArrowBack
16 import androidx.compose.material.icons.rounded.Add
17 import androidx.compose.material.icons.rounded.Delete
18 import androidx.compose.material.icons.rounded.ExpandLess
19 import androidx.compose.material.icons.rounded.ExpandMore
20 import androidx.compose.material.icons.rounded.Sync
21 import androidx.compose.material3.AlertDialog
22 import androidx.compose.material3.Button
23 import androidx.compose.material3.Card
24 import androidx.compose.material3.CardDefaults
25 import androidx.compose.material3.FloatingActionButton
26 import androidx.compose.material3.HorizontalDivider
27 import androidx.compose.material3.Icon
28 import androidx.compose.material3.IconButton
29 import androidx.compose.material3.LinearProgressIndicator
30 import androidx.compose.material3.MaterialTheme
31 import androidx.compose.material3.SnackbarHost
32 import androidx.compose.material3.SnackbarHostState
33 import androidx.compose.material3.Scaffold
34 import androidx.compose.material3.Text
35 import androidx.compose.runtime.Composable
36 import androidx.compose.runtime.LaunchedEffect
37 import androidx.compose.runtime.getValue
38 import androidx.compose.runtime.mutableStateOf
39 import androidx.compose.runtime.remember
40 import androidx.compose.runtime.setValue
41 import androidx.compose.ui.Alignment
42 import androidx.compose.ui.Modifier
```

```

43 import androidx.compose.foundation.layout.navigationBarsPadding
44 import androidx.compose.ui.draw.alpha
45 import androidx.compose.ui.res.stringResource
46 import androidx.compose.ui.text.font.FontFamily
47 import androidx.compose.ui.unit.dp
48 import androidx.hilt.lifecycle.viewmodel.compose.hiltViewModel
49 import androidx.lifecycle.compose.collectAsStateWithLifecycle
50 import com.github.nullptroma.wallenc.ui.R
51 import com.github.nullptroma.wallenc.ui.elements.AnimatedFloatingBackButton
52 import com.github.nullptroma.wallenc.ui.resources.UserNotification
53 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroupEncryptionKind
54 import java.util.UUID
55
56 @Composable
57 fun StorageSyncScreen(
58     modifier: Modifier = Modifier,
59     viewModel: StorageSyncViewModel = hiltViewModel(),
60 ) {
61     val state by viewModel.state.collectAsStateWithLifecycle()
62     val snackbarHostState = remember { SnackbarHostState() }
63     val userMessageText = when (val um = state.userMessage) {
64         is UserNotification.TextRes -> {
65             if (um.formatArgs.isEmpty()) {
66                 stringResource(um.id)
67             } else {
68                 stringResource(um.id, *um.formatArgs.toTypedArray())
69             }
70         }
71         is UserNotification.Plain -> um.message
72         null -> null
73     }
74     LaunchedEffect(state.userMessage) {
75         val text = userMessageText ?: return@LaunchedEffect
76         snackbarHostState.showSnackbar(text)
77         viewModel.consumeUserMessage()
78     }
79
80     var pendingRemoveGroupId by remember { mutableStateOf<String?>(null) }
81     var pendingRemoveStorage by remember { mutableStateOf<Pair<String, UUID>?>(null) }
82     val pickerGroupId = state.pickerGroupId
83     if (pickerGroupId != null) {
84         StoragePickerScreen(
85             modifier = modifier,
86             state = state,
87             groupId = pickerGroupId,

```

```

88         onBack = viewModel::closePicker,
89         onAddStorage = viewModel::addStorageToCurrentGroup,
90         onToggleVault = viewModel::toggleVaultExpanded,
91         snackbarHostState = snackbarHostState,
92     )
93     return
94 }
95
96 val storageByUuid = state.vaults
97     .flatMap { vault -> flattenStorageTree(vault.storages) }
98     .associateBy { it.uuid }
99
100 val groupEditLocked = state.isBusy || state.isStorageSyncRunning
101
102 Scaffold(
103     modifier = modifier,
104     contentWindowInsets = WindowInsets(0.dp),
105     snackbarHost = { SnackbarHost(snackbarHostState) },
106     floatingActionButton = {
107         FloatingActionButton(
108             onClick = { if (!groupEditLocked) viewModel.createGroup() },
109             modifier = Modifier.alpha(if (groupEditLocked) 0.38f else 1f),
110         ) {
111             Icon(
112                 imageVector = Icons.Rounded.Add,
113                 contentDescription = stringResource(R.string.sync_fab_create_group_cd),
114             )
115         }
116     },
117 ) { inner ->
118     Column(
119         modifier = Modifier
120             .padding(inner)
121             .fillMaxSize()
122             .padding(16.dp),
123         verticalArrangement = Arrangement.spacedBy(12.dp),
124     ) {
125         Row(
126             horizontalArrangement = Arrangement.spacedBy(8.dp),
127             verticalAlignment = Alignment.CenterVertically,
128         ) {
129             Button(
130                 onClick = viewModel::runSyncNow,
131                 enabled = !groupEditLocked && !state.anyVaultStoragesScanning,
132             ) {

```

```

133         Icon(
134             Icons.Rounded.Sync,
135             contentDescription = null,
136             modifier = Modifier.padding(end = 8.dp),
137         )
138         Text(stringResource(id = R.string.sync_run_now))
139     }
140 }
141
142 when {
143     state.isStorageSyncRunning -> {
144         Column(
145             modifier = Modifier.fillMaxWidth(),
146             verticalArrangement = Arrangement.spacedBy(6.dp),
147         ) {
148             Text(
149                 text = stringResource(R.string.sync_progress_section_title),
150                 style = MaterialTheme.typography.titleSmall,
151             )
152             state.storageSyncProgressLabel?.let { line ->
153                 Text(
154                     text = line,
155                     style = MaterialTheme.typography.bodySmall,
156                     color = MaterialTheme.colorScheme.onSurfaceVariant,
157                 )
158             }
159             val frac = state.storageSyncProgressFraction
160             if (frac != null) {
161                 LinearProgressIndicator(
162                     progress = { frac },
163                     modifier = Modifier.fillMaxWidth(),
164                 )
165             } else {
166                 LinearProgressIndicator(modifier = Modifier.fillMaxWidth())
167             }
168         }
169     }
170     state.isBusy -> {
171         Column(
172             modifier = Modifier.fillMaxWidth(),
173             verticalArrangement = Arrangement.spacedBy(6.dp),
174         ) {
175             Text(
176                 text = stringResource(R.string.sync_groups_busy_section_title),

```

```

177         style = MaterialTheme.typography.titleSmall,
178     )
179     LinearProgressIndicator(modifier = Modifier.fillMaxWidth())
180 }
181 }
182 }
183
184 Text(
185     text = stringResource(id = R.string.sync_groups_title),
186     style = MaterialTheme.typography.titleMedium,
187 )
188 LazyColumn(
189     modifier = Modifier.fillMaxSize(),
190     verticalArrangement = Arrangement.spacedBy(10.dp),
191 ) {
192     items(state.groups, key = { it.id }) { group ->
193         Card(
194             modifier = Modifier
195                 .fillMaxWidth()
196                 .padding(vertical = 2.dp),
197             colors = CardDefaults.cardColors(
198                 containerColor = MaterialTheme.colorScheme.surfaceContainerHigh,
199             ),
200         ) {
201             Column(
202                 modifier = Modifier
203                     .fillMaxWidth()
204                     .padding(12.dp),
205                 verticalArrangement = Arrangement.spacedBy(8.dp),
206             ) {
207                 Row(
208                     modifier = Modifier.fillMaxWidth(),
209                     horizontalArrangement = Arrangement.SpaceBetween,
210                     verticalAlignment = Alignment.Top,
211                 ) {
212                     Text(
213                         text = group.id,
214                         style = MaterialTheme.typography.titleMedium,
215                         modifier = Modifier
216                             .weight(1f)
217                             .padding(end = 4.dp),
218                         maxLines = 4,
219                     )
220                     IconButton(
221                         onClick = { pendingRemoveGroupId = group.id },

```

```

222         enabled = !groupEditLocked,
223     ) {
224         Icon(
225             imageVector = Icons.Rounded.Delete,
226             contentDescription = stringResource(id =
R.string.sync_remove_group),
227         )
228     }
229 }
230
231 if (group.storageUuids.isEmpty()) {
232     Text(
233         text = stringResource(id = R.string.sync_group_empty),
234         style = MaterialTheme.typography.bodySmall,
235         color = MaterialTheme.colorScheme.onSurfaceVariant,
236     )
237 } else {
238     if (group.incompatibleStorageUuids.isNotEmpty()) {
239         Text(
240             text = stringResource(
241                 id = R.string.sync_group_incompatible_warning,
242                 group.incompatibleStorageUuids.size,
243             ),
244             style = MaterialTheme.typography.bodySmall,
245             color = MaterialTheme.colorScheme.error,
246         )
247     }
248     Text(
249         text = stringResource(
250             R.string.sync_group_policy_line,
251             groupPolicyLabel(group.encryptionKind),
252         ),
253         style = MaterialTheme.typography.bodySmall,
254         color = MaterialTheme.colorScheme.onSurfaceVariant,
255     )
256
257     group.storageUuids.forEach { storageUuid ->
258         val storage = storageByUuid[storageUuid]
259         val titleText = storage?.name
260         stringResource(R.string.sync_storage_missing_title)
261         val encLabel =
encryptionKindLabel(storage?.encryptionKind)
262         val statusLine = when {
263             storage != null && !storage.isReachable ->

```

```

264     stringResource(R.string.sync_storage_unreachable)
265         storage == null && state.anyVaultStoragesScanning ->
266     stringResource(R.string.sync_storage_pending_vault_scan)
267         storage == null && !state.anyVaultStoragesScanning -
268     stringResource(R.string.sync_storage_not_in_vaults)
269         else -> null
270     }
271     val statusColor = when {
272         storage != null && !storage.isReachable ->
273             MaterialTheme.colorScheme.error
274         storage == null ->
275             MaterialTheme.colorScheme.tertiary
276         else -> MaterialTheme.colorScheme.onSurfaceVariant
277     }
278     Card(
279         modifier = Modifier.fillMaxWidth(),
280         colors = CardDefaults.cardColors(
281             MaterialTheme.colorScheme.surfaceContainer,
282             ),
283     ) {
284         Row(
285             modifier = Modifier
286                 .fillMaxWidth()
287                 .padding(12.dp),
288             horizontalArrangement =
289                 Arrangement.SpaceBetween,
290             verticalAlignment = Alignment.Top,
291         ) {
292             Column(
293                 modifier = Modifier.weight(1f),
294                 verticalArrangement =
295                     Arrangement.spacedBy(4.dp),
296             ) {
297                 Text(
298                     text = titleText,
299                     style =
300                         MaterialTheme.typography.titleSmall,
301                 )
302                 Text(
303                     text = storageUuid.toString(),
304                     style =
305                         MaterialTheme.typography.bodySmall,
306                     fontFamily = FontFamily.Monospace,

```



```

303         MaterialTheme.colorScheme.onSurfaceVariant,
304         )
305         statusLine?.let { line ->
306             Text(
307                 text = line,
308                 style =
309                     MaterialTheme.typography.bodySmall,
310                     color = statusColor,
311                 )
312             }
313             Text(
314                 text = stringResource(
315                     R.string.sync_storage_encryption_line,
316                     enclabel,
317                 ),
318                 style =
319                     MaterialTheme.typography.bodySmall,
320                 )
321             }
322             IconButton(
323                 to storageUuid },
324                 onClick = { pendingRemoveStorage = group.id
325                 enabled = !groupEditLocked,
326                 ) {
327                 Icon(
328                     R.string.sync_remove_storage),
329                     imageVector = Icons.Rounded.Delete,
330                     contentDescription = stringResource(id =
331                     )
332                 )
333             }
334         }
335     }
336 }
337
338 HorizontalDivider(modifier = Modifier.padding(top = 4.dp))
339
340 Row(
341     modifier = Modifier.fillMaxWidth(),
342     horizontalArrangement = Arrangement.End,
343 ) {
344     IconButton(
345         onClick = { viewModel.openPicker(group.id) },
346         enabled = !groupEditLocked,
347     ) {

```

```

344             Icon(
345                 imageView = Icons.Rounded.Add,
346                 contentDescription = stringResource(id =
R.string.sync_add_storage),
347             )
348         }
349     }
350 }
351 }
352 }
353 }
354 }
355 }
356
357 if (pendingRemoveGroupId != null) {
358     AlertDialog(
359         onDismissRequest = { pendingRemoveGroupId = null },
360         title = { Text(text = stringResource(id =
R.string.sync_remove_group_confirm_title)) },
361         text = {
362             Text(
363                 text = stringResource(
364                     id = R.string.sync_remove_group_confirm_message,
365                     pendingRemoveGroupId.orEmpty(),
366                 ),
367             )
368         },
369         confirmButton = {
370             Button(
371                 onClick = {
372                     val groupId = pendingRemoveGroupId ?: return@Button
373                     viewModel.removeGroup(groupId)
374                     pendingRemoveGroupId = null
375                 },
376             ) {
377                 Text(text = stringResource(id = R.string.sync_confirm_delete))
378             }
379         },
380         dismissButton = {
381             Button(onClick = { pendingRemoveGroupId = null }) {
382                 Text(text = stringResource(id = R.string.sync_cancel))
383             }
384         },
385     )
386 }
387

```

```

388         if (pendingRemoveStorage != null) {
389             AlertDialog(
390                 onDismissRequest = { pendingRemoveStorage = null },
391                 title = { Text(text = stringResource(id =
R.string.sync_remove_storage_confirm_title)) },
392                 text = {
393                     Text(
394                         text = stringResource(
395                             id = R.string.sync_remove_storage_confirm_message,
396                             pendingRemoveStorage?.second.toString(),
397                         ),
398                     )
399                 },
400                 confirmButton = {
401                     Button(
402                         onClick = {
403                             val payload = pendingRemoveStorage ?: return@Button
404                             viewModel.removeStorageFromGroup(payload.first, payload.second)
405                             pendingRemoveStorage = null
406                         },
407                     ) {
408                         Text(text = stringResource(id = R.string.sync_confirm_delete))
409                     }
410                 },
411                 dismissButton = {
412                     Button(onClick = { pendingRemoveStorage = null }) {
413                         Text(text = stringResource(id = R.string.sync_cancel))
414                     }
415                 },
416             )
417         }
418     }
419
420     @Composable
421     private fun encryptionKindLabel(kind: StorageSyncEncryptionKind?): String {
422         return when (kind) {
423             null -> stringResource(R.string.sync_encryption_unknown)
424             StorageSyncEncryptionKind.NotEncrypted ->
stringResource(R.string.enc_status_not_encrypted)
425             StorageSyncEncryptionKind.EncryptedOpened ->
stringResource(R.string.enc_status_encrypted_open)
426             StorageSyncEncryptionKind.EncryptedClosed ->
stringResource(R.string.enc_status_encrypted)
427         }
428     }
429

```

```

430 @Composable
431 private fun StoragePickerScreen(
432     modifier: Modifier,
433     state: StorageSyncScreenState,
434     groupId: String,
435     onBack: () -> Unit,
436     onAddStorage: (UUID) -> Unit,
437     onToggleVault: (UUID) -> Unit,
438     snackbarHostState: SnackbarHostState,
439 ) {
440     val selected = state.groups.firstOrNull { it.id == groupId }?.storageUuids ?: emptySet()
441     val groupEditLocked = state.isBusy || state.isStorageSyncRunning
442     Scaffold(
443         modifier = modifier,
444         contentWindowInsets = WindowInsets(0.dp),
445         snackbarHost = { SnackbarHost(snackbarHostState) },
446     ) { inner ->
447         Box(
448             modifier = Modifier
449                 .padding(inner)
450                 .fillMaxSize(),
451         ) {
452             Column(
453                 modifier = Modifier
454                     .fillMaxSize()
455                     .padding(horizontal = 16.dp, vertical = 12.dp),
456                 verticalArrangement = Arrangement.spacedBy(12.dp),
457             ) {
458                 Text(
459                     text = stringResource(id = R.string.sync_picker_title, groupId),
460                     style = MaterialTheme.typography.titleLarge,
461                 )
462
463                 LazyColumn(
464                     modifier = Modifier.fillMaxSize(),
465                     verticalArrangement = Arrangement.spacedBy(10.dp),
466                 ) {
467                     items(state.vaults, key = { it.uuid }) { vault ->
468                         Card(
469                             modifier = Modifier
470                                 .fillMaxWidth()
471                                 .padding(vertical = 2.dp),
472                             colors = CardDefaults.cardColors(
473                                 containerColor = MaterialTheme.colorScheme.surfaceContainerHigh,

```

```

474         ),
475     ) {
476         Column(
477             modifier = Modifier
478                 .fillMaxWidth()
479                 .padding(10.dp),
480             verticalArrangement = Arrangement.spacedBy(6.dp),
481         ) {
482             val expanded = vault.uuid in state.expandedVaultUuids
483             Row(
484                 modifier = Modifier
485                     .fillMaxWidth()
486                     .clickable { onToggleVault(vault.uuid) },
487                 horizontalArrangement = Arrangement.SpaceBetween,
488                 verticalAlignment = Alignment.CenterVertically,
489             ) {
490                 Text(
491                     text = vault.title,
492                     style = MaterialTheme.typography.titleSmall,
493                     modifier = Modifier.weight(1f),
494                 )
495                 Icon(
496                     Icons.Rounded.ExpandMore,
497                     imageVector = if (expanded) Icons.Rounded.ExpandLess else
498                         contentDescription = if (expanded) {
499                             stringResource(R.string.sync_picker_collapse)
500                         } else {
501                             stringResource(R.string.sync_picker_expand)
502                         },
503                 )
504             }
505             Text(
506                 text = vault.type,
507                 style = MaterialTheme.typography.bodySmall,
508                 color = MaterialTheme.colorScheme.onSurfaceVariant,
509             )
510             Text(
511                 text = vault.uuid.toString(),
512                 style = MaterialTheme.typography.bodySmall,
513                 fontFamily = FontFamily.Monospace,
514                 color = MaterialTheme.colorScheme.onSurfaceVariant,
515             )
516             if (expanded) {
517                 if (vault.storages.isEmpty()) {

```

```

518         Text(
519             R.string.sync_picker_no_storages),
520             style = MaterialTheme.typography.bodySmall,
521         )
522     } else {
523         vault.storages.forEach { storage ->
524             StoragePickerNode(
525                 node = storage,
526                 depth = 0,
527                 selected = selected,
528                 isBusy = groupEditLocked,
529                 onAddStorage = onAddStorage,
530             )
531         }
532     }
533 }
534 }
535 }
536 }
537 }
538 }
539     AnimatedFloatingBackButton(
540         visible = true,
541         onClick = onBack,
542         modifier = Modifier
543             .align(Alignment.BottomStart)
544             .navigationBarsPadding()
545             .padding(start = 12.dp, bottom = 12.dp),
546     )
547 }
548 }
549 }
550
551 @Composable
552 private fun StoragePickerNode(
553     node: StorageSyncStorageUi,
554     depth: Int,
555     selected: Set<UUID>,
556     isBusy: Boolean,
557     onAddStorage: (UUID) -> Unit,
558 ) {
559     val isSelected = node.uuid in selected
560     Card(
561         modifier = Modifier

```

```

562         .fillMaxWidth()
563         .padding(start = (depth * 14).dp),
564     colors = CardDefaults.cardColors(
565         containerColor = MaterialTheme.colorScheme.surfaceContainer,
566     ),
567 ) {
568     Row(
569         modifier = Modifier
570             .fillMaxWidth()
571             .padding(8.dp),
572         horizontalArrangement = Arrangement.SpaceBetween,
573         verticalAlignment = Alignment.Top,
574     ) {
575         Column(
576             modifier = Modifier.weight(1f),
577             verticalArrangement = Arrangement.spacedBy(4.dp),
578         ) {
579             Text(
580                 text = node.name,
581                 style = MaterialTheme.typography.bodyMedium,
582             )
583             Text(
584                 text = node.uuid.toString(),
585                 style = MaterialTheme.typography.bodySmall,
586                 fontFamily = FontFamily.Monospace,
587                 color = MaterialTheme.colorScheme.onSurfaceVariant,
588             )
589             Text(
590                 text = stringResource(
591                     R.string.sync_storage_encryption_line,
592                     encryptionKindLabel(node.encryptionKind),
593                 ),
594                 style = MaterialTheme.typography.bodySmall,
595             )
596             if (!node.isReachable) {
597                 Text(
598                     text = stringResource(R.string.sync_storage_unreachable),
599                     style = MaterialTheme.typography.bodySmall,
600                     color = MaterialTheme.colorScheme.error,
601                 )
602             }
603         }
604         IconButton(
605             enabled = !isSelected && !isBusy,
606             onClick = { onAddStorage(node.uuid) },

```

```

607         ) {
608             Icon(
609                 imageVector = Icons.Rounded.Add,
610                 contentDescription = stringResource(
611                     if (isSelected) R.string.sync_picker_added else
R.string.sync_picker_cd_add,
612                 ),
613             )
614         }
615     }
616 }
617 node.children.forEach { child ->
618     StoragePickerNode(
619         node = child,
620         depth = depth + 1,
621         selected = selected,
622         isBusy = isBusy,
623         onAddStorage = onAddStorage,
624     )
625 }
626 }
627
628 private fun flattenStorageTree(nodes: List<StorageSyncStorageUi>):
List<StorageSyncStorageUi> {
629     return nodes.flatMap { node ->
630         listOf(node) + flattenStorageTree(node.children)
631     }
632 }
633
634 @Composable
635 private fun groupPolicyLabel(kind: StorageSyncGroupEncryptionKind): String {
636     return when (kind) {
637         StorageSyncGroupEncryptionKind.UNSET ->
stringResource(R.string.sync_group_policy_unset)
638         StorageSyncGroupEncryptionKind.NONE ->
stringResource(R.string.sync_group_policy_plain)
639         StorageSyncGroupEncryptionKind.PASSWORD ->
stringResource(R.string.sync_group_policy_password)
640     }
641 }
642

```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/sync/
StorageSyncScreenState.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.sync
2
3 import com.github.nullptroma.wallenc.ui.resources.UserNotification
4 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroupEncryptionKind
5 import java.util.UUID
6
7 enum class StorageSyncEncryptionKind {
8     NotEncrypted,
9     EncryptedOpened,
10    EncryptedClosed,
11 }
12
13 data class StorageSyncStorageUi(
14     val uuid: UUID,
15     val name: String,
16     val encryptionKind: StorageSyncEncryptionKind,
17     /** false, если storage есть в дереве, но сейчас недоступен (например, vault offline).
18 */
19     val isReachable: Boolean = true,
20     val children: List<StorageSyncStorageUi> = emptyList(),
21 )
22
23 data class StorageSyncVaultUi(
24     val uuid: UUID,
25     val title: String,
26     val type: String,
27     val storages: List<StorageSyncStorageUi>,
28 )
29
30 data class StorageSyncGroupUi(
31     val id: String,
32     val storageUuids: Set<UUID>,
33     val encryptionKind: StorageSyncGroupEncryptionKind =
34         StorageSyncGroupEncryptionKind.UNSET,
35     val incompatibleStorageUuids: Set<UUID> = emptySet(),
36 )
37
38 data class StorageSyncScreenState(
39     val groups: List<StorageSyncGroupUi> = emptyList(),
40     val vaults: List<StorageSyncVaultUi> = emptyList(),
41     val expandedVaultUuids: Set<UUID> = emptySet(),
42     val pickerGroupId: String? = null,
43     val isBusy: Boolean = false,
```

```
42      /** Долгая синхронизация storages через пайплайн задач. */
43      val isStorageSyncRunning: Boolean = false,
44      /** Доля прогресса активной синхронизации; null – неопределённый прогресс. */
45      val storageSyncProgressFraction: Float? = null,
46      /** Подпись этапа из пайплайна (если есть). */
47      val storageSyncProgressLabel: String? = null,
48      /** Любой vault ещё загружает список storages – UUID из группы могут появиться позже. */
49      val anyVaultStoragesScanning: Boolean = false,
50      val userMessage: UserNotification? = null,
51  )
52
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/screens/sync/
StorageSyncViewModel.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.sync
2
3 import androidx.lifecycle.ViewModelScope
4 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
5 import com.github.nullptroma.wallenc.domain.interfaces.IStorageMetaInfo
6 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
7 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
8 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
9 import com.github.nullptroma.wallenc.ui.R
10 import com.github.nullptroma.wallenc.ui.ViewModelBase
11 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
12 import com.github.nullptroma.wallenc.ui.resources.resolve
13 import com.github.nullptroma.wallenc.ui.resources.UserNotification
14 import com.github.nullptroma.wallenc.usecases.AddStorageToSyncGroupResult
15 import com.github.nullptroma.wallenc.usecases.ManageStorageSyncGroupsUseCase
16 import com.github.nullptroma.wallenc.domain.tasks.StorageSyncTriggerReason
17 import com.github.nullptroma.wallenc.usecases.RunStorageSyncUseCase
18 import com.github.nullptroma.wallenc.usecases.StorageSyncCompatibilityInput
19 import com.github.nullptroma.wallenc.usecases.isStorageCompatibleWithGroup
20 import com.github.nullptroma.wallenc.vault.contract.DescribedVault
21 import com.github.nullptroma.wallenc.vault.contract.VaultDescriptor
22 import dagger.hilt.android.lifecycle.HiltViewModel
23 import kotlinx.coroutines.ExperimentalCoroutinesApi
24 import kotlinx.coroutines.flow.combine
25 import kotlinx.coroutines.flow.distinctUntilChanged
26 import kotlinx.coroutines.flow.flatMapLatest
27 import kotlinx.coroutines.flow.flowOf
28 import kotlinx.coroutines.flow.map
29 import kotlinx.coroutines.launch
30 import java.util.UUID
31 import javax.inject.Inject
32
33 @HiltViewModel
34 @OptIn(ExperimentalCoroutinesApi::class)
35 class StorageSyncViewModel @Inject constructor(
36     private val groupsUseCase: ManageStorageSyncGroupsUseCase,
37     private val runStorageSyncUseCase: RunStorageSyncUseCase,
38     private val vaultsManager: IVaultsManager,
39     private val taskOrchestrator: ITaskOrchestrator,
40     private val uiStrings: UiStringResolver,
41 ) : ViewModelBase<StorageSyncScreenState>(StorageSyncScreenState()) {
42     private var storageByUuid: Map<UUID, IStorage> = emptyMap()
```

```

43
44     init {
45         refreshGroups()
46         observeVaults()
47         observeStorageSyncPipeline()
48         viewModelScope.launch {
49             combine(
50                 vaultsManager.vaults,
51                 state.map { it.groups },
52             ) { vaults, groups ->
53                 val requiredUuids = groups.flatMap { it.storageUuids }.toSet()
54                 if (requiredUuids.isEmpty() || vaults.isEmpty()) {
55                     false
56                 } else {
57                     val opened = vaultsManager.unlockManager.openedStorages.value
58                     vaults.any { vault ->
59                         val uuidsInVault = vault.storages.value.flatMap { root ->
60                             flattenStorages(buildStorageTree(root, opened))
61                         }.map { it.uuid }.toSet()
62                         uuidsInVault.any { it in requiredUuids } &&
63                             vault.storagesScanInProgress.value
64                     }
65                 }
66             }
67             .distinctUntilChanged()
68             .collect { scanning ->
69                 updateState(state.value.copy(anyVaultStoragesScanning = scanning))
70             }
71         }
72     }
73
74     private fun observeStorageSyncPipeline() {
75         viewModelScope.launch {
76             combine(
77                 runStorageSyncUseCase.syncRunning,
78                 runStorageSyncUseCase.activeSyncTaskId,
79                 taskOrchestrator.pipelineState,
80             ) { syncRunning, taskId, pipe ->
81                 val progress = taskId?.let { id ->
82                     val task = pipe.tasks.find { it.id == id }
83                     (task?.state as? TaskRunState.Running)?.progress
84                 }
85                 Triple(
86                     syncRunning,
87                     progress?.fraction,

```

```

88         progress?.label?.resolve(uiStrings),
89     )
90 }
91     .distinctUntilChanged()
92     .collect { (running, frac, label) ->
93         updateState(
94             state.value.copy(
95                 isStorageSyncRunning = running,
96                 storageSyncProgressFraction = if (running) frac else null,
97                 storageSyncProgressLabel = if (running) label else null,
98             ),
99         )
100     }
101 }
102 }
103
104 fun refreshGroups() {
105     viewModelScope.launch {
106         updateState(state.value.copy(groups = reloadGroupsUi()))
107     }
108 }
109
110 fun createGroup() {
111     viewModelScope.launch {
112         withGroupMutationBusy(clearPicker = true) {
113             val group = groupsUseCase.createGroup()
114             UserNotification.TextRes(
115                 R.string.sync_msg_group_created,
116                 listOf(group.id),
117             )
118         }
119     }
120 }
121
122 fun removeGroup(groupId: String) {
123     viewModelScope.launch {
124         withGroupMutationBusy {
125             groupsUseCase.removeGroup(groupId)
126             UserNotification.TextRes(R.string.sync_msg_group_removed)
127         }
128     }
129 }
130
131 fun openPicker(groupId: String) {
132     if (state.value.isBusy || state.value.isStorageSyncRunning) return

```

```

133         updateState(
134             state.value.copy(
135                 pickerGroupId = groupId,
136                 userMessage = null,
137             ),
138         )
139     }
140
141     fun closePicker() {
142         updateState(
143             state.value.copy(
144                 pickerGroupId = null,
145                 userMessage = null,
146             ),
147         )
148     }
149
150     fun toggleVaultExpanded(vaultUuid: UUID) {
151         val expanded = state.value.expandedVaultUuids.toMutableSet()
152         if (!expanded.add(vaultUuid)) {
153             expanded.remove(vaultUuid)
154         }
155         updateState(state.value.copy(expandedVaultUuids = expanded))
156     }
157
158     fun addStorageToCurrentGroup(storageUuid: UUID) {
159         val groupId = state.value.pickerGroupId ?: return
160         viewModelScope.launch {
161             withGroupMutationBusy {
162                 val storage = storageByUuid[storageUuid]
163                 ?: return@withGroupMutationBusy
164                 UserNotification.TextRes(R.string.sync_storage_not_in_vaults)
165                 val isEncrypted = storage.metaInfo.value.encInfo != null
166                 val result = groupsUseCase.addStorageToGroup(
167                     groupId = groupId,
168                     storageUuid = storageUuid,
169                     compatibility = StorageSyncCompatibilityInput(
170                         isEncrypted = isEncrypted,
171                         encryptionSecret = null,
172                     ),
173                 )
174                 when (result) {
175                     AddStorageToSyncGroupResult.Added -> UserNotification.TextRes(
176                         R.string.sync_msg_storage_added,
177                         listOf(groupId),
178                     )
179                 }
180             }
181         }
182     }
183 }

```

```

177         )
178         AddStorageToSyncGroupResult.AlreadyInGroup -> UserNotification.TextRes(
179             R.string.sync_msg_storage_already_added,
180         )
181         AddStorageToSyncGroupResult.EncryptedStorageNotAllowed ->
182         UserNotification.TextRes(
183             R.string.sync_msg_only_plain_storage_allowed,
184         )
185         AddStorageToSyncGroupResult.MissingEncryptionSecret ->
186         UserNotification.TextRes(
187             R.string.sync_msg_storage_encryption_key_required,
188         )
189         AddStorageToSyncGroupResult.IncompatibleEncryption ->
190         UserNotification.TextRes(
191             R.string.sync_msg_storage_incompatible_encryption,
192         )
193         AddStorageToSyncGroupResult.GroupNotFound -> UserNotification.TextRes(
194             R.string.sync_msg_group_removed,
195         )
196     }
197 }
198
199 fun removeStorageFromGroup(groupId: String, storageUuid: UUID) {
200     viewModelScope.launch {
201         withGroupMutationBusy {
202             groupsUseCase.removeStorageFromGroup(groupId, storageUuid)
203             UserNotification.TextRes(
204                 R.string.sync_msg_storage_removed,
205                 listOf(groupId),
206             )
207         }
208     }
209
210     fun runSyncNow() {
211         val started = runStorageSyncUseCase.enqueue(StorageSyncTriggerReason.SyncTab)
212         if (!started) {
213             updateState(
214                 state.value.copy(
215                     userMessage =
216                     UserNotification.TextRes(R.string.sync_msg_sync_already_running),
217                 ),
218             )
219         }
220     }

```

```

219     }
220
221     fun consumeUserMessage() {
222         updateState(state.value.copy(userMessage = null))
223     }
224
225     private fun observeVaults() {
226         viewModelScope.launch {
227             vaultsManager.vaults
228                 .flatMapLatest { vaults ->
229                     if (vaults.isEmpty()) {
230                         flowOf(emptyList())
231                     } else {
232                         combine(vaults.map { it.storages }) { rootStoragesByVault ->
233                             vaults.zip(rootStoragesByVault.toList())
234                         }.flatMapLatest { vaultWithRoots ->
235                             vaultsManager.unlockManager.openedStorages.flatMapLatest
236 { opened ->
237
238                                     val vaultNodes = vaultWithRoots.map { (vault, roots) ->
239                                         vault to roots.map { root ->
240                                             buildStorageTree(
241                                                 root = root,
242                                                 opened = opened,
243                                             )
244                                         }
245                                     }
246                                     val allStorages = vaultNodes
247                                         .flatMap { (_, trees) ->
248                                             trees.flatMap(::flattenStorages) }
249                                         .distinctBy { it.uuid }
250                                     storageByUuid = allStorages.associateBy { it.uuid }
251
252                                     if (allStorages.isEmpty()) {
253                                         flowOf(
254                                             vaultNodes.map { (vault, trees) ->
255                                                 StorageSyncVaultUi(
256                                                     uuid = vault.uuid,
257                                                     title = vaultTitle(vault as?
258 DescribedVault),
259                                                     type = vaultType(vault as? DescribedVault),
260                                                     storages = trees.map { tree ->
261                                                         toStorageUi(tree)
262                                                     },
263                                                 )
264                                         }
265                                     )
266                                     },

```



```

261         )
262     } else {
263         combine(
264             allStorages.map { storage ->
265                 { meta, avail ->
266                     storage.uuid to StorageSnapshot(meta, avail)
267                 }
268             },
269         ) { pairs ->
270             val snapByUuid = pairs.associate { it.first to
271                 it.second }
272             vaultNodes.map { (vault, trees) ->
273                 StorageSyncVaultUi(
274                     uuid = vault.uuid,
275                     title = vaultTitle(vault as?
276                         DescribedVault),
277                     type = vaultType(vault as? DescribedVault),
278                     storages = trees.map { tree ->
279                         toStorageUi(tree, snapByUuid)
280                     },
281                 )
282             }
283         }
284     }
285 }
286
287 .collect { mapped ->
288     updateState(
289         state.value.copy(
290             vaults = mapped,
291             groups = reloadGroupsUi(),
292         ),
293     )
294 }
295 }
296
297
298 private fun vaultType(vault: DescribedVault?): String {
299     return when (val descriptor = vault?.descriptor) {
300         is VaultDescriptor.LocalDevice -> uiStrings(R.string.vault_type_local_device)
301         is VaultDescriptor.LinkedRemote -> uiStrings(R.string.vault_type_remote,
302             descriptor.brand.name)
303         null -> uiStrings(R.string.vault_type_unknown)
304     }
305 }

```

```

303     }
304 }
305
306 private fun vaultTitle(vault: DescribedVault?): String {
307     return when (val descriptor = vault?.descriptor) {
308         is VaultDescriptor.LocalDevice -> uiStrings(R.string.vault_title_local)
309         is VaultDescriptor.LinkedRemote -> descriptor.accountDisplayName
310         null -> uiStrings(R.string.vault_title_unknown)
311     }
312 }
313
314 private fun buildStorageTree(
315     root: IStorage,
316     opened: Map<UUID, IStorage>,
317     visited: MutableSet<UUID> = mutableSetOf(),
318 ): StorageTreeNode {
319     if (!visited.add(root.uuid)) {
320         return StorageTreeNode(storage = root, children = emptyList())
321     }
322     val child = opened[root.uuid]
323     val children = if (child == null) {
324         emptyList()
325     } else {
326         listOf(
327             buildStorageTree(
328                 root = child,
329                 opened = opened,
330                 visited = visited,
331             ),
332         )
333     }
334     return StorageTreeNode(storage = root, children = children)
335 }
336
337 private fun flattenStorages(node: StorageTreeNode): List<IStorage> {
338     return listOf(node.storage) + node.children.flatMap(::flattenStorages)
339 }
340
341 private fun toStorageUi(
342     node: StorageTreeNode,
343     snapshotByUuid: Map<UUID, StorageSnapshot> = emptyMap(),
344 ): StorageSyncStorageUi {
345     val snap = snapshotByUuid[node.storage.uuid]
346     val meta = snap?.meta ?: node.storage.metaInfo.value
347     val isReachable = snap?.isAvailable ?: node.storage.isAvailable.value

```

```

348         val encryptionKind = when {
349             meta.encInfo == null -> StorageSyncEncryptionKind.NotEncrypted
350             node.storage.isVirtualStorage -> StorageSyncEncryptionKind.EncryptedOpened
351             else -> StorageSyncEncryptionKind.EncryptedClosed
352         }
353         return StorageSyncStorageUi(
354             uuid = node.storage.uuid,
355             name = meta.name ?: uiStrings(R.string.no_name),
356             encryptionKind = encryptionKind,
357             isReachable = isReachable,
358             children = node.children.map { child ->
359                 toStorageUi(
360                     node = child,
361                     snapshotByUuid = snapshotByUuid,
362                 )
363             },
364         )
365     }
366
367     private data class StorageSnapshot(
368         val meta: IStorageMetaInfo,
369         val isAvailable: Boolean,
370     )
371
372     private data class StorageTreeNode(
373         val storage: IStorage,
374         val children: List<StorageTreeNode>,
375     )
376
377     private suspend fun reloadGroupsUi(): List<StorageSyncGroupUi> =
378         groupsUseCase.getGroups().map { group ->
379             val incompatible = group.storageUuids.filterTo(mutableSetOf()) { uuid ->
380                 val storage = storageByUuid[uuid] ?: return@filterTo true
381                 !isStorageCompatibleWithGroup(
382                     storage = storage,
383                     group = group,
384                 )
385             }
386             StorageSyncGroupUi(
387                 id = group.id,
388                 storageUuids = group.storageUuids,
389                 encryptionKind = group.encryptionKind,
390                 incompatibleStorageUuids = incompatible,
391             )
392         }

```

```

393
394     private suspend fun withGroupMutationBusy(
395         clearPicker: Boolean = false,
396         block: suspend () -> UserNotification?,
397     ) {
398         if (state.value.isStorageSyncRunning) {
399             updateState(
400                 state.value.copy(
401                     userMessage =
402                     UserNotification.TextRes(R.string.sync_msg_blocked_during_sync),
403                 ),
404             )
405             return
406         }
407         updateState(state.value.copy(isBusy = true, userMessage = null))
408         val message = block()
409         val groups = reloadGroupsUi()
410         updateState(
411             state.value.copy(
412                 groups = groups,
413                 pickerGroupId = if (clearPicker) null else state.value.pickerGroupId,
414                 isBusy = false,
415                 userMessage = message,
416             ),
417         )
418     }
419 }

```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/theme/
Color.kt

```
1  package com.github.nullptroma.wallenc.ui.theme
2
3  import androidx.compose.ui.graphics.Color
4
5  val Purple80 = Color(0xFFD0BCFF)
6  val PurpleGrey80 = Color(0xFFCCC2DC)
7  val Pink80 = Color(0xFFE8B8C8)
8
9  val Purple40 = Color(0xFF6650a4)
10 val PurpleGrey40 = Color(0xFF625b71)
11 val Pink40 = Color(0xFF7D5260)
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/theme/ Theme.kt

```
1 package com.github.nullptroma.wallenc.ui.theme
2
3 import android.os.Build
4 import androidx.compose.foundation.isSystemInDarkTheme
5 import androidx.compose.material3.MaterialTheme
6 import androidx.compose.material3.darkColorScheme
7 import androidx.compose.material3.dynamicDarkColorScheme
8 import androidx.compose.material3.dynamicLightColorScheme
9 import androidx.compose.material3.lightColorScheme
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.platform.LocalContext
12
13 private val DarkColorScheme = darkColorScheme(
14     primary = Purple80,
15     secondary = PurpleGrey80,
16     tertiary = Pink80
17 )
18
19 private val LightColorScheme = lightColorScheme(
20     primary = Purple40,
21     secondary = PurpleGrey40,
22     tertiary = Pink40
23
24     /* Other default colors to override
25     background = Color(0xFFFFFBFE),
26     surface = Color(0xFFFFFBFE),
27     onPrimary = Color.White,
28     onSecondary = Color.White,
29     onTertiary = Color.White,
30     onBackground = Color(0xFF1C1B1F),
31     onSurface = Color(0xFF1C1B1F),
32     */
33 )
34
35 @Composable
36 fun WallencTheme(
37     darkTheme: Boolean = isSystemInDarkTheme(),
38     // Dynamic color is available on Android 12+
39     dynamicColor: Boolean = true,
40     content: @Composable () -> Unit
41 ) {
42     val colorScheme = when {
```

```
43         dynamicColor && Build.VERSION.SDK_INT >= Build.VERSION_CODES.S -> {
44             val context = LocalContext.current
45             if (darkTheme) dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
46         }
47
48         darkTheme -> DarkColorScheme
49         else -> LightColorScheme
50     }
51
52     MaterialTheme(
53         colorScheme = colorScheme,
54         typography = Typography,
55         content = content
56     )
57 }
```

Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/theme/Type.kt

```
1 package com.github.nullptroma.wallenc.ui.theme
2
3 import androidx.compose.material3.Typography
4 import androidx.compose.ui.text.TextStyle
5 import androidx.compose.ui.text.font.FontFamily
6 import androidx.compose.ui.text.font.FontWeight
7 import androidx.compose.ui.unit.sp
8
9 // Set of Material typography styles to start with
10 val Typography = Typography(
11     bodyLarge = TextStyle(
12         fontFamily = FontFamily.Default,
13         fontWeight = FontWeight.Normal,
14         fontSize = 16.sp,
15         lineHeight = 24.sp,
16         letterSpacing = 0.5.sp
17     )
18     /* Other default text styles to override
19     titleLarge = TextStyle(
20         fontFamily = FontFamily.Default,
21         fontWeight = FontWeight.Normal,
22         fontSize = 22.sp,
23         lineHeight = 28.sp,
24         letterSpacing = 0.sp
25     ),
26     labelSmall = TextStyle(
27         fontFamily = FontFamily.Default,
28         fontWeight = FontWeight.Medium,
29         fontSize = 11.sp,
30         lineHeight = 16.sp,
31         letterSpacing = 0.5.sp
32     )
33     */
34 )
```


Исходный файл ui/src/main/java/com/github/nullptroma/wallenc/ui/Utils/
DebouncedLambda.kt

```
1 package com.github.nullptroma.wallenc.ui.Utils
2
3
4 fun debouncedLambda(debounceMs: Long = 300, action: ()->Unit) : ()->Unit {
5     var latest: Long = 0
6     return {
7         val now = System.currentTimeMillis()
8         if (now - latest >= debounceMs) {
9             latest = now
10            action()
11        }
12    }
13 }
```

Исходный файл ui/src/main/res/values-ru/plurals.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <plurals name="sync_progress_preparing">
4          <item quantity="one">Синхронизация: подготовка %d группы</item>
5          <item quantity="few">Синхронизация: подготовка %d групп</item>
6          <item quantity="many">Синхронизация: подготовка %d групп</item>
7          <item quantity="other">Синхронизация: подготовка %d групп</item>
8      </plurals>
9      <plurals name="sync_progress_group_processing">
10         <item quantity="one">Синхронизация: группа «%1$s» – обработка %2$d записи</item>
11         <item quantity="few">Синхронизация: группа «%1$s» – обработка %2$d записей</item>
12         <item quantity="many">Синхронизация: группа «%1$s» – обработка %2$d записей</item>
13         <item quantity="other">Синхронизация: группа «%1$s» – обработка %2$d записей</item>
14     </plurals>
15     <plurals name="sync_progress_group_entries_failed">
16         <item quantity="one">Синхронизация: группа «%1$s» – не применена %2$d запись</item>
17         <item quantity="few">Синхронизация: группа «%1$s» – не применены %2$d записи</item>
18         <item quantity="many">Синхронизация: группа «%1$s» – не применено %2$d записей</
19     item>
20         <item quantity="other">Синхронизация: группа «%1$s» – не применено %2$d записей</
21     item>
22     </plurals>
23 </resources>
```

Исходный файл ui/src/main/res/values-ru/strings.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <string name="nav_label_local_vault">Локальное хранилище</string>
4      <string name="nav_cd_local_vault">Локальное хранилище</string>
5      <string name="nav_label_remote_vaults">Удалённые хранилища</string>
6      <string name="nav_cd_remote_vaults">Удалённые хранилища</string>
7      <string name="nav_label_main">Главная</string>
8      <string name="nav_label_sync">Синхронизация</string>
9      <string name="nav_label_settings">Настройки</string>
10     <string name="nav_cd_back">Назад</string>
11     <string name="screen_title_remote_vault">Удалённое хранилище</string>
12     <string name="screen_title_local_vault">Локальное хранилище</string>
13     <string name="screen_title_yandex_vault">Хранилище Яндекс.Диска</string>
14     <string name="screen_title_storage">Хранилище</string>
15     <string name="screen_title_two_fa">Токены 2FA</string>
16     <string name="screen_title_text_secrets">Текстовые секреты</string>
17     <string name="screen_title_text_edit">Текст</string>
18     <string name="main_work_status_label">Статус:</string>
19     <string name="main_status_multiple_tasks">Выполняется задач: %1$d</string>
20     <string name="main_status_vault_scanning_storages">Сканирование vault: загрузка списка
хранилищ</string>
21     <string name="settings_title">Настройки</string>
22     <string name="sync_groups_title">Группы синхронизации</string>
23     <string name="sync_progress_section_title">Синхронизация хранилищ</string>
24     <string name="sync_groups_busy_section_title">Сохранение групп синхронизации</string>
25     <string name="sync_run_now">Запустить синхронизацию</string>
26     <string name="sync_add_storage">Добавить хранилище в группу</string>
27     <string name="sync_remove_group">Удалить группу</string>
28     <string name="sync_group_empty">В группе нет хранилищ</string>
29     <string name="sync_remove_storage">Убрать хранилище из группы</string>
30     <string name="sync_cd_picker_back">Закрыть выбор хранилища</string>
31     <string name="sync_picker_title">Выбор хранилища для %1$s</string>
32     <string name="sync_picker_add">Добавить</string>
33     <string name="sync_picker_added">Добавлено</string>
34     <string name="sync_picker_cd_add">Добавить хранилище в группу</string>
35     <string name="sync_picker_no_storages">В этом хранилище нет доступных каталогов</string>
36     <string name="sync_picker_expand">Развернуть</string>
37     <string name="sync_picker_collapse">Свернуть</string>
38     <string name="sync_fab_create_group_cd">Создать группу синхронизации</string>
39     <string name="sync_group_incompatible_warning">Несовместимые хранилища в группе: %1$d</
string>
40     <string name="sync_group_policy_line">Политика шифрования группы: %1$s</string>
41     <string name="sync_group_policy_unset">Не определена (группа пуста)</string>
42     <string name="sync_group_policy_plain">Только незашифрованные</string>
```

```

43     <string name="sync_group_policy_password">Только зашифрованные с паролем группы</string>
44     <string name="sync_remove_group_confirm_title">Удалить группу?</string>
45     <string name="sync_remove_group_confirm_message">Удалить группу синхронизации «%1$s»?</
string>
46     <string name="sync_remove_storage_confirm_title">Убрать хранилище?</string>
47     <string name="sync_remove_storage_confirm_message">Убрать хранилище «%1$s» из группы?</
string>
48     <string name="sync_confirm_delete">Удалить</string>
49     <string name="sync_cancel">Отмена</string>
50     <string name="sync_msg_group_created">Создана группа %1$s</string>
51     <string name="sync_msg_group_removed">Группа удалена</string>
52     <string name="sync_msg_storage_added">Хранилище добавлено в %1$s</string>
53     <string name="sync_msg_storage_removed">Хранилище убрано из %1$s</string>
54     <string name="sync_msg_storage_already_added">Хранилище уже добавлено в группу</string>
55     <string name="sync_msg_only_plain_storage_allowed">В группы синхронизации можно
добавлять только незашифрованные хранилища</string>
56     <string name="sync_msg_storage_encryption_key_required">Для зашифрованного хранилища
нужно знать пароль (откройте его перед добавлением)</string>
57     <string name="sync_msg_storage_incompatible_encryption">Хранилище не совместимо с
политикой шифрования группы</string>
58     <string name="sync_msg_sync_already_running">Синхронизация уже выполняется</string>
59     <string name="sync_msg_blocked_during_sync">Дождитесь окончания синхронизации</string>
60     <string name="sync_encryption_unknown">Неизвестно</string>
61     <string name="sync_storage_encryption_line">Шифрование: %1$s</string>
62     <string name="sync_storage_missing_title">Не найдено в текущих vault</string>
63     <string name="sync_storage_pending_vault_scan">Ожидание: список хранилищ в vault ещё
загружается</string>
64     <string name="sync_storage_not_in_vaults">Нет в дереве хранилищ (удалено, другой аккаунт
или не прошёл init)</string>
65     <string name="sync_storage_unreachable">Хранилище недоступно (vault или сеть)</string>
66     <string name="no_name">&lt; без имени&gt;</string>
67     <string name="show_storage_item_menu">Меню хранилища</string>
68     <string name="storage_row_task_running_cd">Выполняется операция с этим хранилищем</
string>
69     <string name="storage_menu_busy">%1$s (задача выполняется)</string>
70     <string name="rename">Переименовать</string>
71     <string name="remove">Удалить</string>
72     <string name="encrypt">Шифрование</string>
73     <string name="new_name_title">Новое имя</string>
74     <string name="remove_confirmation_dialog">Удалить хранилище «%1$s»?</string>
75     <string name="storage_lock_actions">Действия с шифрованием</string>
76     <string name="storage_sync_lock_checking">Проверка блокировки...</string>
77     <string name="storage_sync_unlock_action">Снять блокировку синхронизации</string>
78     <string name="storage_sync_not_locked">Синхронизация не заблокирована</string>
79     <string name="storage_field_available">Доступно: %1$s</string>
80     <string name="storage_value_yes">да</string>
81     <string name="storage_value_no">нет</string>

```

```

82     <string name="storage_field_files">Файлов: %1$s</string>
83     <string name="storage_field_size">Размер: %1$s</string>
84     <string name="storage_field_virtual">Виртуальное: %1$s</string>
85     <string name="storage_unavailable_hint">Хранилище недоступно</string>
86     <string name="storage_meta_unavailable_hint">Метаданные недоступны – переименование,
шифрование и открытие отключены</string>
87     <string name="storage_status_meta_unavailable">Метаданные недоступны</string>
88     <string name="storage_home_meta_unavailable">Не удалось загрузить метаданные хранилища.
2FA и текстовые секреты недоступны.</string>
89     <string name="storage_menu_unavailable">Недоступно: %1$s</string>
90     <string name="storage_status_not_encrypted">Не зашифровано</string>
91     <string name="storage_status_encrypted_open">Зашифровано (открыто)</string>
92     <string name="storage_status_encrypted_closed">Зашифровано (закрыто)</string>
93     <string name="vault_fab_add_storage_cd">Создать хранилище</string>
94     <string name="vault_fab_add_storage_disabled_cd">Создание недоступно: хранилище
недоступно</string>
95     <string name="vault_fab_add_storage_busy_cd">Создание хранилища уже выполняется</string>
96     <string name="vault_msg_storage_pipeline_busy">С этим хранилищем уже выполняется
операция</string>
97     <string name="vault_msg_vault_list_mutation_busy">Список хранилищ сейчас меняется –
подождите</string>
98     <string name="vault_msg_rescan_already_in_progress">Сканирование хранилищ уже
выполняется</string>
99     <string name="vault_unavailable_banner">Хранилище недоступно. Проверьте сеть, путь или
разблокировку.</string>
100    <string name="vault_loading_storages">Загрузка списка хранилищ...</string>
101    <string name="vault_empty_list_hint">В этом хранилище пока нет каталогов. Создайте
хранилище кнопкой «+», когда оно доступно.</string>
102    <string name="vault_empty_list_hint_remote">На удалённом хранилище каталоги не найдены.
Если папки уже есть на сервере, нажмите «Обновить список», либо создайте хранилище кнопкой
«+», когда оно доступно.</string>
103    <string name="vault_rescan_storages_action">Обновить список</string>
104    <string name="task_pipeline_title">Очередь задач</string>
105    <string name="task_pipeline_jobs">Задачи</string>
106    <string name="task_pipeline_log">Журнал</string>
107    <string name="task_pipeline_cancel_all">Отменить все</string>
108    <string name="task_pipeline_open">Открыть очередь задач</string>
109    <string name="task_pipeline_run_test">Тестовая задача</string>
110    <string name="task_pipeline_test_dialog_title">Параметры тестовой задачи</string>
111    <string name="task_pipeline_test_dialog_duration">Длительность: %1$d с</string>
112    <string name="task_pipeline_test_dialog_start">Запустить</string>
113    <string name="task_pipeline_test_dialog_cancel">Отмена</string>
114    <string name="task_pipeline_test_dialog_infinity">Бесконечно (неопределённый прогресс)</
string>
115    <string name="task_pipeline_test_running">Тестовая задача (%1$d с)</string>
116    <string name="task_pipeline_test_running_infinity">Тестовая задача (%1$d с, ∞)</string>
117    <string name="task_state_queued">В очереди</string>
118    <string name="task_state_running">Выполняется</string>
119    <string name="task_state_completed">Завершено</string>

```

```

120     <string name="task_state_cancelled">Отменено</string>
121     <string name="task_state_failed">Ошибка: %1$s</string>
122     <string name="task_title_dump_storage_log">Выгрузка дерева в журнал</string>
123     <string name="task_title_create_storage">Создание хранилища</string>
124     <string name="task_title_enable_encryption">Включение шифрования</string>
125     <string name="task_title_open_encrypted_storage">Расшифровка и открытие хранилища</
string>
126     <string name="task_progress_decrypt_running">Расшифровка</string>
127     <string name="task_progress_dump_storage_log">Сканирование дерева</string>
128     <string name="task_progress_create_storage">Создание хранилища</string>
129     <string name="task_progress_enable_encryption">Шифрование</string>
130     <string name="task_progress_close_storage">Заккрытие хранилища</string>
131     <string name="task_progress_disable_encryption">Очистка содержимого</string>
132     <string name="task_progress_rename_storage">Переименование</string>
133     <string name="task_progress_remove_storage">Удаление</string>
134     <string name="task_progress_clear_sync_lock">Снятие блокировки</string>
135     <string name="task_progress_add_remote_vault">Добавление</string>
136     <string name="task_progress_remove_remote_vault">Удаление</string>
137     <string name="task_progress_retry_remote_vault">Подключение</string>
138     <string name="task_progress_rescan_vault_storages">Сканирование хранилищ</string>
139     <string name="task_progress_save_2fa_token">Сохранение</string>
140     <string name="task_progress_delete_2fa_token">Удаление</string>
141     <string name="task_progress_save_text_secret">Сохранение</string>
142     <string name="task_progress_delete_text_secret">Удаление</string>
143     <string name="task_title_close_encrypted_storage">Заккрытие зашифрованного хранилища</
string>
144     <string name="task_title_disable_encryption">Отключение шифрования</string>
145     <string name="task_title_rename_storage">Переименование хранилища</string>
146     <string name="task_title_remove_storage">Удаление хранилища</string>
147     <string name="task_title_clear_sync_lock">Снятие блокировки синхронизации</string>
148     <string name="task_title_add_remote_vault">Добавление удалённого хранилища</string>
149     <string name="task_title_remove_remote_vault">Удаление удалённого хранилища</string>
150     <string name="task_title_retry_remote_vault">Повторное подключение удалённого
хранилища</string>
151     <string name="task_title_rescan_vault_storages">Обновление списка хранилищ</string>
152     <string name="task_title_storage_sync">Синхронизация хранилищ (%1$s)</string>
153     <string name="task_title_save_2fa_token">Сохранение 2FA токена</string>
154     <string name="task_title_delete_2fa_token">Удаление 2FA токена</string>
155     <string name="task_title_save_text_secret">Сохранение текстового секрета</string>
156     <string name="task_title_delete_text_secret">Удаление текстового секрета</string>
157     <string name="error_storage_not_found">Хранилище не найдено</string>
158     <string name="error_storage_locked_view">Откройте расшифрованное отображение storage для
работы с этим разделом</string>
159     <string name="error_secret_not_found">Секрет не найден</string>
160     <string name="error_storage_not_writable">Хранилище недоступно для записи</string>
161     <string name="error_file_not_found">Файл не найден</string>

```



```

162     <string name="error_incorrect_password">Неверный пароль</string>
163     <string name="error_storage_not_encrypted">Хранилище не зашифровано</string>
164     <string name="error_enc_info_missing">Отсутствуют метаданные шифрования</string>
165     <string name="error_delete_root_forbidden">Нельзя удалить корень хранилища</string>
166     <string name="error_not_a_file">Ожидался файл</string>
167     <string name="error_not_a_directory">Ожидалась папка</string>
168     <string name="error_path_is_file">Путь указывает на файл, а не на папку</string>
169     <string name="error_cannot_write_over_directory">Нельзя записать поверх папки</string>
170     <string name="error_unexpected_state">Хранилище в неожиданном состоянии</string>
171     <string name="error_network">Ошибка сети или сервера</string>
172     <string name="error_disk_resource_locked">Ресурс временно заблокирован. Повторите
позже.</string>
173     <string name="error_unknown">Что-то пошло не так</string>
174     <string name="sync_error_group_not_found">Группа синхронизации не найдена</string>
175     <string name="vault_link_error_auth">Не удалось войти</string>
176     <string name="vault_link_error_not_registered">Вход не готов. Перезапустите
приложение.</string>
177     <string name="vault_link_error_unknown">Не удалось войти</string>
178     <string name="vault_link_error_unsupported_brand">Этот провайдер не поддерживается</
string>
179     <string name="msg_encryption_enabled">Шифрование включено</string>
180     <string name="msg_encryption_enabled_open_failed">Шифрование включено; откройте
хранилище вручную для просмотра</string>
181     <string name="msg_storage_already_encrypted">Хранилище уже зашифровано</string>
182     <string name="msg_storage_not_empty">Хранилище не пустое</string>
183     <string name="msg_storage_empty_state_unknown">Не удалось определить, пусто ли
хранилище</string>
184     <string name="msg_unsupported_storage_type">Неподдерживаемый тип хранилища</string>
185     <string name="msg_encryption_disabled">Шифрование отключено</string>
186     <string name="msg_invalid_storage_for_sync_lock">Некорректное хранилище</string>
187     <string name="msg_sync_lock_cleared">Блокировка синхронизации снята</string>
188     <string name="remote_vaults_add_cd">Добавить удалённое хранилище</string>
189     <string name="remote_vaults_empty_hint">Пока нет удалённых хранилищ. Нажмите «+», чтобы
добавить Yandex.</string>
190     <string name="remote_vaults_add_title">Добавить хранилище</string>
191     <string name="remote_vaults_add_pick_provider">Выберите провайдера:</string>
192     <string name="remote_vaults_provider_yandex">Яндекс</string>
193     <string name="remote_vaults_add_cancel">Отмена</string>
194     <string name="remote_vault_type_yandex">Яндекс</string>
195     <string name="remote_vault_unavailable">Хранилище временно недоступно</string>
196     <string name="remote_vault_retry_action">Повторить подключение</string>
197     <string name="remote_vault_retrying">Пробую подключиться...</string>
198     <string name="remote_vault_delete_cd">Удалить удалённое хранилище</string>
199     <string name="remote_vault_remove_title">Удалить удалённое хранилище?</string>
200     <string name="remote_vault_remove_message">Удалить «%1$s» с этого устройства? Данные на
сервере не удаляются.</string>
201     <string name="dialog_cancel">Отмена</string>

```

```

202     <string name="dialog_ok">OK</string>
203     <string name="dialog_encryption_enable_title">Включить шифрование</string>
204     <string name="dialog_password_label">Пароль</string>
205     <string name="dialog_encrypt_paths">Шифровать пути</string>
206     <string name="dialog_apply">Применить</string>
207     <string name="dialog_open_encrypted_title">Открыть зашифрованное хранилище</string>
208     <string name="dialog_remember_password">Запомнить пароль</string>
209     <string name="dialog_open">Открыть</string>
210     <string name="dialog_close">Закрыть</string>
211     <string name="dialog_disable_encryption">Отключить шифрование</string>
212     <string name="dialog_done">Готово</string>
213     <string name="vault_type_local_device">Локальное устройство</string>
214     <string name="vault_type_remote">Удалённое: %1$s</string>
215     <string name="vault_type_unknown">Неизвестный тип</string>
216     <string name="vault_title_local">Локальное хранилище</string>
217     <string name="vault_title_unknown">Неизвестное хранилище</string>
218     <string name="enc_status_not_encrypted">Не зашифровано</string>
219     <string name="enc_status_encrypted_open">Зашифровано (открыто)</string>
220     <string name="enc_status_encrypted">Зашифровано</string>
221     <string name="text_edit_screen_placeholder">Содержимое: %1$s</string>
222     <string name="storage_home_unnamed_storage">Storage</string>
223     <string name="storage_home_status_line">Статус: %1$s, %2$s</string>
224     <string name="storage_home_status_available">доступно</string>
225     <string name="storage_home_status_unavailable">недоступно</string>
226     <string name="storage_home_status_encrypted">зашифровано</string>
227     <string name="storage_home_status_not_encrypted">не зашифровано</string>
228     <string name="storage_home_two_fa_title">2FA токены (%1$d)</string>
229     <string name="storage_home_two_fa_subtitle">Коды и секреты двухфакторной
аутентификации</string>
230     <string name="storage_home_text_secrets_title">Текстовые секреты (%1$d)</string>
231     <string name="storage_home_text_secrets_subtitle">Заметки, токены и произвольные пары
ключ-значение</string>
232     <string name="storage_home_future_sections">Скоро здесь появятся Files, Media и другие
типы данных.</string>
233     <string name="two_fa_add_token">Добавить токен</string>
234     <string name="two_fa_empty_state">Пока нет 2FA токенов</string>
235     <string name="two_fa_create_title">Новый 2FA токен</string>
236     <string name="two_fa_edit_title">Редактирование 2FA токена</string>
237     <string name="two_fa_field_issuer">Сервис</string>
238     <string name="two_fa_field_account">Аккаунт</string>
239     <string name="two_fa_field_secret">Секрет</string>
240     <string name="two_fa_field_notes_optional">Заметка (опционально)</string>
241     <string name="two_fa_field_algorithm">Алгоритм (SHA1, SHA256, SHA512)</string>
242     <string name="two_fa_field_digits_value">Количество цифр: %1$d</string>
243     <string name="two_fa_field_period_seconds_value">Период обновления: %1$d с</string>

```



```

244     <string name="two_fa_code_unavailable">-----</string>
245     <string name="two_fa_code_refresh_label">Обновление через</string>
246     <string name="two_fa_code_refresh_seconds">%1$d с</string>
247     <string name="two_fa_code_invalid_secret">Неверный секрет или формат</string>
248     <string name="two_fa_scan_qr_action">Сканировать QR</string>
249     <string name="two_fa_scan_qr_title">Сканирование QR-кода TOTP</string>
250     <string name="two_fa_scan_qr_invalid">QR-код не содержит валидный otpauth://totp URI</
string>
251     <string name="two_fa_camera_permission_required">Нужно разрешение на камеру для
сканирования QR</string>
252     <string name="text_secret_create">Создать секрет</string>
253     <string name="text_secret_edit">Редактировать секрет</string>
254     <string name="text_secret_title">Название</string>
255     <string name="text_secret_empty_state">Пока нет текстовых секретов</string>
256     <string name="text_secret_items_count">Элементов: %1$d</string>
257     <string name="text_secret_item_without_label">Без названия</string>
258     <string name="text_secret_more_fields">ещё полей: %1$d</string>
259     <string name="text_secret_item_label_optional">Название (опционально)</string>
260     <string name="text_secret_item_value">Значение</string>
261     <string name="text_secret_add_item">Добавить пару</string>
262     <string name="text_secret_copy_value">Скопировать значение</string>
263     <string name="save">Сохранить</string>
264     <string name="cancel">Отмена</string>
265     <string name="edit">Редактировать</string>
266     <string name="settings_language_section">Язык</string>
267     <string name="settings_language_system">Как в системе</string>
268     <string name="settings_language_english">English</string>
269     <string name="settings_language_russian">Русский</string>
270     <string name="task_pipeline_test_elapsed">Прошло: %1$d с / %2$d с</string>
271     <string name="text_secret_clipboard_fallback_label">значение</string>
272     <string name="sync_progress_no_groups">Синхронизация: группы не настроены</string>
273     <string name="sync_progress_started">Синхронизация: запущена</string>
274     <string name="sync_progress_completed">Синхронизация: завершена</string>
275     <string name="sync_progress_group_preparing">Синхронизация: группа «%1$s» – подготовка</
string>
276     <string name="sync_progress_group_not_found">Синхронизация: группа «%1$s» не найдена</
string>
277     <string name="sync_progress_group_skipped_few_storages">Синхронизация: группа «%1$s»
пропущена (нужно минимум 2 хранилища)</string>
278     <string name="sync_progress_group_skipped_incompatible">Синхронизация: группа «%1$s»
пропущена (несовместимое шифрование: %2$d)</string>
279     <string name="sync_progress_group_acquiring_locks">Синхронизация: группа «%1$s» –
получение блокировок</string>
280     <string name="sync_progress_group_lock">Синхронизация: группа «%1$s» – блокировка %2$d/
%3$d</string>
281     <string name="sync_progress_group_lock_failed">Синхронизация: группа «%1$s» – блокировка
не получена, пропуск</string>

```

```

282     <string name="sync_progress_group_reading_journals">Синхронизация: группа «%1$s» –
        чтение журналов</string>
283     <string name="sync_progress_group_cancelled">Синхронизация: группа «%1$s» отменена новым
        запуском</string>
284     <string name="sync_progress_group_journal">Синхронизация: группа «%1$s» – журнал %2$d/
        %3$d</string>
285     <string name="sync_progress_group_no_entries">Синхронизация: группа «%1$s» – нет записей
        в журнале</string>
286     <string name="sync_progress_group_entry">Синхронизация: группа «%1$s» – запись %2$d/
        %3$d</string>
287     <string name="sync_progress_group_completed">Синхронизация: группа «%1$s» завершена</
        string>
288     <string name="sync_progress_group_renewing_locks">Синхронизация: группа «%1$s» –
        продление блокировок</string>
289     <string name="sync_progress_group_lock_renewal_failed">Синхронизация: группа «%1$s» – не
        удалось продлить блокировку</string>
290     <string name="task_progress_clear_content">%1$d / %2$d</string>
291     <string name="task_log_sync_started">Синхронизация хранилищ запущена (%1$s)</string>
292     <string name="task_log_sync_finished">Синхронизация хранилищ завершена (%1$s)</string>
293     <string name="task_log_sync_failed">Синхронизация не удалась (%1$s): %2$s</string>
294     <string name="task_sync_trigger_debounce">debounce</string>
295     <string name="task_sync_trigger_sync_tab">sync-tab</string>
296     <string name="task_sync_trigger_background">background</string>
297     <string name="task_log_enumerating">Перечисление файлов и папок...</string>
298     <string name="task_log_creating_storage">Создание хранилища...</string>
299     <string name="task_log_storage_created">Хранилище создано</string>
300     <string name="task_log_checking_storage">Проверка хранилища...</string>
301     <string name="task_log_encrypting">Шифрование...</string>
302     <string name="task_log_encryption_enabled">Шифрование включено</string>
303     <string name="task_log_already_encrypted">Хранилище уже зашифровано</string>
304     <string name="task_log_not_empty">Хранилище не пустое</string>
305     <string name="task_log_empty_unknown">Не удалось определить, пусто ли хранилище</string>
306     <string name="task_log_unsupported_type">Неподдерживаемый тип хранилища</string>
307     <string name="task_log_enable_encryption_failed">Не удалось включить шифрование</string>
308     <string name="task_log_opening_storage">Открытие зашифрованного хранилища...</string>
309     <string name="task_log_storage_opened">Хранилище открыто</string>
310     <string name="task_log_open_storage_failed">Не удалось открыть хранилище</string>
311     <string name="task_log_closing_storage">Закрытие хранилища...</string>
312     <string name="task_log_storage_closed">Хранилище закрыто</string>
313     <string name="task_log_close_storage_failed">Не удалось закрыть хранилище</string>
314     <string name="task_log_disabling_encryption">Отключение шифрования...</string>
315     <string name="task_log_encryption_disabled">Шифрование отключено</string>
316     <string name="task_log_disable_encryption_failed">Не удалось отключить шифрование</
        string>
317     <string name="task_log_renaming">Переименование...</string>
318     <string name="task_log_renamed">Переименовано</string>
319     <string name="task_log_rename_failed">Не удалось переименовать</string>
320     <string name="task_log_removing_storage">Удаление хранилища...</string>

```

```

321     <string name="task_log_removed">Удалено</string>
322     <string name="task_log_remove_failed">Не удалось удалить</string>
323     <string name="task_log_invalid_storage">Некорректное хранилище</string>
324     <string name="task_log_clearing_sync_lock">Снятие блокировки синхронизации...</string>
325     <string name="task_log_sync_lock_cleared">Блокировка синхронизации снята</string>
326     <string name="task_log_clear_sync_lock_failed">Не удалось снять блокировку</string>
327     <string name="task_log_adding_vault">Добавление хранилища...</string>
328     <string name="task_log_vault_added">Хранилище добавлено</string>
329     <string name="task_log_add_vault_failed">Не удалось добавить хранилище</string>
330     <string name="task_log_removing_remote_vault">Удаление удалённого хранилища...</string>
331     <string name="task_log_remote_vault_removed">Удалённое хранилище удалено</string>
332     <string name="task_log_remove_vault_failed">Не удалось удалить хранилище</string>
333     <string name="task_log_retrying_vault">Повторное подключение...</string>
334     <string name="task_log_retry_requested">Повтор запрошен</string>
335     <string name="task_log_retry_vault_failed">Не удалось повторить подключение</string>
336     <string name="task_log_rescanning_vault_storages">Повторное сканирование хранилищ на
удалённом vault...</string>
337     <string name="task_log_rescan_vault_storages_done">Список хранилищ обновлён</string>
338     <string name="task_log_rescan_vault_storages_failed">Не удалось обновить список
хранилищ</string>
339     <string name="task_log_test_started">Тестовая задача запущена на %1$d с</string>
340     <string name="task_log_test_finished">Тестовая задача завершена</string>
341 </resources>
342

```

Исходный файл ui/src/main/res/values/plurals.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <plurals name="sync_progress_preparing">
4          <item quantity="one">Storage sync: preparing %d group</item>
5          <item quantity="other">Storage sync: preparing %d groups</item>
6      </plurals>
7      <plurals name="sync_progress_group_processing">
8          <item quantity="one">Storage sync: group "%1$s" processing %2$d entry</item>
9          <item quantity="other">Storage sync: group "%1$s" processing %2$d entries</item>
10     </plurals>
11     <plurals name="sync_progress_group_entries_failed">
12         <item quantity="one">Storage sync: group "%1$s" – %2$d entry failed</item>
13         <item quantity="other">Storage sync: group "%1$s" – %2$d entries failed</item>
14     </plurals>
15 </resources>
16
```

Исходный файл ui/src/main/res/values/strings.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <string name="nav_label_local_vault">Local vault</string>
4      <string name="nav_cd_local_vault">Local vault</string>
5      <string name="nav_label_remote_vaults">Remote vaults</string>
6      <string name="nav_cd_remote_vaults">Remote vaults</string>
7      <string name="nav_label_main">Home</string>
8      <string name="nav_label_sync">Sync</string>
9      <string name="nav_label_settings">Settings</string>
10     <string name="nav_cd_back">Go back</string>
11     <string name="screen_title_remote_vault">Remote vault</string>
12     <string name="screen_title_local_vault">Local vault</string>
13     <string name="screen_title_yandex_vault">Yandex Disk vault</string>
14     <string name="screen_title_storage">Storage</string>
15     <string name="screen_title_two_fa">2FA tokens</string>
16     <string name="screen_title_text_secrets">Text secrets</string>
17     <string name="screen_title_text_edit">Text</string>
18     <string name="main_work_status_label">Status:</string>
19     <string name="main_status_multiple_tasks">Running tasks: %1$d</string>
20     <string name="main_status_vault_scanning_storages">Scanning vault: loading storage
list</string>
21     <string name="settings_title">Settings</string>
22     <string name="sync_groups_title">Sync groups</string>
23     <string name="sync_progress_section_title">Storage sync</string>
24     <string name="sync_groups_busy_section_title">Saving sync groups</string>
25     <string name="sync_run_now">Run sync now</string>
26     <string name="sync_add_storage">Add storage to group</string>
27     <string name="sync_remove_group">Remove group</string>
28     <string name="sync_group_empty">No storages in this group</string>
29     <string name="sync_remove_storage">Remove storage from group</string>
30     <string name="sync_cd_picker_back">Close storage picker</string>
31     <string name="sync_picker_title">Pick storage for %1$s</string>
32     <string name="sync_picker_add">Add</string>
33     <string name="sync_picker_added">Added</string>
34     <string name="sync_picker_cd_add">Add storage to group</string>
35     <string name="sync_picker_no_storages">No available folders in this vault</string>
36     <string name="sync_picker_expand">Expand</string>
37     <string name="sync_picker_collapse">Collapse</string>
38     <string name="sync_fab_create_group_cd">Create sync group</string>
39     <string name="sync_group_incompatible_warning">Incompatible storages in group: %1$d</
string>
40     <string name="sync_group_policy_line">Group encryption policy: %1$s</string>
41     <string name="sync_group_policy_unset">Not set (group is empty)</string>
42     <string name="sync_group_policy_plain">Unencrypted only</string>
```

```

43     <string name="sync_group_policy_password">Password-encrypted with group password</
string>
44     <string name="sync_remove_group_confirm_title">Remove group?</string>
45     <string name="sync_remove_group_confirm_message">Remove sync group "%1$s"?</string>
46     <string name="sync_remove_storage_confirm_title">Remove storage?</string>
47     <string name="sync_remove_storage_confirm_message">Remove storage "%1$s" from the group?
</string>
48     <string name="sync_confirm_delete">Delete</string>
49     <string name="sync_cancel">Cancel</string>
50     <string name="sync_msg_group_created">Created group %1$s</string>
51     <string name="sync_msg_group_removed">Group removed</string>
52     <string name="sync_msg_storage_added">Storage added to %1$s</string>
53     <string name="sync_msg_storage_removed">Storage removed from %1$s</string>
54     <string name="sync_msg_storage_already_added">Storage is already in the group</string>
55     <string name="sync_msg_only_plain_storage_allowed">Only unencrypted storages can be
added to sync groups</string>
56     <string name="sync_msg_storage_encryption_key_required">Encrypted storage requires the
password (open it before adding)</string>
57     <string name="sync_msg_storage_incompatible_encryption">Storage is not compatible with
the group encryption policy</string>
58     <string name="sync_msg_sync_already_running">Sync is already running</string>
59     <string name="sync_msg_blocked_during_sync">Wait for sync to finish</string>
60     <string name="sync_encryption_unknown">Unknown</string>
61     <string name="sync_storage_encryption_line">Encryption: %1$s</string>
62     <string name="sync_storage_missing_title">Not found in current vaults</string>
63     <string name="sync_storage_pending_vault_scan">Waiting: vault storage list is still
loading</string>
64     <string name="sync_storage_not_in_vaults">Not in storage tree (removed, different
account, or init failed)</string>
65     <string name="sync_storage_unreachable">Storage unavailable (vault or network)</string>
66     <string name="no_name">&lt;no name&gt;</string>
67     <string name="show_storage_item_menu">Storage menu</string>
68     <string name="storage_row_task_running_cd">Operation in progress for this storage</
string>
69     <string name="storage_menu_busy">%1$s (task running)</string>
70     <string name="rename">Rename</string>
71     <string name="remove">Remove</string>
72     <string name="encrypt">Encryption</string>
73     <string name="new_name_title">New name</string>
74     <string name="remove_confirmation_dialog">Remove storage "%1$s"?</string>
75     <string name="storage_lock_actions">Encryption actions</string>
76     <string name="storage_sync_lock_checking">Checking lock...</string>
77     <string name="storage_sync_unlock_action">Clear sync lock</string>
78     <string name="storage_sync_not_locked">Sync is not locked</string>
79     <string name="storage_field_available">Available: %1$s</string>
80     <string name="storage_value_yes">yes</string>
81     <string name="storage_value_no">no</string>

```



```

82     <string name="storage_field_files">Files: %1$s</string>
83     <string name="storage_field_size">Size: %1$s</string>
84     <string name="storage_field_virtual">Virtual: %1$s</string>
85     <string name="storage_unavailable_hint">Storage unavailable</string>
86     <string name="storage_meta_unavailable_hint">Metadata unavailable – rename, encryption,
and open are disabled</string>
87     <string name="storage_status_meta_unavailable">Metadata unavailable</string>
88     <string name="storage_home_meta_unavailable">Storage metadata could not be loaded. 2FA
and text secrets are unavailable.</string>
89     <string name="storage_menu_unavailable">Unavailable: %1$s</string>
90     <string name="storage_status_not_encrypted">Not encrypted</string>
91     <string name="storage_status_encrypted_open">Encrypted (open)</string>
92     <string name="storage_status_encrypted_closed">Encrypted (locked)</string>
93     <string name="vault_fab_add_storage_cd">Create storage</string>
94     <string name="vault_fab_add_storage_disabled_cd">Create unavailable: vault offline</
string>
95     <string name="vault_fab_add_storage_busy_cd">Storage creation already running</string>
96     <string name="vault_msg_storage_pipeline_busy">An operation is already running for this
storage</string>
97     <string name="vault_msg_vault_list_mutation_busy">Storage list is changing – please
wait</string>
98     <string name="vault_msg_rescan_already_in_progress">Storage scan is already in
progress</string>
99     <string name="vault_unavailable_banner">Vault unavailable. Check network, path, or
unlock.</string>
100    <string name="vault_loading_storages">Loading storage list...</string>
101    <string name="vault_empty_list_hint">No folders yet. Create storage with "+" when
available.</string>
102    <string name="vault_empty_list_hint_remote">No storages found on the remote vault. Tap
rescan if folders already exist on the server, or create one with "+" when available.</
string>
103    <string name="vault_rescan_storages_action">Rescan storages</string>
104    <string name="task_pipeline_title">Task queue</string>
105    <string name="task_pipeline_jobs">Tasks</string>
106    <string name="task_pipeline_log">Log</string>
107    <string name="task_pipeline_cancel_all">Cancel all</string>
108    <string name="task_pipeline_open">Open task queue</string>
109    <string name="task_pipeline_run_test">Test task</string>
110    <string name="task_pipeline_test_dialog_title">Test task parameters</string>
111    <string name="task_pipeline_test_dialog_duration">Duration: %1$d s</string>
112    <string name="task_pipeline_test_dialog_start">Start</string>
113    <string name="task_pipeline_test_dialog_cancel">Cancel</string>
114    <string name="task_pipeline_test_dialog_infinity">Infinite (indeterminate progress)</
string>
115    <string name="task_pipeline_test_running">Test task (%1$d s)</string>
116    <string name="task_pipeline_test_running_infinity">Test task (%1$d s, ∞)</string>
117    <string name="task_state_queued">Queued</string>
118    <string name="task_state_running">Running</string>
119    <string name="task_state_completed">Completed</string>

```

```

120     <string name="task_state_cancelled">Cancelled</string>
121     <string name="task_state_failed">Error: %1$s</string>
122     <string name="task_title_dump_storage_log">Export tree to log</string>
123     <string name="task_title_create_storage">Create storage</string>
124     <string name="task_title_enable_encryption">Enable encryption</string>
125     <string name="task_title_open_encrypted_storage">Decrypt and open storage</string>
126     <string name="task_progress_decrypt_running">Decrypting</string>
127     <string name="task_progress_dump_storage_log">Scanning tree</string>
128     <string name="task_progress_create_storage">Creating storage</string>
129     <string name="task_progress_enable_encryption">Encrypting</string>
130     <string name="task_progress_close_storage">Closing storage</string>
131     <string name="task_progress_disable_encryption">Clearing content</string>
132     <string name="task_progress_rename_storage">Renaming</string>
133     <string name="task_progress_remove_storage">Removing</string>
134     <string name="task_progress_clear_sync_lock">Clearing sync lock</string>
135     <string name="task_progress_add_remote_vault">Adding</string>
136     <string name="task_progress_remove_remote_vault">Removing</string>
137     <string name="task_progress_retry_remote_vault">Connecting</string>
138     <string name="task_progress_rescan_vault_storages">Scanning storages</string>
139     <string name="task_progress_save_2fa_token">Saving</string>
140     <string name="task_progress_delete_2fa_token">Removing</string>
141     <string name="task_progress_save_text_secret">Saving</string>
142     <string name="task_progress_delete_text_secret">Removing</string>
143     <string name="task_title_close_encrypted_storage">Close encrypted storage</string>
144     <string name="task_title_disable_encryption">Disable encryption</string>
145     <string name="task_title_rename_storage">Rename storage</string>
146     <string name="task_title_remove_storage">Remove storage</string>
147     <string name="task_title_clear_sync_lock">Clear sync lock</string>
148     <string name="task_title_add_remote_vault">Add remote vault</string>
149     <string name="task_title_remove_remote_vault">Remove remote vault</string>
150     <string name="task_title_retry_remote_vault">Retry remote vault connection</string>
151     <string name="task_title_rescan_vault_storages">Rescan vault storages</string>
152     <string name="task_title_storage_sync">Storage sync (%1$s)</string>
153     <string name="task_title_save_2fa_token">Save 2FA token</string>
154     <string name="task_title_delete_2fa_token">Delete 2FA token</string>
155     <string name="task_title_save_text_secret">Save text secret</string>
156     <string name="task_title_delete_text_secret">Delete text secret</string>
157     <string name="error_storage_not_found">Storage not found</string>
158     <string name="error_storage_locked_view">Open decrypted storage view to use this
section</string>
159     <string name="error_secret_not_found">Secret not found</string>
160     <string name="error_storage_not_writable">Storage is not writable</string>
161     <string name="error_file_not_found">File not found</string>
162     <string name="error_incorrect_password">Incorrect password</string>
163     <string name="error_storage_not_encrypted">Storage is not encrypted</string>

```



```

164     <string name="error_enc_info_missing">Encryption metadata missing</string>
165     <string name="error_delete_root_forbidden">Cannot delete storage root</string>
166     <string name="error_not_a_file">Expected a file</string>
167     <string name="error_not_a_directory">Expected a folder</string>
168     <string name="error_path_is_file">Path points to a file, not a folder</string>
169     <string name="error_cannot_write_over_directory">Cannot write over a folder</string>
170     <string name="error_unexpected_state">Storage in unexpected state</string>
171     <string name="error_network">Network or server error</string>
172     <string name="error_disk_resource_locked">Resource is temporarily locked. Try again
later.</string>
173     <string name="error_unknown">Something went wrong</string>
174     <string name="sync_error_group_not_found">Sync group not found</string>
175     <string name="vault_link_error_auth">Sign-in failed</string>
176     <string name="vault_link_error_not_registered">Sign-in is not ready. Restart the app.</
string>
177     <string name="vault_link_error_unknown">Sign-in failed</string>
178     <string name="vault_link_error_unsupported_brand">This provider is not supported</
string>
179     <string name="msg_encryption_enabled">Encryption enabled</string>
180     <string name="msg_encryption_enabled_open_failed">Encryption enabled; unlock the storage
manually to view contents</string>
181     <string name="msg_storage_already_encrypted">Storage is already encrypted</string>
182     <string name="msg_storage_not_empty">Storage is not empty</string>
183     <string name="msg_storage_empty_state_unknown">Could not determine if storage is empty</
string>
184     <string name="msg_unsupported_storage_type">Unsupported storage type</string>
185     <string name="msg_encryption_disabled">Encryption disabled</string>
186     <string name="msg_invalid_storage_for_sync_lock">Invalid storage</string>
187     <string name="msg_sync_lock_cleared">Sync lock cleared</string>
188     <string name="remote_vaults_add_cd">Add remote vault</string>
189     <string name="remote_vaults_empty_hint">No remote vaults yet. Tap "+" to add Yandex.</
string>
190     <string name="remote_vaults_add_title">Add vault</string>
191     <string name="remote_vaults_add_pick_provider">Choose a provider:</string>
192     <string name="remote_vaults_provider_yandex">Yandex</string>
193     <string name="remote_vaults_add_cancel">Cancel</string>
194     <string name="remote_vault_type_yandex">Yandex</string>
195     <string name="remote_vault_unavailable">Vault temporarily unavailable</string>
196     <string name="remote_vault_retry_action">Retry connection</string>
197     <string name="remote_vault_retrying">Connecting...</string>
198     <string name="remote_vault_delete_cd">Remove remote vault</string>
199     <string name="remote_vault_remove_title">Remove remote vault?</string>
200     <string name="remote_vault_remove_message">Remove "%1$s" from this device? Server data
is not deleted.</string>
201     <string name="dialog_cancel">Cancel</string>
202     <string name="dialog_ok">OK</string>
203     <string name="dialog_encryption_enable_title">Enable encryption</string>

```

```

204     <string name="dialog_password_label">Password</string>
205     <string name="dialog_encrypt_paths">Encrypt paths</string>
206     <string name="dialog_apply">Apply</string>
207     <string name="dialog_open_encrypted_title">Open encrypted storage</string>
208     <string name="dialog_remember_password">Remember password</string>
209     <string name="dialog_open">Open</string>
210     <string name="dialog_close">Close</string>
211     <string name="dialog_disable_encryption">Disable encryption</string>
212     <string name="dialog_done">Done</string>
213     <string name="vault_type_local_device">Local device</string>
214     <string name="vault_type_remote">Remote: %1$s</string>
215     <string name="vault_type_unknown">Unknown type</string>
216     <string name="vault_title_local">Local vault</string>
217     <string name="vault_title_unknown">Unknown vault</string>
218     <string name="enc_status_not_encrypted">Not encrypted</string>
219     <string name="enc_status_encrypted_open">Encrypted (open)</string>
220     <string name="enc_status_encrypted">Encrypted</string>
221     <string name="text_edit_screen_placeholder">Content: %1$s</string>
222     <string name="storage_home_unnamed_storage">Storage</string>
223     <string name="storage_home_status_line">Status: %1$s, %2$s</string>
224     <string name="storage_home_status_available">available</string>
225     <string name="storage_home_status_unavailable">unavailable</string>
226     <string name="storage_home_status_encrypted">encrypted</string>
227     <string name="storage_home_status_not_encrypted">not encrypted</string>
228     <string name="storage_home_two_fa_title">2FA tokens (%1$d)</string>
229     <string name="storage_home_two_fa_subtitle">Two-factor authentication codes and
secrets</string>
230     <string name="storage_home_text_secrets_title">Text secrets (%1$d)</string>
231     <string name="storage_home_text_secrets_subtitle">Notes, tokens, and arbitrary key-value
pairs</string>
232     <string name="storage_home_future_sections">Files, Media, and more will appear here
soon.</string>
233     <string name="two_fa_add_token">Add token</string>
234     <string name="two_fa_empty_state">No 2FA tokens yet</string>
235     <string name="two_fa_create_title">New 2FA token</string>
236     <string name="two_fa_edit_title">Edit 2FA token</string>
237     <string name="two_fa_field_issuer">Service</string>
238     <string name="two_fa_field_account">Account</string>
239     <string name="two_fa_field_secret">Secret</string>
240     <string name="two_fa_field_notes_optional">Note (optional)</string>
241     <string name="two_fa_field_algorithm">Algorithm (SHA1, SHA256, SHA512)</string>
242     <string name="two_fa_field_digits_value">Digits: %1$d</string>
243     <string name="two_fa_field_period_seconds_value">Refresh period: %1$d s</string>
244     <string name="two_fa_code_unavailable">-----</string>
245     <string name="two_fa_code_refresh_label">Refresh in</string>

```

```

246     <string name="two_fa_code_refresh_seconds">%1$d s</string>
247     <string name="two_fa_code_invalid_secret">Invalid secret or format</string>
248     <string name="two_fa_scan_qr_action">Scan QR</string>
249     <string name="two_fa_scan_qr_title">Scan TOTP QR code</string>
250     <string name="two_fa_scan_qr_invalid">QR code does not contain a valid otpauth://totp
URI</string>
251     <string name="two_fa_camera_permission_required">Camera permission is required to scan
QR</string>
252     <string name="text_secret_create">Create secret</string>
253     <string name="text_secret_edit">Edit secret</string>
254     <string name="text_secret_title">Title</string>
255     <string name="text_secret_empty_state">No text secrets yet</string>
256     <string name="text_secret_items_count">Items: %1$d</string>
257     <string name="text_secret_item_without_label">Untitled</string>
258     <string name="text_secret_more_fields">more fields: %1$d</string>
259     <string name="text_secret_item_label_optional">Label (optional)</string>
260     <string name="text_secret_item_value">Value</string>
261     <string name="text_secret_add_item">Add field</string>
262     <string name="text_secret_copy_value">Copy value</string>
263     <string name="save">Save</string>
264     <string name="cancel">Cancel</string>
265     <string name="edit">Edit</string>
266     <string name="settings_language_section">Language</string>
267     <string name="settings_language_system">System default</string>
268     <string name="settings_language_english">English</string>
269     <string name="settings_language_russian">Russian</string>
270     <string name="task_pipeline_test_elapsed">Elapsed: %1$d s / %2$d s</string>
271     <string name="text_secret_clipboard_fallback_label">value</string>
272     <string name="sync_progress_no_groups">Storage sync: no groups configured</string>
273     <string name="sync_progress_started">Storage sync: started</string>
274     <string name="sync_progress_completed">Storage sync: completed</string>
275     <string name="sync_progress_group_preparing">Storage sync: group "%1$s" preparing</
string>
276     <string name="sync_progress_group_not_found">Storage sync: group "%1$s" not found</
string>
277     <string name="sync_progress_group_skipped_few_storages">Storage sync: group "%1$s"
skipped (need at least 2 storages)</string>
278     <string name="sync_progress_group_skipped_incompatible">Storage sync: group "%1$s"
skipped (incompatible encryption: %2$d)</string>
279     <string name="sync_progress_group_acquiring_locks">Storage sync: group "%1$s" acquiring
locks</string>
280     <string name="sync_progress_group_lock">Storage sync: group "%1$s" lock %2$d/%3$d</
string>
281     <string name="sync_progress_group_lock_failed">Storage sync: group "%1$s" lock failed,
group skipped</string>
282     <string name="sync_progress_group_reading_journals">Storage sync: group "%1$s" reading
journals</string>
283     <string name="sync_progress_group_cancelled">Storage sync: group "%1$s" cancelled by
newer run</string>

```

```

284     <string name="sync_progress_group_journal">Storage sync: group "%1$s" journal %2$d/
%3$d</string>
285     <string name="sync_progress_group_no_entries">Storage sync: group "%1$s" no journal
entries</string>
286     <string name="sync_progress_group_entry">Storage sync: group "%1$s" entry %2$d/%3$d</
string>
287     <string name="sync_progress_group_completed">Storage sync: group "%1$s" completed</
string>
288     <string name="sync_progress_group_renewing_locks">Storage sync: group "%1$s" renewing
locks</string>
289     <string name="sync_progress_group_lock_renewal_failed">Storage sync: group "%1$s" lock
renewal failed</string>
290     <string name="task_progress_clear_content">%1$d / %2$d</string>
291     <string name="task_log_sync_started">Storage sync started (%1$s)</string>
292     <string name="task_log_sync_finished">Storage sync finished (%1$s)</string>
293     <string name="task_log_sync_failed">Storage sync failed (%1$s): %2$s</string>
294     <string name="task_sync_trigger_debounce">debounce</string>
295     <string name="task_sync_trigger_sync_tab">sync-tab</string>
296     <string name="task_sync_trigger_background">background</string>
297     <string name="task_log_enumerating">Enumerating files and directories...</string>
298     <string name="task_log_creating_storage">Creating storage...</string>
299     <string name="task_log_storage_created">Storage created</string>
300     <string name="task_log_checking_storage">Checking storage...</string>
301     <string name="task_log_encrypting">Encrypting...</string>
302     <string name="task_log_encryption_enabled">Encryption enabled</string>
303     <string name="task_log_already_encrypted">Storage is already encrypted</string>
304     <string name="task_log_not_empty">Storage is not empty</string>
305     <string name="task_log_empty_unknown">Cannot determine whether storage is empty</string>
306     <string name="task_log_unsupported_type">Unsupported storage type</string>
307     <string name="task_log_enable_encryption_failed">Failed to enable encryption</string>
308     <string name="task_log_opening_storage">Opening encrypted storage...</string>
309     <string name="task_log_storage_opened">Storage opened</string>
310     <string name="task_log_open_storage_failed">Failed to open encrypted storage</string>
311     <string name="task_log_closing_storage">Closing storage...</string>
312     <string name="task_log_storage_closed">Storage closed</string>
313     <string name="task_log_close_storage_failed">Failed to close encrypted storage</string>
314     <string name="task_log_disabling_encryption">Disabling encryption...</string>
315     <string name="task_log_encryption_disabled">Encryption disabled</string>
316     <string name="task_log_disable_encryption_failed">Failed to disable encryption</string>
317     <string name="task_log_renaming">Renaming...</string>
318     <string name="task_log_renamed">Renamed</string>
319     <string name="task_log_rename_failed">Rename failed</string>
320     <string name="task_log_removing_storage">Removing storage...</string>
321     <string name="task_log_removed">Removed</string>
322     <string name="task_log_remove_failed">Remove failed</string>
323     <string name="task_log_invalid_storage">Invalid storage</string>

```

```

324     <string name="task_log_clearing_sync_lock">Clearing sync lock...</string>
325     <string name="task_log_sync_lock_cleared">Sync lock cleared</string>
326     <string name="task_log_clear_sync_lock_failed">Failed to clear sync lock</string>
327     <string name="task_log_adding_vault">Adding vault...</string>
328     <string name="task_log_vault_added">Vault added</string>
329     <string name="task_log_add_vault_failed">Failed to add vault</string>
330     <string name="task_log_removing_remote_vault">Removing remote vault...</string>
331     <string name="task_log_remote_vault_removed">Remote vault removed</string>
332     <string name="task_log_remove_vault_failed">Failed to remove vault</string>
333     <string name="task_log_retrying_vault">Retrying remote vault connection...</string>
334     <string name="task_log_retry_requested">Retry requested</string>
335     <string name="task_log_retry_vault_failed">Failed to retry remote vault</string>
336     <string name="task_log_rescanning_vault_storages">Rescanning storages on remote vault...</
string>
337     <string name="task_log_rescan_vault_storages_done">Storage list updated</string>
338     <string name="task_log_rescan_vault_storages_failed">Failed to rescan storages</string>
339     <string name="task_log_test_started">Test task started for %1$d s</string>
340     <string name="task_log_test_finished">Test task finished</string>
341 </resources>
342

```

Исходный файл ui/src/test/java/com/github/nullptroma/wallenc/ui/navigation/
WallencDeepLinksTest.kt

```
1 package com.github.nullptroma.wallenc.ui.navigation
2
3 import android.content.Intent
4 import android.net.Uri
5 import org.junit.Assert.assertFalse
6 import org.junit.Assert.assertTrue
7 import org.junit.Test
8 import org.junit.runner.RunWith
9 import org.robolectric.RobolectricTestRunner
10 import org.robolectric.annotation.Config
11
12 @RunWith(RobolectricTestRunner::class)
13 @Config(sdk = [33])
14 class WallencDeepLinksTest {
15
16     @Test
17     fun matchesWallencViewIntent() {
18         val intent = Intent(Intent.ACTION_VIEW,
19 Uri.parse(WallencDeepLinks.MAIN_URI_PATTERN))
20         assertTrue(intent.matchesWallencDeepLink())
21     }
22
23     @Test
24     fun rejectsUnrelatedIntent() {
25         val intent = Intent(Intent.ACTION_MAIN)
26         assertFalse(intent.matchesWallencDeepLink())
27     }
28
29     @Test
30     fun matchesTasksAndSettingsHosts() {
31         assertTrue(
32             Intent(Intent.ACTION_VIEW, Uri.parse(WallencDeepLinks.TASKS_URI_PATTERN))
33                 .matchesWallencDeepLink(),
34         )
35         assertTrue(
36             Intent(Intent.ACTION_VIEW, Uri.parse(WallencDeepLinks.SETTINGS_URI_PATTERN))
37                 .matchesWallencDeepLink(),
38         )
39     }
40 }
```

Исходный файл ui/src/test/java/com/github/nullptroma/wallenc/ui/resources/
TaskProgressLabelsTest.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
4 import com.github.nullptroma.wallenc.domain.tasks.VaultTaskStep
5 import com.github.nullptroma.wallenc.ui.R
6 import org.junit.Assert.assertEquals
7 import org.junit.Test
8
9 class TaskProgressLabelsTest {
10
11     private val resolver = object : UiStringResolver {
12         override fun invoke(id: Int, vararg formatArgs: Any): String = "res:$id"
13
14         override fun plurals(id: Int, quantity: Int, vararg formatArgs: Any): String =
15             "plural:$id"
16     }
17
18     @Test
19     fun syncNoGroups_mapsToStringRes() {
20         val text = TaskProgressLabel.SyncNoGroups.resolve(resolver)
21         assertEquals("res:${R.string.sync_progress_no_groups}", text)
22     }
23
24     @Test
25     fun vaultTask_mapsToStringRes() {
26         val text = TaskProgressLabel.VaultTask(VaultTaskStep.Save2FaToken).resolve(resolver)
27         assertEquals("res:${R.string.task_progress_save_2fa_token}", text)
28     }
29
30     @Test
31     fun clearContentProgress_mapsToStringRes() {
32         val text = TaskProgressLabel.ClearContentProgress(3, 10).resolve(resolver)
33         assertEquals("res:${R.string.task_progress_clear_content}", text)
34     }
35 }
```


Исходный файл ui/src/test/java/com/github/nullptroma/wallenc/ui/resources/
WallencUserNotificationMappingTest.kt

```
1 package com.github.nullptroma.wallenc.ui.resources
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.ui.R
5 import org.junit.Assert.assertEquals
6 import org.junit.Test
7
8 class WallencUserNotificationMappingTest {
9
10     @Test
11     fun mapsFeatureStorageNotFound() {
12         val notification = WallencException.Feature.StorageNotFound().toUserNotification()
13         assertEquals(R.string.error_storage_not_found, notification.id)
14     }
15
16     @Test
17     fun mapsStorageIncorrectKey() {
18         val notification = WallencException.Storage.IncorrectKey().toUserNotification()
19         assertEquals(R.string.error_incorrect_password, notification.id)
20     }
21
22     @Test
23     fun mapsUnknown() {
24         val notification = WallencException.Unknown(Exception("x")).toUserNotification()
25         assertEquals(R.string.error_unknown, notification.id)
26     }
27 }
28
```


Исходный файл ui/src/test/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/StorageNavigationRoutesSmokeTest.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage
2
3 import
4 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretDetailsRoute
5 import
6 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretEditRoute
7 import
8 com.github.nullptroma.wallenc.ui.screens.main.screens.storage.secrets.TextSecretsRoute
9 import com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa.TwoFaTokensRoute
10 import org.junit.Assert.assertEquals
11 import org.junit.Assert.assertNull
12 import org.junit.Test
13
14 class StorageNavigationRoutesSmokeTest {
15
16     @Test
17     fun storageHomeRouteCarriesVaultAndStorageIds() {
18         val route = StorageHomeRoute(
19             vaultUuid = "vault-1",
20             storageUuid = "storage-1",
21         )
22         assertEquals("vault-1", route.vaultUuid)
23         assertEquals("storage-1", route.storageUuid)
24     }
25
26     @Test
27     fun textSecretsRoutesCarryRequiredArguments() {
28         val listRoute = TextSecretsRoute(storageUuid = "storage-1")
29         val detailsRoute = TextSecretDetailsRoute(storageUuid = "storage-1", secretId =
30 "secret-1")
31         val editRoute = TextSecretEditRoute(storageUuid = "storage-1", secretId = null)
32         val twoFaRoute = TwoFaTokensRoute(storageUuid = "storage-1")
33
34         assertEquals("storage-1", listRoute.storageUuid)
35         assertEquals("secret-1", detailsRoute.secretId)
36         assertNull(editRoute.secretId)
37         assertEquals("storage-1", twoFaRoute.storageUuid)
38     }
39 }
```

Исходный файл ui/src/test/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/storage/twofa/OtpAuthUriParserTest.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.storage.twofa
2
3 import org.junit.Assert.assertEquals
4 import org.junit.Assert.assertNotNull
5 import org.junit.Assert.assertNull
6 import org.junit.Test
7 import org.junit.runner.RunWith
8 import org.robolectric.RobolectricTestRunner
9 import org.robolectric.annotation.Config
10
11 @RunWith(RobolectricTestRunner::class)
12 @Config(sdk = [33])
13 class OtpAuthUriParserTest {
14
15     @Test
16     fun parsesStandardTotpUri() {
17         val uri = "otpauth://totp/GitHub:user@example.com?secret=JBSWY3DPEHPK3PXP&issuer=
GitHub&digits=6&period=30"
18         val parsed = parseOtpAuthTotpUri(uri)
19         assertNotNull(parsed)
20         assertEquals("GitHub", parsed!!.issuer)
21         assertEquals("user@example.com", parsed.account)
22         assertEquals("JBSWY3DPEHPK3PXP", parsed.secret)
23         assertEquals(6, parsed.digits)
24         assertEquals(30, parsed.periodSeconds)
25         assertEquals("SHA1", parsed.algorithm)
26     }
27
28     @Test
29     fun rejectsNonOtpauthScheme() {
30         assertNull(parseOtpAuthTotpUri("https://example.com"))
31     }
32
33     @Test
34     fun rejectsMissingSecret() {
35         assertNull(parseOtpAuthTotpUri("otpauth://totp/Test:account"))
36     }
37 }
38
```

Исходный файл ui/src/test/java/com/github/nullptroma/wallenc/ui/screens/main/
screens/tasks/TaskPipelineViewModelTest.kt

```
1 package com.github.nullptroma.wallenc.ui.screens.main.screens.tasks
2
3 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
4 import com.github.nullptroma.wallenc.task.runtime.TaskOrchestrator
5 import com.github.nullptroma.wallenc.ui.resources.UiStringResolver
6 import io.mockk.every
7 import io.mockk.mockk
8 import kotlinx.coroutines.Dispatchers
9 import kotlinx.coroutines.delay
10 import kotlinx.coroutines.runBlocking
11 import kotlinx.coroutines.withTimeout
12 import org.junit.Assert.assertTrue
13 import org.junit.Test
14 import org.junit.runner.RunWith
15 import org.robolectric.RobolectricTestRunner
16 import org.robolectric.annotation.Config
17
18 @RunWith(RobolectricTestRunner::class)
19 @Config(sdk = [33])
20 class TaskPipelineViewModelTest {
21
22     @Test
23     fun startTestTaskEnqueuesWork() = runBlocking {
24         val orchestrator = TaskOrchestrator(Dispatchers.Default)
25         val uiStrings = mockk<UiStringResolver>()
26         every { uiStrings.invoke(any<Int>(), any()) } returns "Test task"
27         every { uiStrings.invoke(any<Int>()) } returns "Test"
28         every { uiStrings.plurals(any(), any(), any()) } returns "Plural"
29         val viewModel = TaskPipelineViewModel(orchestrator, uiStrings)
30
31         viewModel.startTestTask(durationSec = 0, infinityIndeterminateProgress = false)
32
33         withTimeout(5_000) {
34             while (true) {
35                 val task = orchestrator.pipelineState.value.tasks.singleOrNull()
36                 if (task?.state is TaskRunState.Completed) break
37                 delay(25)
38             }
39         }
40         val task = orchestrator.pipelineState.value.tasks.single()
41         assertTrue(task.state is TaskRunState.Completed)
42     }
```

43 }
44

ПРИЛОЖЕНИЕ А.6

Модуль :domain-vault

Исходный файл domain-vault/consumer-rules.pro

Исходный файл domain-vault/proguard-rules.pro

```
1  # Add project specific ProGuard rules here.
2  # You can control the set of applied configuration files using the
3  # proguardFiles setting in build.gradle.
4  #
5  # For more details, see
6  #   http://developer.android.com/guide/developing/tools/proguard.html
7
8  # If your project uses WebView with JS, uncomment the following
9  # and specify the fully qualified class name to the JavaScript interface
10 # class:
11 #-keepclassmembers class fqcn.of.javascript.interface.for.webview {
12 #   public *;
13 #}
14
15 # Uncomment this to preserve the line number information for
16 # debugging stack traces.
17 #-keepattributes SourceFile,LineNumberTable
18
19 # If you keep the line number information, uncomment this to
20 # hide the original source file name.
21 #-renamesourcefileattribute SourceFile
```

Исходный файл domain-vault/src/main/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest>
3
4  </manifest>
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/errors/VaultThrowableMapping.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.errors
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.domain.errors.toWallencException
5 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskAuthException
6 import retrofit2.HttpException
7 import java.io.FileNotFoundException
8 import java.io.IOException
9
10 fun Throwable.toVaultWallencException(): WallencException = when (this) {
11     is WallencException -> this
12     is YandexDiskAuthException -> WallencException.Auth.Failed()
13     is HttpException -> WallencException.Network.HttpFailed(
14         operation = "http",
15         statusCode = code(),
16         cause = this,
17     )
18     is FileNotFoundException -> WallencException.Storage.FileNotFound()
19     is IOException -> mapVaultIo(this)
20     is IllegalStateException -> mapIllegalState(this)
21     is IllegalArgumentException -> mapIllegalArgument(this)
22     else -> toWallencException()
23 }
24
25 private fun mapVaultIo(e: IOException): WallencException {
26     val msg = e.message.orEmpty()
27     return when {
28         msg.contains("OAuth token is missing", ignoreCase = true) ->
29             WallencException.Auth.TokenMissing()
30         msg.contains("HTTP 423", ignoreCase = true) || msg.contains("423 after retries",
31             ignoreCase = true) ->
32             WallencException.Network.ResourceLocked()
33         msg.equals("timeout", ignoreCase = true) ||
34             msg.contains("timed out", ignoreCase = true) ||
35             msg.contains("async operation timed out", ignoreCase = true) ->
36             WallencException.Network.OperationTimedOut()
37         msg.contains("async operation failed", ignoreCase = true) ->
38             WallencException.Network.OperationFailed()
39         else -> WallencException.Network.IoFailed(e)
40     }
41 }
```



```

42     private fun mapIllegalState(e: IllegalStateException): WallencException = when {
43         e.message == "Not a file" -> WallencException.Storage.NotAFile()
44         e.message == "Not a directory" -> WallencException.Storage.NotADirectory()
45         e.message?.startsWith("Path segment is a file:") == true ->
WallencException.Storage.PathIsFile()
46         e.message?.startsWith("Cannot openWrite over directory:") == true ->
WallencException.Storage.CannotWriteOverDirectory()
47         e.message?.startsWith("Expected file after upload:") == true ->
WallencException.Storage.UnexpectedState()
48         else -> WallencException.Storage.UnexpectedState()
49     }
50
51
52
53     private fun mapIllegalArgument(e: IllegalArgumentException): WallencException = when {
54         e.message == "Deleting root path is forbidden" ->
WallencException.Storage.DeleteRootForbidden()
55         else -> WallencException.Unknown(e)
56     }
57

```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/model/StorageKeyMap.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.model
2
3 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
4 import java.util.UUID
5
6 data class StorageKeyMap(
7     val sourceUuid: UUID,
8     val key: EncryptKey
9 )
10
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/model/YandexAccount.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.model
2
3 data class YandexAccount(
4     val vaultUuid: String,
5     val yandexUserId: String,
6     val email: String,
7     val oauthToken: String,
8 )
9
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/YandexDiskApi.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk
2
3 import
4 com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.CustomPropertiesPatchDto
5 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.DiskInfoDto
6 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.LinkDto
7 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.OperationStatusDto
8 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.ResourceDto
9 import okhttp3.ResponseBody
10 import retrofit2.Response
11 import retrofit2.http.Body
12 import retrofit2.http.DELETE
13 import retrofit2.http.GET
14 import retrofit2.http.Headers
15 import retrofit2.http.PATCH
16 import retrofit2.http.PUT
17 import retrofit2.http.Query
18 import retrofit2.http.Url
19
20 interface YandexDiskApi {
21     @GET("v1/disk/")
22     suspend fun getDisk(): DiskInfoDto
23
24     @GET("v1/disk/resources")
25     suspend fun listResources(
26         @Query("path") path: String,
27         @Query("limit") limit: Int,
28         @Query("offset") offset: Int,
29         @Query("sort") sort: String? = null,
30         @Query("fields") fields: String,
31     ): ResourceDto
32
33     @GET("v1/disk/resources")
34     suspend fun getResource(
35         @Query("path") path: String,
36         @Query("fields") fields: String,
37     ): ResourceDto
38
39     @PUT("v1/disk/resources")
40     suspend fun createFolder(@Query("path") path: String): Response<ResponseBody>
41
```

```

42     @DELETE("v1/disk/resources")
43     suspend fun deleteResource(
44         @Query("path") path: String,
45         @Query("permanently") permanently: Boolean,
46     ): Response<ResponseBody>
47
48     @GET("v1/disk/resources/upload")
49     suspend fun getUploadLink(
50         @Query("path") path: String,
51         @Query("overwrite") overwrite: Boolean = true,
52     ): LinkDto
53
54     @GET("v1/disk/resources/download")
55     suspend fun getDownloadLink(@Query("path") path: String): LinkDto
56
57     @PATCH("v1/disk/resources")
58     @Headers("Content-Type: application/json")
59     suspend fun patchResource(
60         @Query("path") path: String,
61         @Body body: CustomPropertiesPatchDto,
62     ): Response<ResponseBody>
63
64     @GET
65     suspend fun getOperationByUrl(@Url url: String): OperationStatusDto
66 }
67

```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/YandexDiskApiFactory.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk
2
3 import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
4 import
5 com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository.YandexDiskRepository
6 import com.github.nullptroma.wallenc.domain.vault.ports.YandexAccountStore
7 import kotlinx.coroutines.CoroutineDispatcher
8 import kotlinx.coroutines.runBlocking
9 import okhttp3.OkHttpClient
10 import retrofit2.Retrofit
11 import retrofit2.converter.jackson.JacksonConverterFactory
12 import java.util.UUID
13 import java.util.concurrent.ConcurrentHashMap
14 import java.util.concurrent.TimeUnit
15
16 /**
17  * Фабрика REST-клиента Яндекс.Диска: отдельный [OkHttpClient] с OAuth на каждый vault,
18  * плюс «голый» клиент для PUT/GET по одноразовым upload/download URL.
19  */
20 class YandexDiskApiFactory(
21     private val accountRepository: YandexAccountStore,
22     private val ioDispatcher: CoroutineDispatcher,
23 ) {
24     /** Кеш OAuth-токена по vault, чтобы не дергать БД на каждый HTTP-запрос к cloud-api. */
25     private val oauthTokenCache = ConcurrentHashMap<String, Pair<Long, String>>()
26
27     private val jackson = jacksonObjectMapper().apply { findAndRegisterModules() }
28
29     /** Без авторизации — только для одноразовых ссылок upload/download. */
30     val rawHttpClient: OkHttpClient by lazy {
31         newHttpClientBuilder().build()
32     }
33
34     /**
35      * [tokenProvider] вызывается на каждый HTTP-запрос к cloud-api (свежий токен из БД).
36      */
37     fun createAuthenticatedApi(tokenProvider: () -> String?): YandexDiskApi {
38         val client = newHttpClientBuilder()
39             .addInterceptor { chain ->
40                 val token = tokenProvider()
41                 ?: throw java.io.IOException("Yandex OAuth token is missing")
```

```

42         val req = chain.request().newBuilder()
43             .header("Authorization", "OAuth $token")
44             .header("Accept", "application/json")
45             .build()
46         chain.proceed(req)
47     }
48     .build()
49     return Retrofit.Builder()
50         .baseUrl(BASE_URL)
51         .client(client)
52         .addConverterFactory(JacksonConverterFactory.create(jackson))
53         .build()
54         .create(YandexDiskApi::class.java)
55 }
56
57 /**
58  * OAuth-токен загружается один раз при создании API (не в OkHttpClient interceptor).
59  * При 401 см. [YandexDiskAuthException] и повторную привязку vault.
60  */
61 fun createApiForVault(vaultUuid: UUID): YandexDiskApi {
62     val id = vaultUuid.toString()
63     val token = runBlocking(ioDispatcher) {
64         accountRepository.getByVaultUuid(id)?.oauthToken
65     } ?: throw java.io.IOException("Yandex OAuth token is missing")
66     oauthTokenCache[id] = System.currentTimeMillis() to token
67     return createAuthenticatedApi {
68         oauthTokenCache[id]?.second ?: token
69     }
70 }
71
72 fun invalidateTokenCache(vaultUuid: UUID) {
73     oauthTokenCache.remove(vaultUuid.toString())
74 }
75
76 companion object {
77     const val BASE_URL = "https://cloud-api.yandex.net/"
78
79     private const val CONNECT_TIMEOUT_SEC = 30L
80     private const val READ_TIMEOUT_SEC = 120L
81     private const val WRITE_TIMEOUT_SEC = 120L
82
83     fun newHttpClientBuilder(): OkHttpClient.Builder =
84         OkHttpClient.Builder()
85             .connectTimeout(CONNECT_TIMEOUT_SEC, TimeUnit.SECONDS)
86             .readTimeout(READ_TIMEOUT_SEC, TimeUnit.SECONDS)

```

```

87         .writeTimeout(WRITE_TIMEOUT_SEC, TimeUnit.SECONDS)
88     fun createRepositoryWithToken(
89         oauthToken: String,
90         ioDispatcher: CoroutineDispatcher,
91     ): YandexDiskRepository {
92         val client = newHttpClientBuilder()
93             .addInterceptor { chain ->
94                 chain.proceed(
95                     chain.request().newBuilder()
96                         .header("Authorization", "OAuth $oauthToken")
97                         .header("Accept", "application/json")
98                         .build(),
99                 )
100             }
101             .build()
102         val api = Retrofit.Builder()
103             .baseUrl(BASE_URL)
104             .client(client)
105             .addConverterFactory(JacksonConverterFactory.create(jacksonObjectMapper().findAndR
106             .build()
107             .create(YandexDiskApi::class.java)
108         return YandexDiskRepository(api, OkHttpClient(), ioDispatcher)
109     }
110 }
111 }
112

```


Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/YandexDiskExceptions.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk
2
3 import java.io.IOException
4
5 class YandexDiskAuthException(message: String? = null) : IOException(message)
6
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/dto/YandexDiskDto.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto
2
3 import com.fasterxml.jackson.annotation.JsonIgnoreProperties
4 import com.fasterxml.jackson.annotation.JsonProperty
5 import java.time.Instant
6
7 @JsonIgnoreProperties(ignoreUnknown = true)
8 data class DiskInfoDto(
9     @param:JsonProperty("trash_size") val trashSize: Long? = null,
10    @param:JsonProperty("total_space") val totalSpace: Long? = null,
11    @param:JsonProperty("used_space") val usedSpace: Long? = null,
12)
13
14 @JsonIgnoreProperties(ignoreUnknown = true)
15 data class LinkDto(
16    val href: String,
17    val method: String,
18    val templated: Boolean? = null,
19)
20
21 @JsonIgnoreProperties(ignoreUnknown = true)
22 data class EmbeddedResourceListDto(
23    val items: List<ResourceDto>? = null,
24    val total: Int? = null,
25    val path: String? = null,
26    val sort: String? = null,
27    val limit: Int? = null,
28    val offset: Int? = null,
29)
30
31 @JsonIgnoreProperties(ignoreUnknown = true)
32 data class ResourceDto(
33    val path: String? = null,
34    val type: String? = null,
35    val name: String? = null,
36    val size: Long? = null,
37    val modified: Instant? = null,
38    val created: Instant? = null,
39    @param:JsonProperty("mime_type") val mimeType: String? = null,
40    val md5: String? = null,
41    @param:JsonProperty("custom_properties") val customProperties: Map<String, Any?>? =
    null,
```

```
42         @param:JsonProperty("_embedded") val embedded: EmbeddedResourceListDto? = null,
43     )
44
45     @JsonIgnoreProperties(ignoreUnknown = true)
46     data class OperationStatusDto(
47         val status: String? = null,
48     )
49
50     @JsonIgnoreProperties(ignoreUnknown = true)
51     data class CustomPropertiesPatchDto(
52         @param:JsonProperty("custom_properties") val customProperties: Map<String, String>,
53     )
54
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/dto/YandexVaultPersistedStats.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto
2
3 import com.fasterxml.jackson.annotation.JsonIgnoreProperties
4
5 @JsonIgnoreProperties(ignoreUnknown = true)
6 data class YandexVaultPersistedStats(
7     val totalBytes: Long = 0L,
8     val fileCount: Int = 0,
9 )
10
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/repository/YandexDiskRepository.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository
2
3 import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
4 import com.fasterxml.jackson.module.kotlin.readValue
5 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskApi
6 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskAuthException
7 import
8 com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.CustomPropertiesPatchDto
9 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.DiskInfoDto
10 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.EmbeddedResourceListDto
11 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.LinkDto
12 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.OperationStatusDto
13 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.ResourceDto
14 import kotlinx.coroutines.CoroutineDispatcher
15 import kotlinx.coroutines.delay
16 import kotlinx.coroutines.withContext
17 import okhttp3.MediaType.Companion.toMediaType
18 import okhttp3.Request
19 import okhttp3.RequestBody.Companion.asRequestBody
20 import okhttp3.RequestBody.Companion.toRequestBody
21 import okhttp3.ResponseBody
22 import retrofit2.HttpException
23 import retrofit2.Response
24 import java.io.FileNotFoundException
25 import java.io.FilterInputStream
26 import java.io.IOException
27 import java.io.InputStream
28 import java.util.concurrent.ConcurrentHashMap
29 import java.util.concurrent.atomic.AtomicLong
30
31 class YandexDiskRepository(
32     private val api: YandexDiskApi,
33     private val rawHttp: okhttp3.OkHttpClient,
34     private val ioDispatcher: CoroutineDispatcher,
35 ) {
36
37     private val diskCacheLock = Any()
38     private var diskInfoCached: DiskInfoDto? = null
39     private var diskInfoCachedUntilMs: Long = 0L
40
41     private val listCache = ConcurrentHashMap<ListCacheKey, ResourceDto>()
42     private val getCache = ConcurrentHashMap<String, ResourceDto>()
```

```

42     private val cloudApiCallCount = AtomicLong(0)
43
44     fun cloudApiCallCount(): Long = cloudApiCallCount.get()
45
46     fun resetCloudApiCallCount() {
47         cloudApiCallCount.set(0)
48     }
49
50     suspend fun diskInfo(): DiskInfoDto = withContext(ioDispatcher) {
51         val now = System.currentTimeMillis()
52         synchronized(diskCacheLock) {
53             val cached = diskInfoCached
54             if (cached != null && now < diskInfoCachedUntilMs) {
55                 return@withContext cached
56             }
57         }
58         val fresh = wrapAuth { api.getDisk() }
59         synchronized(diskCacheLock) {
60             diskInfoCached = fresh
61             diskInfoCachedUntilMs = System.currentTimeMillis() + DISK_INFO_TTL_MS
62         }
63         fresh
64     }
65
66     suspend fun list(path: String, limit: Int, offset: Int, sort: String? = null):
ResourceDto =
67         withContext(ioDispatcher) {
68             val key = ListCacheKey(path = path, limit = limit, offset = offset, sort = sort)
69             listCache[key]?.let { return@withContext it }
70
71             suspend fun tryList(p: String): ResourceDto? =
72                 try {
73                     wrapAuth { api.listResources(p, limit, offset, sort, FIELDS_LIST) }
74                 } catch (e: HttpException) {
75                     if (e.code() == 404) null else throw e
76                 }
77             val primary = tryList(path)
78             val secondary = if (!path.endsWith('/') && path != "app:/") tryList("$path/")
else null
79             val result = primary ?: secondary
80                 ?: ResourceDto(embedded = EmbeddedResourceListDto(items = emptyList()))
81             putListCache(key, result)
82             result
83         }
84

```

```

85     suspend fun get(path: String): ResourceDto = withContext(ioDispatcher) {
86         getCache[path]?.let { return@withContext it }
87         val result = fetchResource(path)
88         putGetCache(path, result)
89         result
90     }
91
92     suspend fun getOrNull(path: String): ResourceDto? = withContext(ioDispatcher) {
93         getCache[path]?.let { return@withContext it }
94         val result = fetchResourceOrNull(path)
95         if (result != null) {
96             putGetCache(path, result)
97         }
98         result
99     }
100
101     suspend fun createFolder(path: String): Unit = withContext(ioDispatcher) {
102         val resp = wrapAuth { api.createFolder(path) }
103         when (resp.code()) {
104             201, 409 -> invalidateDiskMetaCaches(path)
105             else -> throw failure("createFolder", resp)
106         }
107     }
108
109     suspend fun delete(path: String, permanently: Boolean = true): Unit =
withContext(ioDispatcher) {
110         val resp = wrapAuth { api.deleteResource(path, permanently) }
111         when (resp.code()) {
112             204 -> invalidateDiskMetaCaches(path)
113             202 -> {
114                 val link = resp.body()?.use { body -> parseLink(body) }
115                 ?: throw IOException("DELETE 202 without body")
116                 awaitOperation(link.href)
117                 invalidateDiskMetaCaches(path)
118             }
119             404 -> invalidateDiskMetaCaches(path)
120             else -> throw failure("delete", resp)
121         }
122     }
123
124     suspend fun setCustomProperties(path: String, props: Map<String, String>): Unit =
withContext(ioDispatcher) {
125         val resp = wrapAuth {
126             api.patchResource(path, CustomPropertiesPatchDto(props))
127         }
128     }

```

```

129         if (!resp.isSuccessful) {
130             throw failure("patch", resp)
131         }
132         resp.body()?.close()
133         invalidateDiskMetaCaches(path)
134     }
135
136     suspend fun uploadBytes(path: String, bytes: ByteArray, overwrite: Boolean = true): Unit
137     =
138         withContext(ioDispatcher) {
139             val link = uploadLinkOrThrow(path, overwrite)
140             require(link.method.equals("PUT", ignoreCase = true)) {
141                 "Unexpected upload method ${link.method}"
142             }
143             val body = bytes.toRequestBody(OCTET_STREAM)
144             val req = Request.Builder().url(link.href).put(body).build()
145             repeat(LOCKED_RETRY_MAX) { attempt ->
146                 recordCloudApiCall()
147                 rawHttp.newCall(req).execute().use { resp ->
148                     when {
149                         resp.isSuccessful -> {
150                             invalidateDiskMetaCaches(path)
151                             return@withContext
152                         }
153                         resp.code == 423 && attempt < LOCKED_RETRY_MAX - 1 ->
154                             delay(lockedBackoffMs(attempt))
155                         else ->
156                             throw IOException("Upload failed: HTTP ${resp.code}")
157                     }
158                 }
159             }
160
161     suspend fun uploadFile(path: String, file: java.io.File, overwrite: Boolean = true):
162     Unit =
163         withContext(ioDispatcher) {
164             val link = uploadLinkOrThrow(path, overwrite)
165             require(link.method.equals("PUT", ignoreCase = true)) {
166                 "Unexpected upload method ${link.method}"
167             }
168             val body = file.asRequestBody(OCTET_STREAM)
169             val req = Request.Builder().url(link.href).put(body).build()
170             repeat(LOCKED_RETRY_MAX) { attempt ->
171                 recordCloudApiCall()
172                 rawHttp.newCall(req).execute().use { resp ->
173                     when {

```



```

173         resp.isSuccessful -> {
174             invalidateDiskMetaCaches(path)
175             return@withContext
176         }
177         resp.code == 423 && attempt < LOCKED_RETRY_MAX - 1 ->
178             delay(lockedBackoffMs(attempt))
179         else ->
180             throw IOException("Upload failed: HTTP ${resp.code}")
181     }
182 }
183 }
184 }
185
186 /** Поток должен быть закрыт вызывающим кодом – закроет HTTP-ответ. */
187 suspend fun openDownloadStream(path: String): InputStream = withContext(ioDispatcher) {
188     val link = try {
189         wrapAuth { api.getDownloadLink(path) }
190     } catch (e: HttpException) {
191         if (e.code() == 404) throw FileNotFoundException(path)
192         throw e
193     }
194     require(link.method.equals("GET", ignoreCase = true)) {
195         "Unexpected download method ${link.method}"
196     }
197     val req = Request.Builder().url(link.href).get().build()
198     repeat(LOCKED_RETRY_MAX) { attempt ->
199         recordCloudApiCall()
200         val resp = rawHttp.newCall(req).execute()
201         when {
202             resp.isSuccessful -> {
203                 val body = resp.body
204                 val stream = body.byteStream()
205                 return@withContext object : FilterInputStream(stream) {
206                     override fun close() {
207                         `in`.use {
208                             // Response must be closed after the wrapped stream.
209                         }
210                         resp.close()
211                     }
212                 }
213             }
214             resp.code == 423 && attempt < LOCKED_RETRY_MAX - 1 -> {
215                 resp.close()
216                 delay(lockedBackoffMs(attempt))

```

```

217         }
218         else -> {
219             val code = resp.code
220             resp.close()
221             throw IOException("Download failed: HTTP $code")
222         }
223     }
224 }
225 throw IOException("Download failed: HTTP 423 after retries")
226 }
227
228 private suspend fun awaitOperation(href: String) {
229     repeat(OPERATION_POLL_MAX) {
230         delay(OPERATION_POLL_DELAY_MS)
231         val st: OperationStatusDto = try {
232             wrapAuth { api.getOperationByUrl(href) }
233         } catch (e: HttpException) {
234             if (e.code() == 404) {
235                 throw IOException("Disk async operation status not found (404)")
236             }
237             throw e
238         }
239         when (st.status?.lowercase()) {
240             "success" -> return
241             "failure", "failed" -> throw IOException("Disk async operation failed")
242             else -> { }
243         }
244     }
245     throw IOException("Disk async operation timed out")
246 }
247
248 private suspend fun uploadLinkOrThrow(path: String, overwrite: Boolean): LinkDto {
249     try {
250         return wrapAuth { api.getUploadLink(path, overwrite) }
251     } catch (e: HttpException) {
252         if (e.code() == 404) {
253             throw FileNotFoundException("Upload path or parent not found: $path")
254         }
255         throw e
256     }
257 }
258
259 private suspend fun fetchResource(path: String): ResourceDto {
260     suspend fun tryGet(p: String): ResourceDto? =
261         try {

```

```

262         wrapAuth { api.getResource(p, FIELDS_RESOURCE) }
263     } catch (e: HttpException) {
264         if (e.code() == 404) null else throw e
265     }
266     val primary = tryGet(path)
267     val secondary = if (!path.endsWith('/')) tryGet("$path/") else null
268     return primary ?: secondary ?: throw FileNotFoundException(path)
269 }
270
271 private suspend fun fetchResourceOrNull(path: String): ResourceDto? {
272     suspend fun tryGet(p: String): ResourceDto? =
273         try {
274             wrapAuth { api.getResource(p, FIELDS_RESOURCE) }
275         } catch (e: HttpException) {
276             if (e.code() == 404) null else throw e
277         }
278     return tryGet(path) ?: if (!path.endsWith('/')) tryGet("$path/") else null
279 }
280
281 private fun putListCache(key: ListCacheKey, value: ResourceDto) {
282     if (listCache.size >= LIST_CACHE_MAX_ENTRIES) {
283         listCache.clear()
284     }
285     listCache[key] = value
286 }
287
288 private fun putGetCache(path: String, value: ResourceDto) {
289     if (getCache.size >= GET_CACHE_MAX_ENTRIES) {
290         getCache.clear()
291     }
292     getCache[path] = value
293 }
294
295 private fun invalidateDiskMetaCaches(changedDiskPath: String? = null) {
296     synchronized(diskCacheLock) {
297         diskInfoCached = null
298         diskInfoCachedUntilMs = 0L
299     }
300     if (changedDiskPath == null) {
301         listCache.clear()
302         getCache.clear()
303         return
304     }
305     val prefixes = cachePrefixesForPath(changedDiskPath)
306     listCache.keys.removeAll { key ->

```

```

307         prefixes.any { prefix ->
308             key.path.startsWith(prefix) || prefix.startsWith(key.path.trimEnd('/'))
309         }
310     }
311     getCache.keys.removeAll { cachedPath ->
312         prefixes.any { prefix ->
313             cachedPath.startsWith(prefix) || prefix.startsWith(cachedPath.trimEnd('/'))
314         }
315     }
316 }
317
318 private fun cachePrefixesForPath(diskPath: String): List<String> {
319     val normalized = diskPath.trimEnd('/')
320     val out = mutableListOf<String>()
321     var current = normalized
322     while (current.isNotEmpty()) {
323         out.add(current)
324         out.add("$current/")
325         val slash = current.lastIndexOf('/')
326         if (slash <= 0) break
327         current = current.substring(0, slash)
328     }
329     return out
330 }
331
332 private suspend inline fun <T> wrapAuth(crossinline block: suspend () -> T): T {
333     repeat(LOCKED_RETRY_MAX) { attempt ->
334         try {
335             recordCloudApiCall()
336             return block()
337         } catch (e: HttpException) {
338             when (e.code()) {
339                 401 -> throw YandexDiskAuthException(e.message())
340                 423 -> {
341                     if (attempt >= LOCKED_RETRY_MAX - 1) {
342                         throw IOException(
343                             "Yandex Disk: ресурс временно заблокирован (HTTP 423).
Повторите позже.",
344                             e,
345                         )
346                     }
347                     delay(lockedBackoffMs(attempt))
348                 }
349                 else -> throw e
350             }
351         }
352     }
353 }

```

```

351         }
352     }
353     error("unreachable")
354 }
355
356 private fun recordCloudApiCall() {
357     cloudApiCallCount.incrementAndGet()
358 }
359
360 private fun failure(op: String, resp: Response<ResponseBody>): IOException {
361     val msg = resp.errorBody()?.string() ?: resp.message()
362     return IOException("$op failed: HTTP ${resp.code()} $msg")
363 }
364
365 private fun parseLink(body: ResponseBody): LinkDto =
366     jackson.readValue(body.string())
367
368 private data class ListCacheKey(
369     val path: String,
370     val limit: Int,
371     val offset: Int,
372     val sort: String?,
373 )
374
375 companion object {
376     private val jackson = jacksonObjectMapper().apply { findAndRegisterModules() }
377     private val OCTET_STREAM = "application/octet-stream".toMediaType()
378     private const val OPERATION_POLL_DELAY_MS = 300L
379     private const val OPERATION_POLL_MAX = 200
380
381     private const val DISK_INFO_TTL_MS = 45_000L
382     private const val LIST_CACHE_MAX_ENTRIES = 384
383     private const val GET_CACHE_MAX_ENTRIES = 384
384
385     /** Сколько раз повторять запрос при HTTP 423 (ресурс временно заблокирован). */
386     private const val LOCKED_RETRY_MAX = 6
387
388     private fun lockedBackoffMs(attempt: Int): Long =
389         (100L * (1L shl attempt)).coerceAtMost(2000L)
390
391     /**
392      * Урезанный набор полей для листинга каталога (см. параметр `fields` в Disk API).
393      */
394     private const val FIELDS_LIST =
395         "path,type,name,size,modified,created,custom_properties," +

```

```
396         "_embedded.items.path,_embedded.items.type,_embedded.items.name," +
397         "_embedded.items.size,_embedded.items.modified,_embedded.items.created," +
398         "_embedded.items.custom_properties"
399
400     private const val FIELDS_RESOURCE =
401         "path,type,name,size,modified,created,custom_properties"
402     }
403 }
404
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/repository/YandexDiskRepositoryFactory.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository
2
3 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskApiFactory
4 import kotlinx.coroutines.CoroutineDispatcher
5 import java.util.UUID
6
7 class YandexDiskRepositoryFactory(
8     private val apiFactory: YandexDiskApiFactory,
9     private val ioDispatcher: CoroutineDispatcher,
10 ) {
11     fun create(vaultUuid: UUID): YandexDiskRepository =
12         YandexDiskRepository(
13             api = apiFactory.createApiForVault(vaultUuid),
14             rawHttp = apiFactory.rawHttpClient,
15             ioDispatcher = ioDispatcher,
16         )
17 }
18
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexuserinfo/YandexUserInfoApi.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo
2
3 import retrofit2.http.GET
4 import retrofit2.http.Header
5 import retrofit2.http.Query
6
7 interface YandexUserInfoApi {
8     @GET("info")
9     suspend fun userInfo(
10         @Query("format") format: String,
11         @Header("Authorization") authorization: String,
12     ): YandexUserInfoDto
13 }
14
```


Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexuserinfo/YandexUserInfoApiFactory.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo
2
3 import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
4 import retrofit2.Retrofit
5 import retrofit2.converter.jackson.JacksonConverterFactory
6
7 object YandexUserInfoApiFactory {
8     fun create(): YandexUserInfoApi {
9         val retrofit = Retrofit.Builder()
10             .baseUrl("https://login.yandex.ru/")
11             .addConverterFactory(
12 JacksonConverterFactory.create(jacksonObjectMapper().findAndRegisterModules()),
13             )
14             .build()
15         return retrofit.create(YandexUserInfoApi::class.java)
16     }
17 }
18
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexuserinfo/YandexUserInfoDto.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo
2
3 import com.fasterxml.jackson.annotation.JsonIgnoreProperties
4 import com.fasterxml.jackson.annotation.JsonProperty
5
6 @JsonIgnoreProperties(ignoreUnknown = true)
7 data class YandexUserInfoDto(
8     val id: String,
9     val login: String,
10    @param:JsonProperty("default_email") val defaultEmail: String? = null,
11 )
12
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexuserinfo/repository/YandexUserInfoRepository.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo.repository
2
3 import com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo.YandexUserInfoApi
4 import com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo.YandexUserInfoDto
5 import kotlinx.coroutines.CoroutineDispatcher
6 import kotlinx.coroutines.withContext
7
8 class YandexUserInfoRepository(
9     private val api: YandexUserInfoApi,
10    private val ioDispatcher: CoroutineDispatcher
11 ) {
12     suspend fun userInfo(accessToken: String): YandexUserInfoDto = withContext(ioDispatcher)
13     {
14         api.userInfo(format = "json", authorization = "OAuth $accessToken")
15     }
16 }
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/ports/StorageKeyMapStore.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.ports
2
3 import com.github.nullptroma.wallenc.domain.vault.model.StorageKeyMap
4
5
6 interface StorageKeyMapStore {
7     suspend fun add(value: StorageKeyMap)
8     suspend fun getAll(): List<StorageKeyMap>
9     suspend fun delete(vararg values: StorageKeyMap)
10 }
11
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/ports/YandexAccountStore.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.ports
2
3 import com.github.nullptroma.wallenc.domain.vault.model.YandexAccount
4 import kotlinx.coroutines.flow.Flow
5
6 interface YandexAccountStore {
7     fun observeAll(): Flow<List<YandexAccount>>
8     suspend fun getByYandexUserId(id: String): YandexAccount?
9     suspend fun getByVaultUuid(vaultUuid: String): YandexAccount?
10    suspend fun insert(account: YandexAccount)
11    suspend fun updateCredentials(vaultUuid: String, email: String, token: String)
12    suspend fun deleteByVaultUuid(vaultUuid: String)
13 }
14
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/UnlockManager.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.domain.vault.model.StorageKeyMap
5 import com.github.nullptroma.wallenc.domain.vault.ports.StorageKeyMapStore
6 import com.github.nullptroma.wallenc.domain.vault.storages.encrypt.EncryptedStorage
7 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
8 import com.github.nullptroma.wallenc.domain.encrypt.Encryptor
9 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
10 import com.github.nullptroma.wallenc.domain.interfaces.IUnlockManager
11 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
12 import kotlinx.coroutines.CoroutineDispatcher
13 import kotlinx.coroutines.CoroutineScope
14 import kotlinx.coroutines.flow.MutableStateFlow
15 import kotlinx.coroutines.flow.StateFlow
16 import kotlinx.coroutines.launch
17 import kotlinx.coroutines.sync.Mutex
18 import kotlinx.coroutines.sync.withLock
19 import kotlinx.coroutines.withContext
20 import java.nio.ByteBuffer
21 import java.security.MessageDigest
22 import java.util.UUID
23
24 class UnlockManager(
25     private val keymapRepository: StorageKeyMapStore,
26     private val ioDispatcher: CoroutineDispatcher,
27     vaultsManager: IVaultsManager
28 ) : IUnlockManager {
29     private val _openedStorages = MutableStateFlow<Map<UUID, EncryptedStorage>>(emptyMap())
30     override val openedStorages: StateFlow<Map<UUID, IStorage>>
31         get() = _openedStorages
32     private val mutex = Mutex()
33
34     override fun getOpenedStorageKey(uuid: UUID): EncryptKey? {
35         val opened = _openedStorages.value
36         val direct = opened[uuid]
37         return direct?.getKey()
38     }
39
40     init {
41         CoroutineScope(ioDispatcher).launch {
42             vaultsManager.allStorages.collect {
```

```

43         mutex.withLock {
44             val allKeys = keymapRepository.getAll()
45             val keysToRemove = mutableList0f<StorageKeyMap>()
46             val allStorages = it.toMutableList()
47             val map = _openedStorages.value.toMutableMap()
48             while(allStorages.isNotEmpty()) {
49                 val storage = allStorages[allStorages.size-1]
50                 val key = allKeys.find { key -> key.sourceUuid == storage.uuid }
51                 if(key == null) {
52                     allStorages.removeAt(allStorages.size - 1)
53                     continue
54                 }
55                 try {
56                     val encStorage = createEncryptedStorage(storage, key.key,
getKeyDestUuid(storage.uuid))
57                     map[storage.uuid] = encStorage
58                     allStorages.removeAt(allStorages.size - 1)
59                     allStorages.add(encStorage)
60                 }
61                 catch (e: WallencException.Storage.IncorrectKey) {
62                     keysToRemove.add(key)
63                     allStorages.removeAt(allStorages.size - 1)
64                 }
65                 catch (_: WallencException.Storage.EncInfoMissing) {
66                     keysToRemove.add(key)
67                     allStorages.removeAt(allStorages.size - 1)
68                 }
69                 catch (_: Exception) {
70                     allStorages.removeAt(allStorages.size - 1)
71                 }
72             }
73             keymapRepository.delete(*keysToRemove.toTypedArray()) // удалить мёртвые
ключи
74             _openedStorages.value = map.toMap()
75         }
76     }
77 }
78 }
79
80 private suspend fun createEncryptedStorage(storage: IStorage, key: EncryptKey, uuid:
UUID): EncryptedStorage {
81     return EncryptedStorage.create(
82         source = storage,
83         key = key,
84         ioDispatcher = ioDispatcher,

```

```

85         uuid = uuid
86     )
87 }
88
89 private fun getDestUuid(sourceUuid: UUID): UUID {
90     return uuid5(
91         namespace = UUID.fromString("6ba7b811-9dad-11d1-80b4-00c04fd430c8"), // URL
namespace
92         name = "$sourceUuid:open"
93     )
94 }
95
96 private fun uuid5(namespace: UUID, name: String): UUID {
97     val digest = MessageDigest.getInstance("SHA-1")
98     val nsBytes = ByteBuffer.allocate(16)
99         .putLong(namespace.mostSignificantBits)
100        .putLong(namespace.leastSignificantBits)
101        .array()
102    digest.update(nsBytes)
103    digest.update(name.toByteArray(Charsets.UTF_8))
104    val hash = digest.digest()
105
106    hash[6] = (hash[6].toInt() and 0x0f or 0x50).toByte() // version 5
107    hash[8] = (hash[8].toInt() and 0x3f or 0x80).toByte() // RFC 4122 variant
108
109    val bb = ByteBuffer.wrap(hash, 0, 16)
110    return UUID(bb.long, bb.long)
111 }
112
113 override suspend fun rememberKey(storage: IStorage, key: EncryptKey) =
withContext(ioDispatcher) {
114     mutex.withLock {
115         val encInfo = storage.metaInfo.value.encInfo ?: throw
WallencException.Storage.EncInfoMissing()
116         if (!Encryptor.checkKey(key, encInfo)) {
117             throw WallencException.Storage.IncorrectKey()
118         }
119         keymapRepository.add(
120             StorageKeyMap(
121                 sourceUuid = storage.uuid,
122                 key = key,
123             ),
124         )
125     }
126 }
127

```



```

128         override suspend fun open(
129             storage: IStorage,
130             key: EncryptKey,
131             rememberPassword: Boolean
132         ): EncryptedStorage = withContext(ioDispatcher) {
133             return@withContext mutex.withLock {
134                 val encInfo = storage.metaInfo.value.encInfo ?: throw
WallencException.Storage.EncInfoMissing()
135                 if (!Encryptor.checkKey(key, encInfo))
136                     throw WallencException.Storage.IncorrectKey()
137
138                 val opened = _openedStorages.value.toMutableMap()
139                 val cur = opened[storage.uuid]
140                 if (cur != null)
141                     return@withLock cur
142
143                 val keymap = StorageKeyMap(
144                     sourceUuid = storage.uuid,
145                     key = key
146                 )
147                 val encStorage = createEncryptedStorage(storage, keymap.key,
getDestUuid(storage.uuid))
148                 opened[storage.uuid] = encStorage
149                 _openedStorages.value = opened
150                 if (rememberPassword) {
151                     keymapRepository.add(keymap)
152                 }
153                 encStorage
154             }
155         }
156
157         /**
158          * Закрыть шифрование хранилища, закрывает рекурсивно, удаляя все ключи
159          * @param uuid uuid исходного хранилища
160          */
161         override suspend fun close(uuid: UUID): Unit = withContext(ioDispatcher) {
162             mutex.withLock {
163                 val opened = _openedStorages.value.toMutableMap()
164                 closeBySourceUuid(opened, uuid)
165                 _openedStorages.value = opened
166             }
167         }
168
169         // Закрытие только по source-экземпляру.
170         override suspend fun close(storage: IStorage) {
171             val opened = _openedStorages.value

```

```

172         if (opened.containsKey(storage.uuid)) {
173             close(storage.uuid)
174         }
175     }
176
177     private fun closeBySourceUuid(opened: MutableMap<UUID, EncryptedStorage>, sourceUuid:
178     UUID) {
179         val enc = opened[sourceUuid] ?: return
180         val nestedSourceUuid = enc.uuid
181         if (nestedSourceUuid != sourceUuid && opened.containsKey(nestedSourceUuid)) {
182             closeBySourceUuid(opened, nestedSourceUuid)
183         }
184         opened.remove(sourceUuid)
185         enc.dispose()
186     }

```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/common/BaseStorage.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.common
2
3 import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
4 import com.github.nullptroma.wallenc.domain.common.impl.CommonStorageMetaInfo
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
6 import com.github.nullptroma.wallenc.domain.datatypes.StorageMetaLoadState
7 import com.github.nullptroma.wallenc.domain.errors.WallencException
8 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
9 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
10 import com.github.nullptroma.wallenc.domain.interfaces.IStorageMetaInfo
11 import com.github.nullptroma.wallenc.domain.tasks.TaskProgress
12 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
13 import kotlinx.coroutines.CoroutineDispatcher
14 import kotlinx.coroutines.ensureActive
15 import kotlinx.coroutines.flow.Flow
16 import kotlinx.coroutines.flow.MutableStateFlow
17 import kotlinx.coroutines.flow.StateFlow
18 import kotlinx.coroutines.flow.map
19 import kotlinx.coroutines.withContext
20 import java.io.InputStream
21 import java.util.UUID
22
23 /**
24  * Общий «скелет» для [IStorage]: единая логика meta-info, rename, setEncInfo,
25  * clearAllContent и делегирование размеров и доступности в [accessor].
26  *
27  * Подклассы определяют только способ создания [accessor], значение
28  * [isVirtualStorage] и (при необходимости) расширяют [init] своими шагами
29  * (например, проверкой ключа или инициализацией внешней связи).
30  */
31 abstract class BaseStorage(
32     override val uuid: UUID,
33     private val ioDispatcher: CoroutineDispatcher,
34     metaInfoFilePostfix: String,
35 ) : IStorage {
36
37     protected val metaInfoFileName: String = "${metaInfoUuidPart()}$metaInfoFilePostfix"
38
39     private val _metaInfo = MutableStateFlow<IStorageMetaInfo>(CommonStorageMetaInfo())
40     final override val metaInfo: StateFlow<IStorageMetaInfo>
41         get() = _metaInfo
42 }
```

```

43     private val _metaLoadState = MutableStateFlow(StorageMetaLoadState.Loading)
44     final override val metaLoadState: StateFlow<StorageMetaLoadState>
45         get() = _metaLoadState
46
47     final override val size: StateFlow<Long?>
48         get() = accessor.size
49
50     final override val numberOfFiles: StateFlow<Int?>
51         get() = accessor.numberOfFiles
52
53     final override val isAvailable: StateFlow<Boolean>
54         get() = accessor.isAvailable
55
56     final override val isEmpty: Flow<Boolean?>
57         get() = accessor.numberOfFiles.map { n -> n?.let { it == 0 } }
58
59     abstract override val accessor: IStorageAccessor
60
61     /**
62      * Базовая реализация [IStorageAccessor] передаёт UUID полностью; подклассы
63      * могут переопределить, чтобы сохранить совместимость с уже существующими
64      * именами файлов (например,
65      * [com.github.nullptroma.wallenc.domain.vault.storages.encrypt.EncryptedStorage]
66      * раньше использовал первые 8 символов).
67      */
68     protected open fun metaInfoUuidPart(): String = uuid.toString()
69
70     /**
71      * Запускается единожды при первом использовании хранилища. Подклассы могут
72      * переопределить,
73      * * добавив свои шаги (инициализацию [accessor], проверку ключа и т.п.). Обязательно
74      * * должен в какой-то момент вызвать [readMetaInfo].
75      */
76     open suspend fun init() {
77         readMetaInfo()
78     }
79
80     private suspend fun readMetaInfo() = withContext(ioDispatcher) {
81         val (meta, state) = loadMetaFromDisk()
82         _metaInfo.value = meta
83         _metaLoadState.value = state
84     }
85
86     private suspend fun loadMetaFromDisk(): Pair<IStorageMetaInfo, StorageMetaLoadState> {
87         return try {

```

```

86         val bytes = accessor.openReadSystemFile(metaInfoFileName).use { it.readBytes() }
87     when {
88         bytes.isEmpty() -> {
89             val default = CommonStorageMetaInfo()
90             updateMetaInfo(default)
91             default to StorageMetaLoadState.Ready
92         }
93         else -> try {
94             jackson.readValue(bytes, CommonStorageMetaInfo::class.java) to
StorageMetaLoadState.Ready
95         } catch (_: Exception) {
96             CommonStorageMetaInfo() to StorageMetaLoadState.Unavailable
97         }
98     }
99     } catch (_: WallencException.Storage.FileNotFound) {
100         val default = CommonStorageMetaInfo()
101         updateMetaInfo(default)
102         default to StorageMetaLoadState.Ready
103     } catch (_: Exception) {
104         CommonStorageMetaInfo() to StorageMetaLoadState.Unavailable
105     }
106 }
107
108 private suspend fun requireMetaReady() {
109     if (_metaLoadState.value != StorageMetaLoadState.Ready) {
110         throw WallencException.Storage.NotAvailable()
111     }
112 }
113
114 private suspend fun updateMetaInfo(meta: IStorageMetaInfo) = withContext(ioDispatcher) {
115     val writer = accessor.openWriteSystemFile(metaInfoFileName)
116     writer.use { writer ->
117         jackson.writeValue(writer, meta)
118     }
119     _metaInfo.value = meta
120 }
121
122 final override suspend fun rename(newName: String) = withContext(ioDispatcher) {
123     requireMetaReady()
124     val cur = metaInfo.value
125     updateMetaInfo(
126         CommonStorageMetaInfo(
127             encInfo = cur.encInfo,
128             name = newName,
129             ),

```

```

130         )
131     }
132
133     final override suspend fun setEncInfo(encInfo: StorageEncryptionInfo?) =
withContext(ioDispatcher) {
134         requireMetaReady()
135         val cur = metaInfo.value
136         updateMetaInfo(
137             CommonStorageMetaInfo(
138                 encInfo = encInfo,
139                 name = cur.name,
140             ),
141         )
142     }
143
144     final override suspend fun clearAllContent(onProgress: suspend (TaskProgress) -> Unit) =
withContext(ioDispatcher) {
145         val files = accessor.getAllFiles()
146         val dirs = accessor.getAllDirs()
147         val paths = buildList {
148             addAll(files.map { it.metaInfo.path })
149             addAll(dirs.map { it.metaInfo.path })
150         }
151         .filter { it != "/" && it.isNotBlank() }
152         .sortedByDescending { it.length }
153         val total = paths.size
154         if (total == 0) {
155             return@withContext
156         }
157         paths.forEachIndexed { index, path ->
158             accessor.delete(path)
159             if (index % PROGRESS_REPORT_INTERVAL == 0 || index == paths.lastIndex) {
160                 val done = index + 1
161                 onProgress(
162                     TaskProgress(
163                         fraction = done.toFloat() / total,
164                         label = TaskProgressLabel.ClearContentProgress(done, total),
165                     ),
166                 )
167                 coroutineContext.ensureActive()
168             }
169         }
170     }
171
172     companion object {
173         private const val PROGRESS_REPORT_INTERVAL = 16

```

```
174         private val jackson = jacksonObjectMapper().apply { findAndRegisterModules() }
175     }
176 }
177
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/common/StorageSyncJournalBuffer.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.common
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournal
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalMerge
6 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncRevision
7 import kotlinx.coroutines.sync.Mutex
8 import kotlinx.coroutines.sync.withLock
9 import java.util.UUID
10
11 /**
12  * Буфер журнала sync: накопление записей и безопасный flush (без потери pending при ошибке
13  * записи).
14  */
15 class StorageSyncJournalBuffer(
16     private val syncActorId: String,
17     private val originStorageUuid: UUID?,
18     private val readJournal: suspend () -> StorageSyncJournal,
19     private val writeJournal: suspend (StorageSyncJournal) -> Unit,
20 ) {
21     private val pendingMutex = Mutex()
22     private val flushMutex = Mutex()
23     private var pendingJournalEntries: StorageSyncJournal = emptyMap()
24
25     @Volatile
26     private var sequenceHighWatermark: Long? = null
27
28     suspend fun flushPending() {
29         flushMutex.withLock {
30             flushPendingUnderLock()
31         }
32     }
33
34     suspend fun putEntries(entries: StorageSyncJournal) {
35         flushPending()
36         if (entries.isEmpty()) {
37             return
38         }
39         val current = readJournal()
40         val merged = StorageSyncJournalMerge.merge(current, entries)
41         if (merged == current) {
42             return
43         }
44         writeJournal(merged)
45     }
46 }
```



```

42         }
43         writeJournal(merged)
44         refreshSequenceHighWatermark(merged)
45     }
46
47     suspend fun appendEntry(path: String, entry: StorageSyncJournalEntry) {
48         pendingMutex.withLock {
49             pendingJournalEntries = pendingJournalEntries + (path to entry)
50         }
51         flushPending()
52     }
53
54     suspend fun nextSequence(): Long {
55         flushPending()
56         val diskMax = readJournal().values.maxOfOrNull { it.revision.sequence } ?: 0L
57         val pendingMax = pendingMutex.withLock {
58             pendingJournalEntries.values.maxOfOrNull { it.revision.sequence } ?: 0L
59         }
60         val base = maxOf(diskMax, pendingMax, sequenceHighWatermark ?: 0L)
61         val next = base + 1L
62         sequenceHighWatermark = next
63         return next
64     }
65
66     fun buildEntry(
67         path: String,
68         operation: com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation,
69         sequence: Long,
70     ): StorageSyncJournalEntry {
71         return StorageSyncJournalEntry(
72             path = path,
73             operation = operation,
74             revision = StorageSyncRevision(
75                 sequence = sequence,
76                 actorId = syncActorId,
77                 createdAt = java.time.Instant.now(),
78             ),
79             originStorageUuid = originStorageUuid,
80         )
81     }
82
83     private suspend fun flushPendingUnderLock() {
84         val pending = pendingMutex.withLock {
85             if (pendingJournalEntries.isEmpty()) {
86                 return

```

```

87         }
88         val snapshot = pendingJournalEntries
89         pendingJournalEntries = emptyMap()
90         snapshot
91     }
92     try {
93         val current = readJournal()
94         val merged = StorageSyncJournalMerge.merge(current, pending)
95         if (merged != current) {
96             writeJournal(merged)
97         }
98         refreshSequenceHighWatermark(merged)
99     } catch (e: Exception) {
100         pendingMutex.withLock {
101             pendingJournalEntries = StorageSyncJournalMerge.merge(pending,
102             pendingJournalEntries)
103             }
104             throw e
105         }
106     }
107     private fun refreshSequenceHighWatermark(journal: StorageSyncJournal) {
108         val maxSeq = journal.values.maxOfOrNull { it.revision.sequence } ?: return
109         val current = sequenceHighWatermark
110         sequenceHighWatermark = if (current == null) maxSeq else maxOf(current, maxSeq)
111     }
112 }
113

```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/common/StorageSyncJournalCodec.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.common
2
3 import com.fasterxml.jackson.databind.ObjectMapper
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournal
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
6 import java.util.LinkedHashMap
7
8 internal object StorageSyncJournalCodec {
9     private val journalJavaType = ObjectMapper().typeFactory.constructMapType(
10         LinkedHashMap::class.java,
11         String::class.java,
12         StorageSyncJournalEntry::class.java,
13     )
14
15     fun read(mapper: ObjectMapper, bytes: ByteArray): StorageSyncJournal {
16         if (bytes.isEmpty()) {
17             return emptyMap()
18         }
19         return runCatching {
20             @Suppress("UNCHECKED_CAST")
21             mapper.readValue(bytes, journalJavaType) as StorageSyncJournal
22         }.getOrElse {
23             emptyMap<String, StorageSyncJournalEntry>()
24         }
25     }
26
27     fun write(mapper: ObjectMapper, journal: StorageSyncJournal): ByteArray =
28         mapper.writeValueAsBytes(journal)
29 }
30
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/common/SystemFileRead.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.common
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import java.io.InputStream
5
6 /** Читает системный файл; отсутствие файла – пустой массив байт (не исключение). */
7 internal suspend fun readSystemFileBytesOrEmpty(open: suspend () -> InputStream): ByteArray
8 =
9     try {
10         open().use { it.readBytes() }
11     } catch (_: WallencException.Storage.FileNotFound) {
12         ByteArray(0)
13     }
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/encrypt/EncryptedStorage.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.encrypt
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4
5 import com.github.nullptroma.wallenc.domain.vault.storages.common.BaseStorage
6 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
7 import com.github.nullptroma.wallenc.domain.encrypt.Encryptor
8 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
9 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
10 import kotlinx.coroutines.CoroutineDispatcher
11 import kotlinx.coroutines.CoroutineExceptionHandler
12 import kotlinx.coroutines.CoroutineScope
13 import kotlinx.coroutines.DisposableHandle
14 import kotlinx.coroutines.SupervisorJob
15 import kotlinx.coroutines.withContext
16 import java.util.UUID
17
18 class EncryptedStorage private constructor(
19     private val source: IStorage,
20     private val key: EncryptKey,
21     ioDispatcher: CoroutineDispatcher,
22     uuid: UUID = UUID.randomUUID()
23 ) : BaseStorage(
24     uuid = uuid,
25     ioDispatcher = ioDispatcher,
26     metaInfoFilePostfix = STORAGE_INFO_FILE_POSTFIX,
27 ), DisposableHandle {
28
29     private val job = SupervisorJob()
30     private val scope = CoroutineScope(
31         ioDispatcher + job + CoroutineExceptionHandler { _, throwable ->
32             System.err.println("EncryptedStorage: uncaught coroutine failure:
33             ${throwable.message}")
34             throwable.printStackTrace()
35         },
36     )
37
38     private val encInfo =
39         source.metaInfo.value.encInfo ?: throw WallencException.Storage.NotEncrypted()
40
41     override val isVirtualStorage: Boolean = true
```

```

42     private val _accessor: EncryptedStorageAccessor =
43         EncryptedStorageAccessor(
44             source = source.accessor,
45             pathIv = encInfo.pathIv,
46             key = key,
47             systemHiddenDirName =
48                 "${uuid.toString().take(8)}$SYSTEM_HIDDEN_DIRNAME_POSTFIX",
49             scope = scope,
50         )
51     override val accessor: IStorageAccessor = _accessor
52
53     override fun metaInfoUuidPart(): String = uuid.toString().take(8)
54
55     override suspend fun init() {
56         checkKey()
57         super.init()
58     }
59
60     private fun checkKey() {
61         if (!Encryptor.checkKey(key, encInfo))
62             throw WallencException.Storage.IncorrectKey()
63     }
64
65     fun getKey(): EncryptKey = EncryptKey(key.bytes)
66
67     override fun dispose() {
68         _accessor.dispose()
69         job.cancel()
70     }
71
72     companion object {
73         suspend fun create(
74             source: IStorage,
75             key: EncryptKey,
76             ioDispatcher: CoroutineDispatcher,
77             uuid: UUID = UUID.randomUUID()
78         ): EncryptedStorage = withContext(ioDispatcher) {
79             val storage = EncryptedStorage(
80                 source = source,
81                 key = key,
82                 ioDispatcher = ioDispatcher,
83                 uuid = uuid
84             )
85             try {
86                 storage.init()
87             }
88         }
89     }

```

```
86         } catch (e: Exception) {
87             storage.dispose()
88             throw e
89         }
90         return@withContext storage
91     }
92
93     private const val SYSTEM_HIDDEN_DIRNAME_POSTFIX = "-enc-dir"
94     const val STORAGE_INFO_FILE_POSTFIX = ".enc-meta"
95 }
96 }
97
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/encrypt/EncryptedStorageAccessor.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.encrypt
2
3 import com.fasterxml.jackson.module.kotlin.readValue
4 import com.github.nullptroma.wallenc.domain.errors.WallencException
5 import com.github.nullptroma.wallenc.domain.vault.storages.common.StorageSyncJournalBuffer
6 import com.github.nullptroma.wallenc.domain.vault.storages.common.StorageSyncJournalCodec
7 import com.github.nullptroma.wallenc.domain.vault.storages.common.readSystemFileBytesOrEmpty
8 import
9 com.github.nullptroma.wallenc.domain.vault.utils.CloseHandledStreamExtension.Companion.onClose
10 import com.github.nullptroma.wallenc.domain.vault.utils.CloseHandledStreamExtension.Companion.onClose
11 import com.github.nullptroma.wallenc.domain.common.impl.CommonDirectory
12 import com.github.nullptroma.wallenc.domain.common.impl.CommonFile
13 import com.github.nullptroma.wallenc.domain.common.impl.CommonMetaInfo
14 import com.github.nullptroma.wallenc.domain.datatypes.DataPage
15 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
16 import com.github.nullptroma.wallenc.domain.encrypt.Encryptor
17 import com.github.nullptroma.wallenc.domain.encrypt.EncryptorWithStaticIv
18 import com.github.nullptroma.wallenc.domain.interfaces.IDirectory
19 import com.github.nullptroma.wallenc.domain.interfaces.IFile
20 import com.github.nullptroma.wallenc.domain.interfaces.IMetaInfo
21 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
22 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournal
23 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths
24 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncLock
25 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation
26 import kotlinx.coroutines.CoroutineScope
27 import kotlinx.coroutines.DisposableHandle
28 import kotlinx.coroutines.runBlocking
29 import kotlinx.coroutines.flow.Flow
30 import kotlinx.coroutines.flow.MutableSharedFlow
31 import kotlinx.coroutines.flow.MutableStateFlow
32 import kotlinx.coroutines.flow.SharedFlow
33 import kotlinx.coroutines.flow.StateFlow
34 import kotlinx.coroutines.flow.map
35 import kotlinx.coroutines.launch
36 import kotlinx.coroutines.sync.Mutex
37 import kotlinx.coroutines.sync.withLock
38 import java.io.ByteArrayInputStream
39 import java.io.InputStream
40 import java.io.OutputStream
41 import java.time.Instant
```



```

41 import java.util.UUID
42 import kotlin.io.path.Path
43 import kotlin.io.path.pathString
44
45 class EncryptedStorageAccessor(
46     private val source: IStorageAccessor,
47     pathIv: ByteArray?,
48     key: EncryptKey,
49     private val systemHiddenDirName: String,
50     private val scope: CoroutineScope
51 ) : IStorageAccessor, DisposableHandle {
52     private val syncActorId = UUID.randomUUID().toString()
53     private val syncLockMutex = Mutex()
54     private val journalBuffer = StorageSyncJournalBuffer(
55         syncActorId = syncActorId,
56         originStorageUuid = null,
57         readJournal = { readSyncJournalUnchecked() },
58         writeJournal = { writeSyncJournal(it) },
59     )
60     private val _size = MutableStateFlow<Long?>(null)
61     override val size: StateFlow<Long?> = _size
62
63     private val _numberOfFiles = MutableStateFlow<Int?>(null)
64     override val numberOfFiles: StateFlow<Int?> = _numberOfFiles
65
66     override val isAvailable: StateFlow<Boolean> = source.isAvailable
67
68     private val _filesUpdates = MutableSharedFlow<DataPage<IFile>>()
69     override val filesUpdates: SharedFlow<DataPage<IFile>> = _filesUpdates
70
71     private val _dirsUpdates = MutableSharedFlow<DataPage<IDirectory>>()
72     override val dirsUpdates: SharedFlow<DataPage<IDirectory>> = _dirsUpdates
73
74     private val dataEncryptor = Encryptor(key.toAesKey())
75     private val pathEncryptor: EncryptorWithStaticIv? = if(pathIv != null)
EncryptorWithStaticIv(key.toAesKey(), pathIv) else null
76
77     private var systemHiddenFilesIsActual = false
78
79     init {
80         collectSourceState()
81     }
82     private fun collectSourceState() {
83         scope.launch {
84             launch {
85                 try {

```

```

86         source.filesUpdates.collect { page ->
87             val files = page.data.map(::decryptEntity).filterSystemHiddenFiles()
88             _filesUpdates.emit(
89                 DataPage(
90                     list = files,
91                     isLoading = page.isLoading,
92                     isError = page.isError,
93                     hasNext = page.hasNext,
94                     pageLength = page.pageLength,
95                     pageIndex = page.pageIndex,
96                 ),
97             )
98         }
99     } catch (_: Exception) {
100     }
101 }
102
103 launch {
104     try {
105         source.dirsUpdates.collect { page ->
106             val dirs = page.data.map(::decryptEntity).filterSystemHiddenDirs()
107             _dirsUpdates.emit(
108                 DataPage(
109                     list = dirs,
110                     isLoading = page.isLoading,
111                     isError = page.isError,
112                     hasNext = page.hasNext,
113                     pageLength = page.pageLength,
114                     pageIndex = page.pageIndex,
115                 ),
116             )
117         }
118     } catch (_: Exception) {
119     }
120 }
121
122 launch {
123     source.numberOfFiles.collect {
124         if (it == null) {
125             _numberOfFiles.value = null
126         } else {
127             val hiddenCount = runCatching
128             { getSystemFiles().size }.getOrNull() ?: return@collect
129             _numberOfFiles.value = it - hiddenCount
130         }
131     }
132 }

```

```

130         }
131     }
132
133     launch {
134         source.size.collect { sourceSize ->
135             if (sourceSize == null) {
136                 _size.value = null
137             } else {
138                 val hiddenBytes = runCatching {
139                     getSystemFiles().sumOf { file -> file.metaInfo.size }
140                 }.getOrNull() ?: return@collect
141                 _size.value = sourceSize - hiddenBytes
142             }
143         }
144     }
145 }
146
147
148 private suspend fun getSystemFiles(): List<IFile> {
149     return source.GetFiles(encryptPath(systemHiddenDirName))
150 }
151
152 private fun decryptEntity(file: IFile): IFile {
153     return CommonFile(decryptMeta(file.metaInfo))
154 }
155
156 private fun decryptEntity(dir: IDirectory): IDirectory {
157     return CommonDirectory(decryptMeta(dir.metaInfo), dir.elementsCount)
158 }
159
160 private fun encryptMeta(meta: IMetaInfo): CommonMetaInfo {
161     return CommonMetaInfo(
162         size = meta.size,
163         isDeleted = meta.isDeleted,
164         isHidden = meta.isHidden,
165         lastModified = meta.lastModified,
166         path = encryptPath(meta.path)
167     )
168 }
169
170 private fun decryptMeta(meta: IMetaInfo): CommonMetaInfo {
171     return CommonMetaInfo(
172         size = meta.size,
173         isDeleted = meta.isDeleted,
174         isHidden = meta.isHidden,

```

```

175         lastModified = meta.lastModified,
176         path = decryptPath(meta.path)
177     )
178 }
179
180 private fun encryptPath(pathStr: String): String {
181     if(pathEncryptor == null)
182         return pathStr
183     val path = Path(pathStr)
184     val segments = mutableList0f<String>()
185     for (segment in path)
186         segments.add(pathEncryptor.encryptString(segment.pathString))
187     val res = Path("/", *(segments.toTypedArray()))
188     return res.pathString
189 }
190
191 private fun decryptPath(pathStr: String): String {
192     if(pathEncryptor == null)
193         return pathStr
194
195     val path = Path(pathStr)
196     val segments = mutableList0f<String>()
197     for (segment in path)
198         segments.add(pathEncryptor.decryptString(segment.pathString))
199     val res = Path("/", *(segments.toTypedArray()))
200     return res.pathString
201 }
202
203 override suspend fun getAllFiles(): List<IFile> {
204     return source.getAllFiles().map(::decryptEntity).filterSystemHiddenFiles()
205 }
206
207 override suspend fun getFiles(path: String): List<IFile> {
208     return
209     source.getFiles(encryptPath(path)).map(::decryptEntity).filterSystemHiddenFiles()
210 }
211
212 override fun getFilesFlow(path: String): Flow<DataPage<IFile>> {
213     val flow = source.getFilesFlow(encryptPath(path)).map { page ->
214         DataPage(
215             list = page.data.map(::decryptEntity).filterSystemHiddenFiles(),
216             isLoading = page.isLoading,
217             isError = page.isError,
218             hasNext = page.hasNext,
219             pageLength = page.pageLength,

```

```

219         pageIndex = page.pageIndex,
220     )
221 }
222 return flow
223 }
224
225 override suspend fun getAllDirs(): List<IDirectory> {
226     return source.getAllDirs().map(::decryptEntity).filterSystemHiddenDirs()
227 }
228
229 override suspend fun getDirs(path: String): List<IDirectory> {
230     return
231     source.getDirs(encryptPath(path)).map(::decryptEntity).filterSystemHiddenDirs()
232 }
233
234 override fun getDirsFlow(path: String): Flow<DataPage<IDirectory>> {
235     val flow = source.getDirsFlow(encryptPath(path)).map { page ->
236         DataPage(
237             // включать все папки, кроме системной
238             list = page.data.map(::decryptEntity).filterSystemHiddenDirs(),
239             isLoading = page.isLoading,
240             isError = page.isError,
241             hasNext = page.hasNext,
242             pageLength = page.pageLength,
243             pageIndex = page.pageIndex,
244         )
245     }
246     return flow
247 }
248
249 override suspend fun getFileInfo(path: String): IFile {
250     val file = source.getFileInfo(encryptPath(path))
251     val meta = decryptMeta(file.metaInfo)
252     return CommonFile(meta)
253 }
254
255 override suspend fun getDirInfo(path: String): IDirectory {
256     val dir = source.getDirInfo(encryptPath(path))
257     val meta = decryptMeta(dir.metaInfo)
258     return CommonDirectory(meta, dir.elementsCount)
259 }
260
261 override suspend fun setHidden(path: String, hidden: Boolean) {
262     source.setHidden(encryptPath(path), hidden)
263 }

```

```

263
264     override suspend fun touchFile(path: String) {
265         source.touchFile(encryptPath(path))
266         appendSyncEntry(path = path, operation = StorageSyncOperation.UPSERT)
267     }
268
269     override suspend fun touchDir(path: String) {
270         source.touchDir(encryptPath(path))
271     }
272
273     override suspend fun delete(path: String, recordSyncJournal: Boolean) {
274         source.delete(encryptPath(path), recordSyncJournal = false)
275         if (recordSyncJournal) {
276             appendSyncEntry(path = path, operation = StorageSyncOperation.DELETE)
277         }
278     }
279
280     override suspend fun openWrite(path: String, recordSyncJournal: Boolean): OutputStream =
281         openWriteInternal(path, recordJournal = recordSyncJournal)
282
283     override suspend fun openRead(path: String): InputStream {
284         val stream = source.openRead(encryptPath(path))
285         return dataEncryptor.decryptStream(stream)
286     }
287
288     override suspend fun moveToTrash(path: String, recordSyncJournal: Boolean) {
289         source.moveToTrash(encryptPath(path), recordSyncJournal = false)
290         if (recordSyncJournal) {
291             appendSyncEntry(path = path, operation = StorageSyncOperation.TRASH)
292         }
293     }
294
295     override fun dispose() {
296         runBlocking {
297             runCatching { journalBuffer.flushPending() }
298         }
299         dataEncryptor.dispose()
300     }
301
302     override suspend fun openReadSystemFile(name: String): InputStream = scope.run {
303         val path = Path(systemHiddenDirName, name).pathString
304         try {
305             openRead(path)
306         } catch (_, WallencException.Storage.FileNotFound) {
307             ByteArrayInputStream(ByteArray(0))

```

```

308     }
309 }
310
311 override suspend fun openWriteSystemFile(name: String): OutputStream = scope.run {
312     val path = Path(systemHiddenDirName, name).pathString
313     systemHiddenFilesIsActual = false
314     return@run openWriteInternal(path, recordJournal = false).onClosing {
315         systemHiddenFilesIsActual = false
316     }
317 }
318
319 override suspend fun readSyncJournal(): StorageSyncJournal {
320     journalBuffer.flushPending()
321     return readSyncJournalUnchecked()
322 }
323
324 override suspend fun flushPendingSyncJournal() {
325     journalBuffer.flushPending()
326 }
327
328 override suspend fun putSyncJournalEntries(entries: StorageSyncJournal) {
329     journalBuffer.putEntries(entries)
330 }
331
332 private suspend fun readSyncJournalUnchecked(): StorageSyncJournal {
333     val bytes = readSystemFileBytesOrEmpty { openReadSystemFile(SYNC_JOURNAL_FILENAME) }
334     return StorageSyncJournalCodec.read(jackson, bytes)
335 }
336
337 private suspend fun writeSyncJournal(journal: StorageSyncJournal) {
338     openWriteSystemFile(SYNC_JOURNAL_FILENAME).use { out ->
339         jackson.writeValue(out, journal)
340     }
341 }
342
343 override suspend fun readSyncLock(): StorageSyncLock? {
344     val bytes = readSystemFileBytesOrEmpty { openReadSystemFile(SYNC_LOCK_FILENAME) }
345     if (bytes.isEmpty()) {
346         return null
347     }
348     return runCatching {
349         jackson.readValue(bytes, StorageSyncLock::class.java)
350     }.getOrNull()
351 }
352

```

```

353     override suspend fun tryAcquireSyncLock(holderId: String, leaseUntil: Instant): Boolean
354     {
355         return syncLockMutex.withLock {
356             val current = readSyncLock()
357             val now = Instant.now()
358             val foreignLockActive = current != null &&
359                 current.holderId != holderId &&
360                 current.leaseUntil.isAfter(now) &&
361                 current.updatedAt.plusSeconds(SYNC_LOCK_STALE_TIMEOUT_SECONDS).isAfter(now)
362             if (foreignLockActive) {
363                 return@withLock false
364             }
365             val next = StorageSyncLock(
366                 holderId = holderId,
367                 leaseUntil = leaseUntil,
368                 updatedAt = now,
369             )
370             openWriteSystemFile(SYNC_LOCK_FILENAME).use { out ->
371                 jackson.writeValue(out, next)
372             }
373             readSyncLock()?.holderId == holderId
374         }
375     }
376
377     override suspend fun releaseSyncLock(holderId: String) {
378         syncLockMutex.withLock {
379             val current = readSyncLock() ?: return@withLock
380             if (current.holderId != holderId) {
381                 return@withLock
382             }
383             openWriteSystemFile(SYNC_LOCK_FILENAME).use { out ->
384                 out.write(ByteArray(0))
385             }
386         }
387     }
388
389     override suspend fun forceClearSyncLock() {
390         syncLockMutex.withLock {
391             openWriteSystemFile(SYNC_LOCK_FILENAME).use { out ->
392                 out.write(ByteArray(0))
393             }
394         }
395     }
396
397     private suspend fun appendSyncEntry(path: String, operation: StorageSyncOperation) {

```



```

397         val cleanedPath = StorageSyncPaths.normalize(path)
398         if (!StorageSyncPaths.isSyncableUserPath(cleanedPath)) {
399             return
400         }
401         val sequence = journalBuffer.nextSequence()
402         val entry = journalBuffer.buildEntry(cleanedPath, operation, sequence)
403         journalBuffer.appendEntry(cleanedPath, entry)
404     }
405
406     private suspend fun openWriteInternal(path: String, recordJournal: Boolean):
OutputStream {
407         val stream = source.openWrite(encryptPath(path))
408         val encrypted = dataEncryptor.encryptStream(stream)
409         if (!recordJournal) {
410             return encrypted
411         }
412         return encrypted.onClosed {
413             scope.launch {
414                 appendSyncEntry(path = path, operation = StorageSyncOperation.UPSERT)
415             }
416         }
417     }
418
419     private fun Iterable<IFile>.filterSystemHiddenFiles(): List<IFile> {
420         return this.filter { file ->
421             !file.metaInfo.path.contains(
422                 systemHiddenDirName
423             )
424         }
425     }
426
427     private fun Iterable<IDirectory>.filterSystemHiddenDirs(): List<IDirectory> {
428         return this.filter { dir ->
429             !dir.metaInfo.path.contains(
430                 systemHiddenDirName
431             )
432         }
433     }
434
435     private companion object {
436         private const val SYNC_JOURNAL_FILENAME = "sync-journal.json"
437         private const val SYNC_LOCK_FILENAME = "sync-lock.json"
438         private const val SYNC_LOCK_STALE_TIMEOUT_SECONDS: Long = 60 * 60
439         private val jackson = com.fasterxml.jackson.module.kotlin.jacksonObjectMapper()
440             .apply { findAndRegisterModules() }

```

441 }
442 }

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/local/LocalStorage.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.local
2
3 import com.github.nullptroma.wallenc.domain.vault.storages.common.BaseStorage
4 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
5 import kotlinx.coroutines.CoroutineDispatcher
6 import java.util.UUID
7
8 class LocalStorage(
9     uuid: UUID,
10     val absolutePath: String,
11     ioDispatcher: CoroutineDispatcher,
12 ) : BaseStorage(
13     uuid = uuid,
14     ioDispatcher = ioDispatcher,
15     metaInfoFilePostfix = STORAGE_INFO_FILE_POSTFIX,
16 ) {
17     private val _accessor = LocalStorageAccessor(absolutePath, ioDispatcher)
18     override val accessor: IStorageAccessor = _accessor
19     override val isVirtualStorage: Boolean = false
20
21     override suspend fun init() {
22         _accessor.init()
23         super.init()
24     }
25
26     companion object {
27         const val STORAGE_INFO_FILE_POSTFIX = ".storage-info"
28     }
29 }
30
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/local/LocalStorageAccessor.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.local
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.domain.vault.storages.common.StorageSyncJournalBuffer
5 import com.github.nullptroma.wallenc.domain.vault.storages.common.StorageSyncJournalCodec
6 import com.github.nullptroma.wallenc.domain.vault.storages.common.readSystemFileBytesOrEmpty
7
8 import com.fasterxml.jackson.core.JsonException
9 import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
10 import com.fasterxml.jackson.module.kotlin.readValue
11 import
12 com.github.nullptroma.wallenc.domain.vault.utils.CloseHandledStreamExtension.Companion.onClose
13 import com.github.nullptroma.wallenc.domain.common.impl.CommonDirectory
14 import com.github.nullptroma.wallenc.domain.common.impl.CommonFile
15 import com.github.nullptroma.wallenc.domain.common.impl.CommonMetaInfo
16 import com.github.nullptroma.wallenc.domain.datatypes.DataPage
17 import com.github.nullptroma.wallenc.domain.interfaces.IDirectory
18 import com.github.nullptroma.wallenc.domain.interfaces.IFile
19 import com.github.nullptroma.wallenc.domain.interfaces.IMetaInfo
20 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
21 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournal
22 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
23 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalMerge
24 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncLock
25 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation
26 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths
27 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncRevision
28 import kotlinx.coroutines.CoroutineDispatcher
29 import kotlinx.coroutines.CoroutineScope
30 import kotlinx.coroutines.flow.Flow
31 import kotlinx.coroutines.sync.Mutex
32 import kotlinx.coroutines.sync.withLock
33 import kotlinx.coroutines.flow.MutableSharedFlow
34 import kotlinx.coroutines.flow.MutableStateFlow
35 import kotlinx.coroutines.flow.SharedFlow
36 import kotlinx.coroutines.flow.StateFlow
37 import kotlinx.coroutines.flow.flow
38 import kotlinx.coroutines.flow.flowOn
39 import kotlinx.coroutines.launch
40 import kotlinx.coroutines.withContext
41 import java.io.File
42 import java.io.InputStream
```

```

42 import java.io.OutputStream
43 import java.nio.ByteBuffer
44 import java.nio.channels.FileChannel
45 import java.nio.file.Files
46 import java.nio.file.Path
47 import java.nio.file.StandardOpenOption
48 import java.time.Clock
49 import java.time.Instant
50 import java.util.UUID
51 import kotlin.io.path.Path
52 import kotlin.io.path.absolute
53 import kotlin.io.path.fileSize
54 import kotlin.io.path.pathString
55 import kotlin.io.path.relativeTo
56
57 class LocalStorageAccessor(
58     filesystemBasePath: String,
59     private val ioDispatcher: CoroutineDispatcher
60 ) : IStorageAccessor {
61     private val syncActorId = UUID.randomUUID().toString()
62     private val _filesystemBasePath: Path = Path(filesystemBasePath).normalize().absolute()
63
64     private val _size = MutableStateFlow<Long?>(null)
65     override val size: StateFlow<Long?> = _size
66
67     private val _numberOfFiles = MutableStateFlow<Int?>(null)
68     override val numberOfFiles: StateFlow<Int?> = _numberOfFiles
69
70     private val _isAvailable = MutableStateFlow(false)
71     override val isAvailable: StateFlow<Boolean> = _isAvailable
72
73     private val _filesUpdates = MutableSharedFlow<DataPage<IFile>>()
74     override val filesUpdates: SharedFlow<DataPage<IFile>> = _filesUpdates
75
76     private val _dirsUpdates = MutableSharedFlow<DataPage<IDirectory>>()
77     override val dirsUpdates: SharedFlow<DataPage<IDirectory>> = _dirsUpdates
78
79     private val journalBuffer = StorageSyncJournalBuffer(
80         syncActorId = syncActorId,
81         originStorageUuid = null,
82         readJournal = { readSyncJournalUnchecked() },
83         writeJournal = { writeSyncJournal(it) },
84     )
85
86     suspend fun init() = withContext(ioDispatcher) {

```

```

87         // запускаем сканирование хранилища
88         scanSizeAndNumOfFiles()
89     }
90
91     /**
92     * Проверяет существование корневого пути Storage в файловой системе, изменяет
    _isAvailable
93     */
94     private fun checkAvailable(): Boolean {
95         _isAvailable.value = _filesystemBasePath.toFile().exists()
96         return _isAvailable.value
97     }
98
99     /**
100     * Перебирает все файлы в файловой системе
101     * @param dir стартовый каталог
102     * @param maxDepth максимальная глубина (отрицательное для бесконечной)
103     * @param callback метод обратного вызова для каждого файла и директории
104     */
105     private suspend fun scanFileSystem(
106         dir: File,
107         maxDepth: Int,
108         callback: suspend (File) -> Unit,
109         useCallbackForSelf: Boolean = true
110     ) {
111         if (!dir.exists())
112             return
113
114
115         val children = dir.listFiles()
116         if (children != null) {
117
118             // вызвать коллбек для каждого элемента директории
119             for (child in children) {
120                 if (child.name != SYSTEM_HIDDEN_DIRNAME)
121                     callback(child)
122             }
123
124             if (useCallbackForSelf)
125                 callback(dir)
126
127             if (maxDepth != 0) {
128                 val nextMaxDepth = if (maxDepth > 0) maxDepth - 1 else maxDepth
129                 for (child in children) {
130                     if (child.isDirectory && child.name != SYSTEM_HIDDEN_DIRNAME) {

```

```

131         scanFileSystem(child, nextMaxDepth, callback, false)
132     }
133 }
134 }
135 } else if (useCallbackForSelf) {
136     callback(dir)
137 }
138 }
139
140 private class LocalStorageFilePair private constructor(
141     val file: File,
142     val metaFile: File,
143     val meta: CommonMetaInfo
144 ) {
145
146     companion object {
147         private val jackson = jacksonObjectMapper().apply { findAndRegisterModules() }
148
149         fun fromFile(filesystemBasePath: Path, file: File): LocalStorageFilePair? {
150             if (!file.exists())
151                 return null
152             if (file.name.endsWith(META_INFO_POSTFIX))
153                 return fromMetaFile(filesystemBasePath, file)
154
155             val filePath = file.toPath()
156             val metaFilePath = Path(
157                 if (file.isFile) {
158                     file.absolutePath + META_INFO_POSTFIX
159                 } else {
160                     Path(file.absolutePath, META_INFO_POSTFIX).pathString
161                 }
162             )
163             val metaFile = metaFilePath.toFile()
164             val metaInfo: CommonMetaInfo
165             val storageFilePath = "/" + filePath.relativeTo(filesystemBasePath)
166
167             if (!metaFile.exists()) {
168                 metaInfo = CommonMetaInfo(
169                     size = filePath.fileSize(),
170                     path = storageFilePath
171                 )
172                 jackson.writeValue(metaFile, metaInfo)
173             } else {
174                 var readMeta: CommonMetaInfo
175                 try {

```

```

176         val reader = metaFile.bufferedReader()
177         readMeta = jackson.readValue(reader)
178     } catch (e: JacksonException) {
179         // если файл повреждён - пересоздать
180         readMeta = CommonMetaInfo(
181             size = filePath.fileSize(),
182             path = storageFilePath
183         )
184         jackson.writeValue(metaFile, readMeta)
185     }
186     metaInfo = readMeta
187 }
188 return LocalStorageFilePair(
189     file = file,
190     metaFile = metaFile,
191     meta = metaInfo
192 )
193 }
194
195 fun fromMetaFile(filesystemBasePath: Path, metaFile: File):
LocalStorageFilePair? {
196     if (!metaFile.exists())
197         return null
198     if (!metaFile.name.endsWith(META_INFO_POSTFIX))
199         return fromFile(filesystemBasePath, metaFile)
200     var pair: LocalStorageFilePair? = null
201     try {
202         val reader = metaFile.bufferedReader()
203         val metaInfo: CommonMetaInfo = jackson.readValue(reader)
204         val pathString = Path(filesystemBasePath.pathString,
metaInfo.path).pathString
205         val file = File(pathString)
206         if (!file.exists()) {
207             metaFile.delete()
208         } else {
209             pair = LocalStorageFilePair(
210                 file = file,
211                 metaFile = metaFile,
212                 meta = metaInfo
213             )
214         }
215     } catch (e: JacksonException) {
216         metaFile.delete()
217     }
218     return pair
219 }

```



```

220
221         fun from(filesystemBasePath: Path, anyFile: File): LocalStorageFilePair? {
222             return if (anyFile.name.endsWith(META_INFO_POSTFIX))
223                 fromMetaFile(filesystemBasePath, anyFile)
224             else
225                 fromFile(filesystemBasePath, anyFile)
226         }
227
228         fun from(filesystemBasePath: Path, storagePath: String): LocalStorageFilePair? {
229             val filePath = Path(filesystemBasePath.pathString, storagePath)
230             return from(filesystemBasePath, filePath.toFile())
231         }
232     }
233 }
234
235 /**
236  * Перебирает все файлы и каталоги в relativePath и возвращает с мета-информацией
237  *
238  */
239 private suspend fun scanStorage(
240     baseStoragePath: String,
241     maxDepth: Int,
242     fileCallback: (suspend (File, CommonFile) -> Unit)? = null,
243     dirCallback: (suspend (File, CommonDirectory) -> Unit)? = null
244 ) {
245     if (!checkAvailable())
246         throw WallencException.Storage.NotAvailable()
247     val basePath = Path(_filesystemBasePath.pathString, baseStoragePath)
248     val workedFiles = mutableSetOf<String>()
249     val workedMetaFiles = mutableSetOf<String>()
250
251     scanFileSystem(basePath.toFile(), maxDepth, { file ->
252         // Если парный файл уже был обработан - скип. Это позволит не читать metaFile
253         дважды
254         if (workedFiles.contains(file.absolutePath) ||
255             workedMetaFiles.contains(file.absolutePath)) {
256             return@scanFileSystem
257         }
258
259         val pair = LocalStorageFilePair.from(_filesystemBasePath, file)
260         if (pair != null) {
261             workedFiles.add(pair.file.absolutePath)
262             workedMetaFiles.add(pair.metaFile.absolutePath)
263
264             if (pair.file.isFile) {

```

```

263         fileCallback?.invoke(pair.file, CommonFile(pair.meta))
264     } else {
265         dirCallback?.invoke(pair.file, CommonDirectory(pair.meta, null))
266     }
267 }
268 })
269 }
270
271
272 /**
273  * Считает файлы и их размер. Не бросает исключения, если файлы недоступны
274  * @throws none Если возникла ошибка, оставляет размер и количества файлов равными null
275  */
276 private suspend fun scanSizeAndNumOfFiles() = withContext(ioDispatcher) {
277     if (!checkAvailable()) {
278         _size.value = null
279         _numberOfFiles.value = null
280         return@withContext
281     }
282
283     var size = 0L
284     var numOfFiles = 0
285
286     scanStorage(baseStoragePath = "/", maxDepth = -1, fileCallback = { _, commonFile ->
287         size += commonFile.metaInfo.size
288         numOfFiles++
289
290         if (numOfFiles % DATA_PAGE_LENGTH == 0) {
291             _size.value = size
292             _numberOfFiles.value = numOfFiles
293         }
294     })
295
296     _size.value = size
297     _numberOfFiles.value = numOfFiles
298 }
299
300 override suspend fun getAllFiles(): List<IFile> = withContext(ioDispatcher) {
301     if (!checkAvailable())
302         return@withContext listOf()
303
304     val list = mutableListOf<IFile>()
305     scanStorage(baseStoragePath = "/", maxDepth = -1, fileCallback = { _, commonFile ->
306         list.add(commonFile)
307     })

```

```

308         return@withContext list
309     }
310
311     override suspend fun getFiles(path: String): List<IFile> = withContext(ioDispatcher) {
312         if (!checkAvailable())
313             return@withContext listOf()
314
315         val list = mutableListOf<IFile>()
316         scanStorage(baseStoragePath = path, maxDepth = 0, fileCallback = { _, commonFile ->
317             list.add(commonFile)
318         })
319         return@withContext list
320     }
321
322     override fun getFilesFlow(path: String): Flow<DataPage<IFile>> = flow {
323         if (!checkAvailable())
324             return@flow
325
326         val buf = mutableListOf<IFile>()
327         var pageNumber = 0
328         scanStorage(baseStoragePath = path, maxDepth = 0, fileCallback = { _, commonFile ->
329             if (buf.size == DATA_PAGE_LENGTH) {
330                 val page = DataPage(
331                     list = buf.toList(),
332                     isLoading = false,
333                     isError = false,
334                     hasNext = true,
335                     pageLength = DATA_PAGE_LENGTH,
336                     pageIndex = pageNumber++
337                 )
338                 emit(page)
339                 buf.clear()
340             }
341             buf.add(commonFile)
342         })
343         // отправка последней страницы
344         val page = DataPage(
345             list = buf.toList(),
346             isLoading = false,
347             isError = false,
348             hasNext = false,
349             pageLength = DATA_PAGE_LENGTH,
350             pageIndex = pageNumber++
351         )
352         emit(page)

```

```

353     }.flowOn(ioDispatcher)
354
355     override suspend fun getAllDirs(): List<IDirectory> = withContext(ioDispatcher) {
356         if (!checkAvailable())
357             return@withContext listOf()
358
359         val list = mutableListOf<IDirectory>()
360         scanStorage(baseStoragePath = "/", maxDepth = -1, dirCallback = { _, localDir ->
361             list.add(localDir)
362         })
363         return@withContext list
364     }
365
366     {
367         override suspend fun getDirs(path: String): List<IDirectory> = withContext(ioDispatcher) {
368             if (!checkAvailable())
369                 return@withContext listOf()
370
371             val list = mutableListOf<IDirectory>()
372             scanStorage(baseStoragePath = path, maxDepth = 0, dirCallback = { _, localDir ->
373                 list.add(localDir)
374             })
375             return@withContext list
376         }
377
378         override fun getDirsFlow(path: String): Flow<DataPage<IDirectory>> = flow {
379             if (!checkAvailable())
380                 return@flow
381
382             val buf = mutableListOf<IDirectory>()
383             var pageNumber = 0
384             scanStorage(baseStoragePath = path, maxDepth = 0, dirCallback = { _, localDir ->
385                 if (buf.size == DATA_PAGE_LENGTH) {
386                     val page = DataPage(
387                         list = buf.toList(),
388                         isLoading = false,
389                         isError = false,
390                         hasNext = true,
391                         pageLength = DATA_PAGE_LENGTH,
392                         pageIndex = pageNumber++
393                     )
394                     emit(page)
395                     buf.clear()
396                 }
397                 buf.add(localDir)

```

```

397     })
398     // отправка последней страницы
399     val page = DataPage(
400         list = buf.toList(),
401         isLoading = false,
402         isError = false,
403         hasNext = false,
404         pageLength = DATA_PAGE_LENGTH,
405         pageIndex = pageNumber++
406     )
407     emit(page)
408 }.flowOn(ioDispatcher)
409
410 override suspend fun getFileInfo(path: String): IFile {
411     val pair = LocalStorageFilePair.from(_filesystemBasePath, path)
412         ?: throw WallencException.Storage.UnexpectedState()
413     return CommonFile(
414         metaInfo = pair.meta,
415     )
416 }
417
418 override suspend fun getDirInfo(path: String): IDirectory {
419     val pair = LocalStorageFilePair.from(_filesystemBasePath, path)
420         ?: throw WallencException.Storage.UnexpectedState()
421     return CommonDirectory(
422         metaInfo = pair.meta,
423         elementsCount = null
424     )
425 }
426
427 override suspend fun setHidden(path: String, hidden: Boolean) {
428     val pair = LocalStorageFilePair.from(_filesystemBasePath, path)
429         ?: throw WallencException.Storage.UnexpectedState()
430     if (pair.meta.isHidden == hidden)
431         return
432     val newMeta = pair.meta.copy(isHidden = hidden)
433     writeMeta(pair.metaFile, newMeta)
434     _filesUpdates.emit(
435         DataPage(
436             list = listOf(
437                 CommonFile(
438                     metaInfo = newMeta
439                 )
440             ),
441             pageLength = 1,

```

```

442         pageIndex = 0
443     )
444 )
445 }
446
447
448 private fun writeMeta(metaFile: File, meta: IMetaInfo) {
449     jackson.writeValue(metaFile, meta)
450 }
451
452 private suspend fun createFile(storagePath: String): CommonFile =
withContext(ioDispatcher) {
453     val path = Path(_filesystemBasePath.pathString, storagePath)
454     val file = pathToFile()
455     if (file.exists() && file.isDirectory) {
456         throw WallencException.Storage.UnexpectedState()
457     } else if (!file.exists()) {
458         val parent = Path(storagePath).parent
459         createDir(parent.pathString)
460         file.createNewFile()
461
462         val cur = _numberOfFiles.value
463         _numberOfFiles.value = if (cur == null) null else cur + 1
464     }
465
466     val pair = LocalStorageFilePair.from(_filesystemBasePath, file)
467     ?: throw WallencException.Storage.UnexpectedState()
468     val newMeta = pair.meta.copy(lastModified = Clock.systemUTC().instant(), size =
Files.size(pair.file.toPath()))
469     writeMeta(pair.metaFile, newMeta)
470     _filesUpdates.emit(
471         DataPage(
472             list = listOf(CommonFile(pair.meta)),
473             pageLength = 1,
474             pageIndex = 0
475         )
476     )
477     return@withContext CommonFile(newMeta)
478 }
479
480 private suspend fun createDir(storagePath: String): CommonDirectory =
withContext(ioDispatcher) {
481     val path = Path(_filesystemBasePath.pathString, storagePath)
482     val file = pathToFile()
483     if (file.exists() && !file.isDirectory) {
484         throw WallencException.Storage.UnexpectedState()

```

```

485         } else if(!file.exists()) {
486             Files.createDirectories(path)
487         }
488
489         val pair = LocalStorageFilePair.from(_filesystemBasePath, file)
490             ?: throw WallencException.Storage.UnexpectedState()
491         val newMeta = pair.meta.copy(lastModified = Clock.systemUTC().instant())
492         writeMeta(pair.metaFile, newMeta)
493         _dirsUpdates.emit(
494             DataPage(
495                 list = listOf(CommonDirectory(pair.meta, null)),
496                 pageLength = 1,
497                 pageIndex = 0
498             )
499         )
500         return@withContext CommonDirectory(newMeta, 0)
501     }
502
503     override suspend fun touchFile(path: String): Unit = withContext(ioDispatcher) {
504         touchFileInternal(path, recordJournal = true)
505     }
506
507     private suspend fun touchFileInternal(path: String, recordJournal: Boolean) {
508         createFile(path)
509
510         if (recordJournal) {
511             appendSyncEntry(path = path, operation = StorageSyncOperation.UPSERT)
512         }
513
514         // перебор все каталогов и обновление их времени модификации
515         var parent = Path(path).parent
516         while(parent != null) {
517             touchDir(parent.pathString)
518             parent = parent.parent
519         }
520     }
521
522     override suspend fun touchDir(path: String): Unit = withContext(ioDispatcher) {
523         createDir(path)
524     }
525
526     override suspend fun delete(path: String, recordSyncJournal: Boolean) =
527         withContext(ioDispatcher) {
528             if (path == "/" || path.isBlank()) {
529                 throw WallencException.Storage.DeleteRootForbidden()

```

```

529     }
530     val pair = LocalStorageFilePair.from(_filesystemBasePath, path)
531     if (pair != null) {
532         if (pair.file.isDirectory) pair.file.deleteRecursively()
533         else pair.file.delete()
534         pair.metaFile.delete()
535         scanSizeAndNumOfFiles()
536         if (recordSyncJournal) {
537             appendSyncEntry(path = path, operation = StorageSyncOperation.DELETE)
538         }
539     }
540 }
541
542 override suspend fun openWrite(path: String, recordSyncJournal: Boolean): OutputStream =
withContext(ioDispatcher) {
543     touchFileInternal(path, recordJournal = false)
544     val pair = LocalStorageFilePair.from(_filesystemBasePath, path)
545     ?: throw WallencException.Storage.FileNotFound()
546     return@withContext pair.file.outputStream().onClosed {
547         CoroutineScope(ioDispatcher).launch {
548             touchFileInternal(path, recordJournal = false)
549             scanSizeAndNumOfFiles()
550             if (recordSyncJournal) {
551                 appendSyncEntry(path = path, operation = StorageSyncOperation.UPSERT)
552             }
553         }
554     }
555 }
556
557 override suspend fun openRead(path: String): InputStream = withContext(ioDispatcher) {
558     val pair = LocalStorageFilePair.from(_filesystemBasePath, path)
559     ?: throw WallencException.Storage.FileNotFound()
560     return@withContext pair.file.inputStream()
561 }
562
563 override suspend fun moveToTrash(path: String, recordSyncJournal: Boolean) =
withContext(ioDispatcher) {
564     val pair = LocalStorageFilePair.from(_filesystemBasePath, path)
565     ?: throw WallencException.Storage.FileNotFound()
566     val newMeta = pair.meta.copy(isDeleted = true)
567     writeMeta(pair.metaFile, newMeta)
568     if (recordSyncJournal) {
569         appendSyncEntry(path = path, operation = StorageSyncOperation.TRASH)
570     }
571 }
572

```



```

573         override suspend fun openReadSystemFile(name: String): InputStream =
withContext(ioDispatcher) {
574             val dirPath = _filesystemBasePath.resolve(SYSTEM_HIDDEN_DIRNAME)
575             val path = dirPath.resolve(name)
576             val file = path.toFile()
577             if (!file.exists()) {
578                 throw WallencException.Storage.FileNotFound()
579             }
580             return@withContext file.inputStream()
581         }
582
583         override suspend fun openWriteSystemFile(name: String): OutputStream =
withContext(ioDispatcher) {
584             val dirPath = _filesystemBasePath.resolve(SYSTEM_HIDDEN_DIRNAME)
585             val path = dirPath.resolve(name)
586             val file = path.toFile()
587             if(!file.exists()) {
588                 Files.createDirectories(dirPath)
589                 file.createNewFile()
590             }
591
592             return@withContext file.outputStream()
593         }
594
595         override suspend fun readSyncJournal(): StorageSyncJournal = withContext(ioDispatcher) {
596             journalBuffer.flushPending()
597             readSyncJournalUnchecked()
598         }
599
600         override suspend fun flushPendingSyncJournal() = withContext(ioDispatcher) {
601             journalBuffer.flushPending()
602         }
603
604         override suspend fun putSyncJournalEntries(entries: StorageSyncJournal) =
withContext(ioDispatcher) {
605             journalBuffer.putEntries(entries)
606         }
607
608         private suspend fun readSyncJournalUnchecked(): StorageSyncJournal {
609             val bytes = readSystemFileBytesOrEmpty { openReadSystemFile(SYNC_JOURNAL_FILENAME) }
610             return StorageSyncJournalCodec.read(jackson, bytes)
611         }
612
613         private suspend fun writeSyncJournal(journal: StorageSyncJournal) {
614             openWriteSystemFile(SYNC_JOURNAL_FILENAME).use { out ->
615                 jackson.writeValue(out, journal)

```

```

616     }
617 }
618
619 override suspend fun readSyncLock(): StorageSyncLock? = withContext(ioDispatcher) {
620     val bytes = readSystemFileBytesOrEmpty { openReadSystemFile(SYNC_LOCK_FILENAME) }
621     if (bytes.isEmpty()) {
622         return@withContext null
623     }
624     return@withContext runCatching {
625         jackson.readValue<StorageSyncLock>(bytes)
626     }.getOrNull()
627 }
628
629 override suspend fun tryAcquireSyncLock(holderId: String, leaseUntil: Instant): Boolean
= withContext(ioDispatcher) {
630     val lockPath = systemLockPath()
631     Files.createDirectories(lockPath.parent)
632     FileChannel.open(
633         lockPath,
634         StandardOpenOption.CREATE,
635         StandardOpenOption.READ,
636         StandardOpenOption.WRITE,
637     ).use { channel ->
638         val fileLock = channel.lock()
639         try {
640             val current = readSyncLockFromChannel(channel)
641             val now = Instant.now()
642             val foreignLockActive = current != null &&
643                 current.holderId != holderId &&
644                 current.leaseUntil.isAfter(now) &&
645             current.updatedAt.plusSeconds(SYNC_LOCK_STALE_TIMEOUT_SECONDS).isAfter(now)
646             if (foreignLockActive) {
647                 return@withContext false
648             }
649
650             writeSyncLockToChannel(
651                 channel = channel,
652                 lock = StorageSyncLock(
653                     holderId = holderId,
654                     leaseUntil = leaseUntil,
655                     updatedAt = now,
656                 ),
657             )
658             return@withContext true
659         } finally {

```

```

660         fileLock.release()
661     }
662 }
663 }
664
665 override suspend fun releaseSyncLock(holderId: String) = withContext(ioDispatcher) {
666     val lockPath = systemLockPath()
667     Files.createDirectories(lockPath.parent)
668     FileChannel.open(
669         lockPath,
670         StandardOpenOption.CREATE,
671         StandardOpenOption.READ,
672         StandardOpenOption.WRITE,
673     ).use { channel ->
674         val fileLock = channel.lock()
675         try {
676             val current = readSyncLockFromChannel(channel) ?: return@withContext
677             if (current.holderId != holderId) {
678                 return@withContext
679             }
680             channel.truncate(0)
681             channel.force(true)
682         } finally {
683             fileLock.release()
684         }
685     }
686 }
687
688 override suspend fun forceClearSyncLock() = withContext(ioDispatcher) {
689     val lockPath = systemLockPath()
690     if (!Files.exists(lockPath)) {
691         return@withContext
692     }
693     Files.createDirectories(lockPath.parent)
694     FileChannel.open(
695         lockPath,
696         StandardOpenOption.READ,
697         StandardOpenOption.WRITE,
698     ).use { channel ->
699         val fileLock = channel.lock()
700         try {
701             channel.truncate(0)
702             channel.force(true)
703         } finally {
704             fileLock.release()

```

```

705         }
706     }
707 }
708
709 private fun systemLockPath(): Path =
710     _filesystemBasePath.resolve(SYSTEM_HIDDEN_DIRNAME).resolve(SYNC_LOCK_FILENAME)
711
712 private fun readSyncLockFromChannel(channel: FileChannel): StorageSyncLock? {
713     channel.position(0)
714     val size = channel.size().toInt()
715     if (size <= 0) {
716         return null
717     }
718     val buffer = ByteBuffer.allocate(size)
719     while (buffer.hasRemaining()) {
720         val read = channel.read(buffer)
721         if (read <= 0) break
722     }
723     val bytes = buffer.array().copyOf(buffer.position())
724     if (bytes.isEmpty()) {
725         return null
726     }
727     return runCatching { jackson.readValue(bytes,
StorageSyncLock::class.java) }.getOrNull()
728 }
729
730 private fun writeSyncLockToChannel(channel: FileChannel, lock: StorageSyncLock) {
731     val bytes = jackson.writeValueAsBytes(lock)
732     channel.truncate(0)
733     channel.position(0)
734     channel.write(ByteBuffer.wrap(bytes))
735     channel.force(true)
736 }
737
738 private suspend fun appendSyncEntry(path: String, operation: StorageSyncOperation) {
739     val cleanedPath = StorageSyncPaths.normalize(path)
740     if (!StorageSyncPaths.isSyncableUserPath(cleanedPath)) {
741         return
742     }
743     val sequence = journalBuffer.nextSequence()
744     val entry = journalBuffer.buildEntry(cleanedPath, operation, sequence)
745     journalBuffer.appendEntry(cleanedPath, entry)
746 }
747
748 companion object {

```

```

749         // Файлы, которые можно использовать для чтения и записи, но не отображаются в
хранилище
750         private const val SYSTEM_HIDDEN_DIRNAME = "wallenc-local-storage-meta-dir"
751         private const val META_INFO_POSTFIX = ".wallenc-meta"
752         private const val DATA_PAGE_LENGTH = 10
753         private const val SYNC_JOURNAL_FILENAME = "sync-journal.json"
754         private const val SYNC_LOCK_FILENAME = "sync-lock.json"
755         private const val SYNC_LOCK_STALE_TIMEOUT_SECONDS: Long = 60 * 60
756         private val jackson = jacksonObjectMapper().apply { findAndRegisterModules() }
757     }
758 }

```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/yandex/YandexStorage.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.yandex
2
3 import
4 com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository.YandexDiskRepository
5 import com.github.nullptroma.wallenc.domain.vault.storages.common.BaseStorage
6 import com.github.nullptroma.wallenc.domain.vault.storages.local.LocalStorage
7 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
8 import kotlinx.coroutines.CoroutineDispatcher
9 import kotlinx.coroutines.CoroutineScope
10 import kotlinx.coroutines.flow.StateFlow
11 import java.util.UUID
12
13 class YandexStorage(
14     uuid: UUID,
15     private val repo: YandexDiskRepository,
16     vaultAvailability: StateFlow<Boolean>,
17     private val ioDispatcher: CoroutineDispatcher,
18     accessorScope: CoroutineScope,
19     private val reportAuthFailure: () -> Unit,
20 ) : BaseStorage(
21     uuid = uuid,
22     ioDispatcher = ioDispatcher,
23     metaInfoFilePostfix = LocalStorage.STORAGE_INFO_FILE_POSTFIX,
24 ) {
25
26     private val _accessor = YandexStorageAccessor(
27         storageUuid = uuid,
28         repo = repo,
29         ioDispatcher = ioDispatcher,
30         vaultAvailability = vaultAvailability,
31         accessorScope = accessorScope,
32         reportAuthFailure = reportAuthFailure,
33     )
34
35     override val accessor: IStorageAccessor = _accessor
36     override val isVirtualStorage: Boolean = false
37
38     override suspend fun init() {
39         _accessor.init()
40         super.init()
41     }
42 }
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/storages/yandex/YandexStorageAccessor.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.yandex
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.domain.vault.errors.toVaultWallencException
5 import com.github.nullptroma.wallenc.domain.vault.storages.common.StorageSyncJournalBuffer
6 import com.github.nullptroma.wallenc.domain.vault.storages.common.StorageSyncJournalCodec
7 import com.github.nullptroma.wallenc.domain.vault.storages.common.readSystemFileBytesOrEmpty
8
9 import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
10 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskAuthException
11 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.ResourceDto
12 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.dto.YandexVaultPersistedStats
13 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository.YandexDiskRepository
14 import com.github.nullptroma.wallenc.domain.vault.utils.CloseHandledStreamExtension.Companion.onClose
15 import com.github.nullptroma.wallenc.domain.common.impl.CommonDirectory
16 import com.github.nullptroma.wallenc.domain.common.impl.CommonFile
17 import com.github.nullptroma.wallenc.domain.common.impl.CommonMetaInfo
18 import com.github.nullptroma.wallenc.domain.datatypes.DataPage
19 import com.github.nullptroma.wallenc.domain.interfaces.IDirectory
20 import com.github.nullptroma.wallenc.domain.interfaces.IFile
21 import com.github.nullptroma.wallenc.domain.interfaces.IStorageAccessor
22 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournal
23 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncPaths
24 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncLock
25 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation
26 import kotlinx.coroutines.CancellationException
27 import kotlinx.coroutines.CoroutineDispatcher
28 import kotlinx.coroutines.CoroutineScope
29 import kotlinx.coroutines.Job
30 import kotlinx.coroutines.delay
31 import kotlin.coroutines.coroutineContext
32 import kotlinx.coroutines.ensureActive
33 import kotlinx.coroutines.flow.Flow
34 import kotlinx.coroutines.flow.FlowCollector
35 import kotlinx.coroutines.flow.MutableSharedFlow
36 import kotlinx.coroutines.flow.MutableStateFlow
37 import kotlinx.coroutines.flow.SharingStarted
38 import kotlinx.coroutines.flow.SharedFlow
39 import kotlinx.coroutines.flow.StateFlow
40 import kotlinx.coroutines.flow.combine
```

```

41 import kotlinx.coroutines.flow.flow
42 import kotlinx.coroutines.flow.flowOn
43 import kotlinx.coroutines.flow.stateIn
44 import kotlinx.coroutines.launch
45 import kotlinx.coroutines.runBlocking
46 import kotlinx.coroutines.withContext
47 import kotlinx.coroutines.sync.Mutex
48 import kotlinx.coroutines.sync.withLock
49 import kotlinx.coroutines.withContext
50 import java.io.ByteArrayOutputStream
51 import java.io.File
52 import java.io.FileNotFoundException
53 import java.io.FileOutputStream
54 import java.io.IOException
55 import java.io.InputStream
56 import java.io.OutputStream
57 import java.time.Instant
58 import java.util.UUID
59 import kotlin.jvm.Volatile
60
61 /**
62  * Реализация [IStorageAccessor] для дерева файлов `app:/<storageUuid>/...` на Яндекс.Диске.
63  *
64  * [isAvailable] объединяет доступность удалённого хранилища ([vaultAvailability])
65  * и успешную локальную инициализацию ([storageReady]) этого аксессуара.
66  */
67 class YandexStorageAccessor(
68     private val storageUuid: UUID,
69     private val repo: YandexDiskRepository,
70     private val ioDispatcher: CoroutineDispatcher,
71     vaultAvailability: StateFlow<Boolean>,
72     private val accessorScope: CoroutineScope,
73     private val reportAuthFailure: () -> Unit,
74 ) : IStorageAccessor {
75     private val syncActorId = UUID.randomUUID().toString()
76     private val syncLockMutex = Mutex()
77
78     private val diskRoot = "app:/$storageUuid"
79
80     private val _size = MutableStateFlow<Long?>(null)
81     override val size: StateFlow<Long?> = _size
82
83     private val _numberOfFiles = MutableStateFlow<Int?>(null)
84     override val numberOfFiles: StateFlow<Int?> = _numberOfFiles
85

```



```

86     private val _storageReady = MutableStateFlow(false)
87
88     override val isAvailable: StateFlow<Boolean> =
89         combine(vaultAvailability, _storageReady) { vaultOk, ready -> vaultOk && ready }
90         .stateIn(accessorScope, SharingStarted.Eagerly, false)
91
92     private val _filesUpdates = MutableSharedFlow<DataPage<IFile>>()
93     override val filesUpdates: SharedFlow<DataPage<IFile>> = _filesUpdates
94
95     private val _dirsUpdates = MutableSharedFlow<DataPage<IDirectory>>()
96     override val dirsUpdates: SharedFlow<DataPage<IDirectory>> = _dirsUpdates
97
98     private var statsPersistJob: Job? = null
99
100    private val journalBuffer = StorageSyncJournalBuffer(
101        syncActorId = syncActorId,
102        originStorageUuid = storageUuid,
103        readJournal = { readSyncJournalUnchecked() },
104        writeJournal = { writeSyncJournal(it) },
105    )
106
107    @Volatile
108    private var systemDirEnsured: Boolean = false
109
110    suspend fun init() = withContext(ioDispatcher) {
111        try {
112            val persisted = readPersistedStats()
113            if (persisted != null) {
114                _size.value = persisted.totalBytes
115                _numberOfFiles.value = persisted.fileCount
116            } else {
117                try {
118                    scanSizeAndNumOfFiles()
119                    writePersistedStatsInternal()
120                } catch (e: CancellationException) {
121                    throw e
122                } catch (_: Exception) {
123                    // Полный обход дерева не обязателен для работы; при таймауте сети
storage остаётся доступным.
124                }
125            }
126            _storageReady.value = true
127        } catch (e: YandexDiskAuthException) {
128            reportAuthFailure()
129            _storageReady.value = false

```

```

130         throw WallencException.Auth.Failed()
131     } catch (e: Exception) {
132         _storageReady.value = false
133         throw e.toVaultWallencException()
134     }
135 }
136
137 private inline fun <T> guard(block: () -> T): T {
138     try {
139         return block()
140     } catch (e: CancellationException) {
141         throw e
142     } catch (e: YandexDiskAuthException) {
143         reportAuthFailure()
144         throw WallencException.Auth.Failed()
145     } catch (e: Throwable) {
146         throw e.toVaultWallencException()
147     }
148 }
149
150 private fun toDiskPath(rel: String): String {
151     val tail = when {
152         rel.isBlank() || rel == "/" -> ""
153         else -> rel.trimStart('/')
154     }
155     // Корень хранилища – каталог в app;; для list/get API стабильнее с завершающим «/».
156     return if (tail.isEmpty()) "$diskRoot/" else "$diskRoot/$tail"
157 }
158
159 private fun toRelPath(diskPath: String): String {
160     val u = storageUuid.toString()
161     val app = "app:/$u/"
162     if (diskPath.startsWith(app)) {
163         val tail = diskPath.removePrefix(app).removeSuffix("/")
164         return if (tail.isEmpty()) "/" else "$tail"
165     }
166     val needle = "/$u/"
167     val i = diskPath.indexOf(needle)
168     if (i >= 0) {
169         val tail = diskPath.substring(i + needle.length).removeSuffix("/")
170         return if (tail.isEmpty()) "/" else "$tail"
171     }
172     val needleEnd = "/$u"
173     if (diskPath.endsWith(needleEnd, ignoreCase = true)) {
174         return "/"
175     }

```

```

175         }
176         return "/"
177     }
178
179     private fun isSystemDiskPath(diskPath: String?): Boolean {
180         if (diskPath == null) return false
181         return diskPath.contains("/$SYSTEM_HIDDEN_DIRNAME/") ||
182             diskPath.endsWith("/$SYSTEM_HIDDEN_DIRNAME")
183     }
184
185     private fun isSystemRel(rel: String): Boolean =
186         rel == "/$SYSTEM_HIDDEN_DIRNAME" || rel.startsWith("/$SYSTEM_HIDDEN_DIRNAME/")
187
188     private suspend fun ensureSystemDirExists() {
189         if (systemDirEnsured) return
190         val p = toDiskPath("/$SYSTEM_HIDDEN_DIRNAME")
191         if (guard { repo.getOrNull(p) }?.type == "dir") {
192             systemDirEnsured = true
193             return
194         }
195         guard { repo.createFolder(p) }
196         systemDirEnsured = true
197     }
198
199     private fun statsDiskPath(): String = toDiskPath("/$SYSTEM_HIDDEN_DIRNAME/
200 $STATS_FILENAME")
201
202     private suspend fun readPersistedStats(): YandexVaultPersistedStats? {
203         val meta = guard { repo.getOrNull(statsDiskPath()) } ?: return null
204         if (meta.type != "file") return null
205         return try {
206             guard {
207                 repo.openDownloadStream(statsDiskPath()).use {
208                     statsMapper.readValue(it, YandexVaultPersistedStats::class.java)
209                 }
210             }
211         } catch (_: Exception) {
212             null
213         }
214
215     private suspend fun writePersistedStatsInternal() {
216         ensureSystemDirExists()
217         val bytes = statsMapper.writeValueAsBytes(
218             YandexVaultPersistedStats(

```

```

219         totalBytes = _size.value ?: 0L,
220         fileCount = _numberOfFiles.value ?: 0,
221     ),
222 )
223 guard { repo.uploadBytes(statsDiskPath(), bytes, overwrite = true) }
224 }
225
226 private suspend fun persistStatsImmediate() {
227     statsPersistJob?.cancel()
228     statsPersistJob = null
229     writePersistedStatsInternal()
230 }
231
232 private fun scheduleStatsPersist() {
233     statsPersistJob?.cancel()
234     statsPersistJob = accessorScope.launch(ioDispatcher) {
235         delay(STATS_DEBOUNCE_MS)
236         try {
237             writePersistedStatsInternal()
238         } catch (e: CancellationException) {
239             throw e
240         } catch (e: YandexDiskAuthException) {
241             reportAuthFailure()
242             throw e
243         } catch (_: Exception) {
244             // Запись stats – best-effort; сетевые сбои не роняем процесс.
245         }
246     }
247 }
248
249 /**
250  * Сумма размеров и число пользовательских файлов в поддереве [relDir] (сама папка не
    считается файлом).
251  */
252 private suspend fun sumSubtreeStats(relDir: String): Pair<Int, Long> {
253     var fileCount = 0
254     var totalBytes = 0L
255     val queue = ArrayDeque<String>()
256     queue.add(relDir)
257     while (queue.isNotEmpty()) {
258         coroutineContext.ensureActive()
259         val rel = queue.removeFirst()
260         if (isSystemRel(rel)) continue
261         val (files, dirs) = listImmediateChildren(rel)
262         for (d in dirs) {

```

```

263         if (!isSystemRel(d.metaInfo.path)) queue.add(d.metaInfo.path)
264     }
265     for (f in files) {
266         if (!isSystemRel(f.metaInfo.path)) {
267             fileCount++
268             totalBytes += f.metaInfo.size
269         }
270     }
271 }
272 return fileCount to totalBytes
273 }
274
275 private suspend fun scanSizeAndNumOfFiles() {
276     var totalSize = 0L
277     var count = 0
278     val queue = ArrayDeque<String>()
279     queue.add("/")
280     while (queue.isNotEmpty()) {
281         val rel = queue.removeFirst()
282         if (isSystemRel(rel)) continue
283         val (files, dirs) = listImmediateChildren(rel)
284         for (d in dirs) {
285             if (!isSystemRel(d.metaInfo.path)) queue.add(d.metaInfo.path)
286         }
287         for (f in files) {
288             if (!isSystemRel(f.metaInfo.path)) {
289                 totalSize += f.metaInfo.size
290                 count++
291             }
292         }
293     }
294     _size.value = totalSize
295     _numberOfFiles.value = count
296 }
297
298 private suspend fun listImmediateChildren(relDir: String): Pair<List<IFile>,
List<IDirectory>> {
299     val diskPath = toDiskPath(relDir)
300     val files = mutableListOf<IFile>()
301     val dirs = mutableListOf<IDirectory>()
302     var offset = 0
303     while (true) {
304         coroutineContext.ensureActive()
305         val res = guard { repo.list(diskPath, API_LIST_LIMIT, offset) }
306         val items = res.embedded?.items.orEmpty()

```

```

307         for (it in items) {
308             if (isSystemDiskPath(it.path)) continue
309             val rel = toRelPath(it.path ?: continue)
310             if (isSystemRel(rel)) continue
311             when (it.type) {
312                 "file" -> files.add(it.toCommonFile(rel))
313                 "dir" -> dirs.add(it.toCommonDir(rel))
314             }
315         }
316         if (items.size < API_LIST_LIMIT) break
317         offset += items.size
318     }
319     return files to dirs
320 }
321
322 private suspend fun getMetadataAfterWrite(diskPath: String): ResourceDto {
323     val maxAttempts = 3
324     repeat(maxAttempts) { attempt ->
325         try {
326             return guard { repo.get(diskPath) }
327         } catch (e: FileNotFoundException) {
328             if (attempt == maxAttempts - 1) throw e
329             delay(200L * (attempt + 1))
330         }
331     }
332     error("unreachable")
333 }
334
335 private fun ResourceDto.toCommonFile(rel: String): CommonFile =
336     CommonFile(
337         CommonMetaInfo(
338             size = size ?: 0L,
339             isDeleted = boolProp(customProperties, PROP_DELETED),
340             isHidden = boolProp(customProperties, PROP_HIDDEN),
341             lastModified = modified ?: created ?: Instant.EPOCH,
342             path = rel,
343         ),
344     )
345
346 private fun ResourceDto.toCommonDir(rel: String): CommonDirectory =
347     CommonDirectory(
348         CommonMetaInfo(
349             size = 0L,
350             isDeleted = boolProp(customProperties, PROP_DELETED),
351             isHidden = boolProp(customProperties, PROP_HIDDEN),

```

```

352         lastModified = modified ?: created ?: Instant.EPOCH,
353         path = rel,
354     ),
355     elementsCount = null,
356 )
357
358 private fun boolProp(props: Map<String, Any?>?, key: String): Boolean {
359     val v = props?.get(key) ?: return false
360     return when (v) {
361         is Boolean -> v
362         is String -> v.equals("true", ignoreCase = true)
363         else -> v.toString().equals("true", ignoreCase = true)
364     }
365 }
366
367 override suspend fun getAllFiles(): List<IFile> = withContext(ioDispatcher) {
368     val out = mutableListOf<IFile>()
369     val queue = ArrayDeque<String>()
370     queue.add("/")
371     while (queue.isNotEmpty()) {
372         val rel = queue.removeFirst()
373         if (isSystemRel(rel)) continue
374         val (files, dirs) = listImmediateChildren(rel)
375         out.addAll(files.filter { !isSystemRel(it.metaInfo.path) })
376         for (d in dirs) {
377             if (!isSystemRel(d.metaInfo.path)) queue.add(d.metaInfo.path)
378         }
379     }
380     out
381 }
382
383 override suspend fun getFiles(path: String): List<IFile> = withContext(ioDispatcher) {
384     listImmediateChildren(path).first
385 }
386
387 override fun getFilesFlow(path: String): Flow<DataPage<IFile>> = flow {
388     val all = try {
389         withContext(ioDispatcher) { listImmediateChildren(path).first }
390     } catch (e: CancellationException) {
391         throw e
392     } catch (_: Exception) {
393         emit(filesFlowErrorPage())
394         return@flow
395     }
396     emitAllFilesPages(all)

```

```

397     }.flowOn(ioDispatcher)
398
399     override suspend fun getAllDirs(): List<IDirectory> = withContext(ioDispatcher) {
400         val out = mutableListOf<IDirectory>()
401         val queue = ArrayDeque<String>()
402         queue.add("/")
403         while (queue.isNotEmpty()) {
404             val rel = queue.removeFirst()
405             if (isSystemRel(rel)) continue
406             val (_, dirs) = listImmediateChildren(rel)
407             for (d in dirs) {
408                 if (!isSystemRel(d.metaInfo.path)) {
409                     out.add(d)
410                     queue.add(d.metaInfo.path)
411                 }
412             }
413         }
414         out
415     }
416
417     {
418         override suspend fun getDirs(path: String): List<IDirectory> = withContext(ioDispatcher) {
419             listImmediateChildren(path).second
420         }
421
422         override fun getDirsFlow(path: String): Flow<DataPage<IDirectory>> = flow {
423             val all = try {
424                 withContext(ioDispatcher) { listImmediateChildren(path).second }
425             } catch (e: CancellationException) {
426                 throw e
427             } catch (_: Exception) {
428                 emit(dirsFlowErrorPage())
429                 return@flow
430             }
431             emitAllDirsPages(all)
432         }.flowOn(ioDispatcher)
433
434         override suspend fun getFileInfo(path: String): IFile = withContext(ioDispatcher) {
435             val r = guard { repo.get(toDiskPath(path)) }
436             if (r.type != "file") throw WallencException.Storage.NotAFile()
437             r.toCommonFile(path)
438         }
439
440         override suspend fun getDirInfo(path: String): IDirectory = withContext(ioDispatcher) {
441             val r = guard { repo.get(toDiskPath(path)) }

```



```

441         if (r.type != "dir") throw WallencException.Storage.NotADirectory()
442         r.toCommonDir(path)
443     }
444
445     override suspend fun setHidden(path: String, hidden: Boolean) =
446     withContext(ioDispatcher) {
447         patchCustomProps(
448             path,
449             mapOf(PROP_HIDDEN to if (hidden) "true" else "false"),
450         )
451         val f = getFileInfo(path)
452         _filesUpdates.emit(
453             DataPage(listOf(f), pageLength = 1, pageIndex = 0),
454         )
455     }
456
457     override suspend fun touchFile(path: String): Unit = withContext(ioDispatcher) {
458         touchParentDirs(path)
459         var created = false
460         try {
461             guard { repo.uploadBytes(toDiskPath(path), ByteArray(0), overwrite = false) }
462             created = true
463         } catch (_: Exception) {
464             // файл уже есть — ок
465         }
466         if (created) {
467             _numberOfFiles.value = (_numberOfFiles.value ?: 0) + 1
468             persistStatsImmediate()
469             appendSyncEntry(path = path, operation = StorageSyncOperation.UPSERT)
470         }
471     }
472
473     override suspend fun touchDir(path: String): Unit = withContext(ioDispatcher) {
474         val segments = pathSegments(path)
475         var acc = ""
476         for (seg in segments) {
477             acc += "/$seg"
478             val diskPath = toDiskPath(acc)
479             when (guard { repo.getOrNull(diskPath) }?.type) {
480                 "dir" -> continue
481                 "file" -> throw WallencException.Storage.PathIsFile()
482                 else -> guard { repo.createFolder(diskPath) }
483             }
484         }
485     }

```

```

485
486     override suspend fun delete(path: String, recordSyncJournal: Boolean) =
withContext(ioDispatcher) {
487         if (path == "/" || path.isBlank()) {
488             throw WallencException.Storage.DeleteRootForbidden()
489         }
490         val diskPath = toDiskPath(path)
491         val prior = guard { repo.getOrNull(diskPath) }
492         if (prior != null) {
493             when (prior.type) {
494                 "file" -> {
495                     val sz = prior.size ?: 0L
496                     _size.value = ((_size.value ?: 0L) - sz).coerceAtLeast(0L)
497                     _numberOfFiles.value = ((_numberOfFiles.value ?: 0) -
1).coerceAtLeast(0)
498                 }
499                 "dir" -> {
500                     val (fc, total) = sumSubtreeStats(path)
501                     _size.value = ((_size.value ?: 0L) - total).coerceAtLeast(0L)
502                     _numberOfFiles.value = ((_numberOfFiles.value ?: 0) -
fc).coerceAtLeast(0)
503                 }
504             }
505         }
506         guard { repo.delete(diskPath, permanently = true) }
507         scheduleStatsPersist()
508         if (recordSyncJournal) {
509             appendSyncEntry(path = path, operation = StorageSyncOperation.DELETE)
510         }
511     }
512
513     override suspend fun openWrite(path: String, recordSyncJournal: Boolean): OutputStream =
withContext(ioDispatcher) {
514         touchParentDirs(path)
515         val tmp = File.createTempFile("wallenc-yandex-", ".upload")
516         val fos = FileOutputStream(tmp)
517         fos.onClosed {
518             runCommitAfterStreamClosed {
519                 commitUploadedFile(path, tmp, recordSyncJournal)
520             }
521         }
522     }
523
524     override suspend fun openRead(path: String): InputStream = withContext(ioDispatcher) {
525         guard { repo.openDownloadStream(toDiskPath(path)) }
526     }

```

```

527
528     override suspend fun moveToTrash(path: String, recordSyncJournal: Boolean) =
withContext(ioDispatcher) {
529         patchCustomProps(path, mapOf(PROP_DELETED to "true"))
530         if (recordSyncJournal) {
531             appendSyncEntry(path = path, operation = StorageSyncOperation.TRASH)
532         }
533     }
534
535     override suspend fun openReadSystemFile(name: String): InputStream =
withContext(ioDispatcher) {
536         ensureSystemDirExists()
537         val rel = "/$SYSTEM_HIDDEN_DIRNAME/$name"
538         val diskPath = toDiskPath(rel)
539         when (guard { repo.getOrNull(diskPath) }?.type) {
540             "file" -> guard { repo.openDownloadStream(diskPath) }
541             null -> throw WallencException.Storage.FileNotFound()
542             else -> throw WallencException.Storage.FileNotFound()
543         }
544     }
545
546     override suspend fun openWriteSystemFile(name: String): OutputStream =
withContext(ioDispatcher) {
547         ensureSystemDirExists()
548         val rel = "/$SYSTEM_HIDDEN_DIRNAME/$name"
549         val uploadBuffer = ByteArrayOutputStream()
550         uploadBuffer.onClosed {
551             val bytes = uploadBuffer.toByteArray()
552             val diskPath = toDiskPath(rel)
553             runCommitAfterStreamClosed {
554                 guard { repo.uploadBytes(diskPath, bytes, overwrite = true) }
555             }
556         }
557     }
558
559     override suspend fun readSyncJournal(): StorageSyncJournal = withContext(ioDispatcher) {
560         journalBuffer.flushPending()
561         readSyncJournalUnchecked()
562     }
563
564     override suspend fun flushPendingSyncJournal() = withContext(ioDispatcher) {
565         journalBuffer.flushPending()
566     }
567
568     override suspend fun putSyncJournalEntries(entries: StorageSyncJournal) =
withContext(ioDispatcher) {

```

```

569         journalBuffer.putEntries(entries)
570     }
571
572     private suspend fun readSyncJournalUnchecked(): StorageSyncJournal {
573         val bytes = readSystemFileBytesOrEmpty { openReadSystemFile(SYNC_JOURNAL_FILENAME) }
574         return StorageSyncJournalCodec.read(statsMapper, bytes)
575     }
576
577     private suspend fun writeSyncJournal(journal: StorageSyncJournal) {
578         openWriteSystemFile(SYNC_JOURNAL_FILENAME).use { out ->
579             statsMapper.writeValue(out, journal)
580         }
581     }
582
583     override suspend fun readSyncLock(): StorageSyncLock? = withContext(ioDispatcher) {
584         val bytes = readSystemFileBytesOrEmpty { openReadSystemFile(SYNC_LOCK_FILENAME) }
585         if (bytes.isEmpty()) {
586             return@withContext null
587         }
588         return@withContext runCatching {
589             statsMapper.readValue(bytes, StorageSyncLock::class.java)
590         }.getOrNull()
591     }
592
593     /**
594      * Best-effort lock на Диске (read-modify-write без CAS на стороне API).
595      * Межустройственная координация опирается на [StorageSyncEngine] mutex в процессе.
596      */
597     override suspend fun tryAcquireSyncLock(holderId: String, leaseUntil: Instant): Boolean
598     = withContext(ioDispatcher) {
599         repeat(SYNC_LOCK_CAS_RETRIES) { attempt ->
600             val acquired = tryAcquireSyncLockOnce(holderId, leaseUntil)
601             if (acquired) {
602                 return@withContext true
603             }
604             if (attempt < SYNC_LOCK_CAS_RETRIES - 1) {
605                 delay(SYNC_LOCK_CAS_DELAY_MS)
606             }
607         }
608         return@withContext false
609     }
610
611     private suspend fun tryAcquireSyncLockOnce(holderId: String, leaseUntil: Instant):
612     Boolean {
613         return syncLockMutex.withLock {

```

```

612         val current = readSyncLock()
613         val now = Instant.now()
614         val foreignLockActive = current != null &&
615             current.holderId != holderId &&
616             current.leaseUntil.isAfter(now) &&
617             current.updatedAt.plusSeconds(SYNC_LOCK_STALE_TIMEOUT_SECONDS).isAfter(now)
618         if (foreignLockActive) {
619             return@withLock false
620         }
621         val next = StorageSyncLock(
622             holderId = holderId,
623             leaseUntil = leaseUntil,
624             updatedAt = now,
625         )
626         openWriteSystemFile(SYNC_LOCK_FILENAME).use { out ->
627             statsMapper.writeValue(out, next)
628         }
629         readSyncLock()?.holderId == holderId
630     }
631 }
632
633 override suspend fun releaseSyncLock(holderId: String) = withContext(ioDispatcher) {
634     syncLockMutex.withLock {
635         val current = readSyncLock() ?: return@withLock
636         if (current.holderId != holderId) {
637             return@withLock
638         }
639         openWriteSystemFile(SYNC_LOCK_FILENAME).use { out ->
640             out.write(ByteArray(0))
641         }
642     }
643 }
644
645 override suspend fun forceClearSyncLock() = withContext(ioDispatcher) {
646     syncLockMutex.withLock {
647         openWriteSystemFile(SYNC_LOCK_FILENAME).use { out ->
648             out.write(ByteArray(0))
649         }
650     }
651 }
652
653 /**
654  * Выполняется из [OutputStream.close]; ошибки upload пробрасываются вызывающему коду.
655  */
656 private fun runCommitAfterStreamClosed(block: suspend () -> Unit) {

```

```

657         runBlocking {
658             withContext(ioDispatcher) {
659                 block()
660             }
661         }
662     }
663
664     private suspend fun commitUploadedFile(path: String, tmp: File, recordSyncJournal:
Boolean) {
665         try {
666             val diskPath = toDiskPath(path)
667             val prior = guard { repo.getOrNull(diskPath) }
668             if (prior?.type == "dir") {
669                 throw WallencException.Storage.CannotWriteOverDirectory()
670             }
671             val hadFile = prior?.type == "file"
672             val priorSize = if (prior?.type == "file") prior.size ?: 0L else 0L
673             guard { repo.uploadFile(diskPath, tmp, overwrite = true) }
674             val after = guard { getMetadataAfterWrite(diskPath) }
675             if (after.type != "file") {
676                 throw WallencException.Storage.UnexpectedState()
677             }
678             val newSize = after.size ?: 0L
679             _size.value = ((_size.value ?: 0L) + newSize - priorSize).coerceAtLeast(0L)
680             if (!hadFile) {
681                 _numberOfFiles.value = (_numberOfFiles.value ?: 0) + 1
682             }
683             persistStatsImmediate()
684             val info = runCatching { after.toCommonFile(path) }.getOrNull()
685             info?.let {
686                 _filesUpdates.emit(DataPage(listOf(it), pageLength = 1, pageIndex = 0))
687             }
688             if (recordSyncJournal) {
689                 appendSyncEntry(path = path, operation = StorageSyncOperation.UPSERT)
690             }
691         } finally {
692             tmp.delete()
693         }
694     }
695
696     private suspend fun appendSyncEntry(path: String, operation: StorageSyncOperation) {
697         val cleanedPath = StorageSyncPaths.normalize(path)
698         if (!StorageSyncPaths.isSyncableUserPath(cleanedPath)) {
699             return
700         }

```

```

701         val sequence = journalBuffer.nextSequence()
702         val entry = journalBuffer.buildEntry(cleanedPath, operation, sequence)
703         journalBuffer.appendEntry(cleanedPath, entry)
704     }
705
706     private suspend fun touchParentDirs(path: String) {
707         val normalized = path.removeSuffix("/")
708         if (normalized == "/" || normalized.isBlank()) return
709         val lastSlash = normalized.lastIndexOf('/')
710         if (lastSlash <= 0) return
711         val parent = normalized.substring(0, lastSlash).ifEmpty { "/" }
712         touchDir(parent)
713     }
714
715     private fun pathSegments(path: String): List<String> =
716         path.trim('/').split('/').filter { it.isNotBlank() }
717
718     /** PATCH только переданных ключей – API Диска дополняет [custom_properties], без
719     предварительного GET. */
720     private suspend fun patchCustomProps(path: String, props: Map<String, String>) {
721         guard { repo.setCustomProperties(toDiskPath(path), props) }
722     }
723
724     private suspend fun FlowCollector<DataPage<IFile>>.emitAllFilesPages(all: List<IFile>) {
725         var pageIndex = 0
726         var i = 0
727         while (i < all.size) {
728             val chunk = all.subList(i, kotlin.math.min(i + DATA_PAGE_LENGTH,
729 all.size)).toList()
730             emit(
731                 DataPage(
732                     list = chunk,
733                     isLoading = false,
734                     isError = false,
735                     hasNext = i + DATA_PAGE_LENGTH < all.size,
736                     pageLength = DATA_PAGE_LENGTH,
737                     pageIndex = pageIndex++,
738                 ),
739             )
740             i += DATA_PAGE_LENGTH
741         }
742         if (all.isEmpty()) {
743             emit(
744                 DataPage(
745                     list = emptyList(),
746                     isLoading = false,

```

```

745             isError = false,
746             hasNext = false,
747             pageLength = DATA_PAGE_LENGTH,
748             pageIndex = 0,
749         ),
750     )
751 }
752 }
753
754 private suspend fun FlowCollector<DataPage<IDirectory>>.emitAllDirsPages(all:
List<IDirectory>) {
755     var pageIndex = 0
756     var i = 0
757     while (i < all.size) {
758         val chunk = all.subList(i, kotlin.math.min(i + DATA_PAGE_LENGTH,
all.size)).toList()
759         emit(
760             DataPage(
761                 list = chunk,
762                 isLoading = false,
763                 isError = false,
764                 hasNext = i + DATA_PAGE_LENGTH < all.size,
765                 pageLength = DATA_PAGE_LENGTH,
766                 pageIndex = pageIndex++,
767             ),
768         )
769         i += DATA_PAGE_LENGTH
770     }
771     if (all.isEmpty()) {
772         emit(
773             DataPage(
774                 list = emptyList(),
775                 isLoading = false,
776                 isError = false,
777                 hasNext = false,
778                 pageLength = DATA_PAGE_LENGTH,
779                 pageIndex = 0,
780             ),
781         )
782     }
783 }
784
785 private fun filesFlowErrorPage(): DataPage<IFile> =
786     DataPage(
787         list = emptyList(),

```



```

788         isLoading = false,
789         isError = true,
790         hasNext = false,
791         pageLength = DATA_PAGE_LENGTH,
792         pageIndex = 0,
793     )
794
795     private fun dirsFlowErrorPage(): DataPage<IDirectory> =
796         DataPage(
797             list = emptyList(),
798             isLoading = false,
799             isError = true,
800             hasNext = false,
801             pageLength = DATA_PAGE_LENGTH,
802             pageIndex = 0,
803         )
804
805     companion object {
806         private val statsMapper = jacksonObjectMapper().apply { findAndRegisterModules() }
807
808         private const val SYSTEM_HIDDEN_DIRNAME = "wallenc-yandex-system"
809         private const val STATS_FILENAME = "yandex-vault-stats.json"
810         private const val SYNC_JOURNAL_FILENAME = "sync-journal.json"
811         private const val SYNC_LOCK_FILENAME = "sync-lock.json"
812         private const val SYNC_LOCK_STALE_TIMEOUT_SECONDS: Long = 60 * 60
813         private const val SYNC_LOCK_CAS_RETRIES = 3
814         private const val SYNC_LOCK_CAS_DELAY_MS = 80L
815         private const val STATS_DEBOUNCE_MS = 450L
816         private const val DATA_PAGE_LENGTH = 10
817         private const val API_LIST_LIMIT = 1000
818         private const val PROP_HIDDEN = "wallenc.hidden"
819         private const val PROP_DELETED = "wallenc.deleted"
820     }
821 }
822

```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/utils/CloseHandledStream.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.utils
2
3 import java.io.InputStream
4 import java.io.OutputStream
5
6 private class CloseHandledOutputStream(
7     private val stream: OutputStream,
8     private val onClosing: () -> Unit,
9     private val onClose: () -> Unit
10 ) : OutputStream() {
11
12     override fun write(b: Int) {
13         stream.write(b)
14     }
15
16     override fun write(b: ByteArray) {
17         stream.write(b)
18     }
19
20     override fun write(b: ByteArray, off: Int, len: Int) {
21         stream.write(b, off, len)
22     }
23
24     override fun flush() {
25         stream.flush()
26     }
27
28     override fun close() {
29         onClosing()
30         var streamFailure: Throwable? = null
31         try {
32             stream.close()
33         } catch (t: Throwable) {
34             streamFailure = t
35         }
36         try {
37             onClose()
38         } catch (afterCloseFailure: Throwable) {
39             streamFailure?.let { afterCloseFailure.addSuppressed(it) }
40             throw afterCloseFailure
41         }
42         streamFailure?.let { throw it }
```

```

43     }
44 }
45
46 private class CloseHandledInputStream(
47     private val stream: InputStream,
48     private val onClosing: () -> Unit,
49     private val onClose: () -> Unit
50 ) : InputStream() {
51
52     override fun read(): Int {
53         return stream.read()
54     }
55
56     override fun read(b: ByteArray): Int {
57         return stream.read(b)
58     }
59
60     override fun read(b: ByteArray, off: Int, len: Int): Int {
61         return stream.read(b, off, len)
62     }
63
64     override fun skip(n: Long): Long {
65         return stream.skip(n)
66     }
67
68     override fun available(): Int {
69         return stream.available()
70     }
71
72     override fun close() {
73         onClosing()
74         try {
75             stream.close()
76         } finally {
77             onClose()
78         }
79     }
80
81     override fun mark(readlimit: Int) {
82         stream.mark(readlimit)
83     }
84
85     override fun reset() {
86         stream.reset()
87     }

```

```
88
89     override fun markSupported(): Boolean {
90         return stream.markSupported()
91     }
92 }
93
94 class CloseHandledStreamExtension {
95     companion object {
96         fun OutputStream.onClosed(callback: ()->Unit): OutputStream {
97             return CloseHandledOutputStream(this, {}, callback)
98         }
99
100        fun OutputStream.onClosing(callback: ()->Unit): OutputStream {
101            return CloseHandledOutputStream(this, callback) {}
102        }
103    }
104 }
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/Utils/IProvider.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.Utils
2
3 interface IProvider<T> {
4     suspend fun get(): T?
5     suspend fun set(value: T)
6     suspend fun clear()
7 }
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/vaults/VaultsManager.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.vaults
2
3 import com.github.nullptroma.wallenc.domain.vault.model.YandexAccount
4 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository.YandexDiskRepository
5 import com.github.nullptroma.wallenc.domain.vault.network.yandexuserinfo.repository.YandexUserInfoRepository
6 import com.github.nullptroma.wallenc.domain.vault.ports.StorageKeyMapStore
7 import com.github.nullptroma.wallenc.domain.vault.ports.YandexAccountStore
8 import com.github.nullptroma.wallenc.domain.vault.storages.UnlockManager
9 import com.github.nullptroma.wallenc.domain.vault.vaults.yandex.YandexRegistration
10 import com.github.nullptroma.wallenc.domain.vault.vaults.yandex.YandexVault
11 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
12 import com.github.nullptroma.wallenc.domain.interfaces.IUnlockManager
13 import com.github.nullptroma.wallenc.domain.interfaces.IVault
14 import com.github.nullptroma.wallenc.domain.interfaces.IVaultsManager
15 import com.github.nullptroma.wallenc.vault.contract.VaultRegistrar
16 import com.github.nullptroma.wallenc.vault.contract.VaultRegistration
17 import kotlinx.coroutines.CoroutineDispatcher
18 import kotlinx.coroutines.CoroutineExceptionHandler
19 import kotlinx.coroutines.CoroutineScope
20 import kotlinx.coroutines.ExperimentalCoroutinesApi
21 import kotlinx.coroutines.SupervisorJob
22 import kotlinx.coroutines.flow.SharingStarted
23 import kotlinx.coroutines.flow.StateFlow
24 import kotlinx.coroutines.flow.combine
25 import kotlinx.coroutines.flow.flatMapLatest
26 import kotlinx.coroutines.flow.flowOf
27 import kotlinx.coroutines.flow.map
28 import kotlinx.coroutines.flow.stateIn
29 import kotlinx.coroutines.withContext
30 import java.util.UUID
31
32 @OptIn(ExperimentalCoroutinesApi::class)
33 class VaultsManager(
34     private val ioDispatcher: CoroutineDispatcher,
35     private val localVault: IVault,
36     keyRepo: StorageKeyMapStore,
37     private val yandexAccountStore: YandexAccountStore,
38     private val yandexUserInfoRepository: YandexUserInfoRepository,
39     private val yandexDiskRepositoryFactory: YandexDiskRepositoryFactory,
40 ) : IVaultsManager, VaultRegistrar {
41
```

```

42     private val scope = CoroutineScope(
43         SupervisorJob() +
44         ioDispatcher +
45         CoroutineExceptionHandler { _, throwable ->
46             System.err.println("VaultsManager: uncaught coroutine failure:
47             ${throwable.message}")
48             throwable.printStackTrace()
49         },
50     )
51     private val yandexVaults: StateFlow<List<IVault>> = yandexAccountStore.observeAll()
52         .map { rows ->
53             rows.map { row ->
54                 val vaultUuid = UUID.fromString(row.vaultUuid)
55                 YandexVault(
56                     uuid = vaultUuid,
57                     accountEmail = row.email,
58                     repo = yandexDiskRepositoryFactory.create(vaultUuid),
59                     ioDispatcher = ioDispatcher,
60                     parentScope = scope,
61                 )
62             }
63         }
64         .stateIn(scope, SharingStarted.Eagerly, emptyList())
65
66     override val vaults: StateFlow<List<IVault>> = yandexVaults
67         .map { remote -> listOf(localVault) + remote }
68         .stateIn(scope, SharingStarted.Eagerly, listOf(localVault))
69
70     override val allStorages: StateFlow<List<IStorage>> = vaults
71         .flatMapLatest { vs ->
72             if (vs.isEmpty()) flowOf(emptyList())
73             else combine(vs.map { it.storages }) { arr -> arr.toList().flatten() }
74         }
75         .stateIn(scope, SharingStarted.Eagerly, emptyList())
76
77     override val unlockManager: IUnlockManager = UnlockManager(
78         keymapRepository = keyRepo,
79         ioDispatcher = ioDispatcher,
80         vaultsManager = this,
81     )
82
83     override suspend fun register(registration: VaultRegistration) =
84     withContext(ioDispatcher) {
85         when (registration) {

```

```

85         is YandexRegistration -> registerYandex(registration)
86     else -> throw IllegalArgumentException(
87         "Unknown VaultRegistration type: ${registration::class.qualifiedName}",
88     )
89 }
90 }
91
92 override suspend fun unregister(vaultUuid: UUID): Unit = withContext(ioDispatcher) {
93     yandexAccountStore.deleteByVaultUuid(vaultUuid.toString())
94 }
95
96 override suspend fun retry(vaultUuid: UUID): Boolean = withContext(ioDispatcher) {
97     val account = yandexAccountStore.getByVaultUuid(vaultUuid.toString()) ?:
return@withContext false
98     yandexAccountStore.updateCredentials(
99         vaultUuid = account.vaultUuid,
100         email = account.email,
101         token = account.oauthToken,
102     )
103     true
104 }
105
106 private suspend fun registerYandex(registration: YandexRegistration) {
107     val token = registration.oauthToken
108     val info = yandexUserInfoRepository.userInfo(token)
109     val email = info.defaultEmail?.takeIf { it.isNotBlank() }
110         ?: "${info.login}@yandex.ru"
111     val existing = yandexAccountStore.getByYandexUserId(info.id)
112     val vaultUuid = existing?.vaultUuid ?: UUID.randomUUID().toString()
113     if (existing != null) {
114         yandexAccountStore.updateCredentials(vaultUuid, email, token)
115     } else {
116         yandexAccountStore.insert(
117             YandexAccount(
118                 vaultUuid = vaultUuid,
119                 yandexUserId = info.id,
120                 email = email,
121                 oauthToken = token,
122             ),
123         )
124     }
125 }
126 }
127

```


Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/vaults/local/LocalVault.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.vaults.local
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
6 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
7 import com.github.nullptroma.wallenc.domain.vault.storages.local.LocalStorage
8 import com.github.nullptroma.wallenc.vault.contract.DescribedVault
9 import com.github.nullptroma.wallenc.vault.contract.VaultDescriptor
10 import kotlinx.coroutines.CoroutineDispatcher
11 import kotlinx.coroutines.CoroutineScope
12 import kotlinx.coroutines.flow.MutableStateFlow
13 import kotlinx.coroutines.flow.StateFlow
14 import kotlinx.coroutines.launch
15 import kotlinx.coroutines.withContext
16 import java.io.File
17 import java.io.FileOutputStream
18 import java.util.UUID
19 import kotlin.io.path.Path
20 import kotlin.io.path.createDirectory
21 import kotlin.io.path.pathString
22
23 class LocalVault(
24     private val ioDispatcher: CoroutineDispatcher,
25     vaultRoot: File?,
26 ) : DescribedVault {
27
28     override val uuid: UUID = vaultRoot?.let { root ->
29         root.mkdirs()
30         readOrCreateVaultUuid(File(root, UUID_FILE_NAME))
31     } ?: UUID.randomUUID()
32
33     override val descriptor: VaultDescriptor = VaultDescriptor.LocalDevice(uuid)
34
35     private val _storages = MutableStateFlow<List<IStorage>>(emptyList())
36     override val storages: StateFlow<List<IStorage>> = _storages
37
38     private val _storagesScanInProgress = MutableStateFlow(false)
39     override val storagesScanInProgress: StateFlow<Boolean> = _storagesScanInProgress
40
41     private val _isAvailable = MutableStateFlow(false)
42     override val isAvailable: StateFlow<Boolean> = _isAvailable
```

```

43
44     private val _totalSpace = MutableStateFlow<Long?>(null)
45     override val totalSpace: StateFlow<Long?> = _totalSpace
46
47     private val _availableSpace = MutableStateFlow<Long?>(null)
48     override val availableSpace: StateFlow<Long?> = _availableSpace
49
50     private val path = MutableStateFlow(vaultRoot)
51
52     init {
53         CoroutineScope(ioDispatcher).launch {
54             _storagesScanInProgress.value = true
55             try {
56                 _isAvailable.value = path.value != null
57                 if (path.value != null) {
58                     readStorages()
59                 }
60             } finally {
61                 _storagesScanInProgress.value = false
62             }
63         }
64     }
65
66     private suspend fun readStorages() {
67         val path = path.value
68         if (path == null || !_isAvailable.value) {
69             throw WallencException.Storage.NotAvailable()
70         }
71
72         val dirs = path.listFiles()?.filter { it.isDirectory }
73         if (dirs != null) {
74             _storages.value = dirs.map {
75                 val storageUuid = UUID.fromString(it.name)
76                 LocalStorage(storageUuid, it.path, ioDispatcher).apply { init() }
77             }
78         }
79     }
80
81     override suspend fun createStorage(): LocalStorage = withContext(ioDispatcher) {
82         val path = path.value
83         if (path == null || !_isAvailable.value) {
84             throw WallencException.Storage.NotAvailable()
85         }
86
87         val storageUuid = UUID.randomUUID()

```

```

88         val next = Path(path.path, storageUuid.toString())
89         next.createDirectory()
90         val newStorage = LocalStorage(storageUuid, next.pathString, ioDispatcher)
91         newStorage.init()
92         _storages.value = _storages.value.toMutableList().apply {
93             add(newStorage)
94         }
95         return@withContext newStorage
96     }
97
98     override suspend fun createStorage(
99         enc: StorageEncryptionInfo,
100     ): LocalStorage = withContext(ioDispatcher) {
101         val storage = createStorage()
102         storage.setEncInfo(enc)
103         return@withContext storage
104     }
105
106     override suspend fun rescanStorages() = withContext(ioDispatcher) {
107         _storagesScanInProgress.value = true
108         try {
109             if (_isAvailable.value) {
110                 readStorages()
111             }
112         } finally {
113             _storagesScanInProgress.value = false
114         }
115     }
116
117     override suspend fun remove(storage: IStorage) = withContext(ioDispatcher) {
118         val path = path.value
119         if (path == null || !_isAvailable.value) {
120             throw WallencException.Storage.NotAvailable()
121         }
122
123         val curStorages = _storages.value.toMutableList()
124         val index = curStorages.indexOfFirst { it.uuid == storage.uuid }
125         if (index != -1) {
126             val localStorage = curStorages[index] as LocalStorage
127             curStorages.removeAt(index)
128             _storages.value = curStorages
129             File(localStorage.absolutePath).deleteRecursively()
130         }
131     }
132

```

```

133     private companion object {
134         const val UUID_FILE_NAME = ".uuid"
135
136         private val uuidLock = Any()
137
138         private fun readOrCreateVaultUuid(idFile: File): UUID = synchronized(uuidLock) {
139             if (idFile.exists()) {
140                 idFile.bufferedReader().use { reader ->
141                     val line = reader.readLine()?.trim()
142                     if (!line.isNullOrEmpty()) {
143                         runCatching { UUID.fromString(line) }.getOrNull()?.let
144 { return@synchronized it }
145                     }
146                 }
147             }
148             val generated = UUID.randomUUID()
149             val parent = idFile.parentFile ?: throw IllegalStateException("No parent for
150 $idFile")
151             parent.mkdirs()
152             val tmp = File.createTempFile("vault-uuid-", ".tmp", parent)
153             try {
154                 FileOutputStream(tmp).use { fos ->
155                     fos.write(generated.toString().toByteArray(Charsets.UTF_8))
156                     fos.fd.sync()
157                 }
158                 if (!tmp.renameTo(idFile)) {
159                     tmp.copyTo(idFile, overwrite = true)
160                 }
161             } finally {
162                 if (tmp.exists()) {
163                     tmp.delete()
164                 }
165             }
166             return@synchronized generated
167         }
168     }

```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/vaults/yandex/YandexRegistration.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.vaults.yandex
2
3 import com.github.nullptroma.wallenc.vault.contract.VaultRegistration
4
5 /**
6  * Регистрация удалённого хранилища Яндекс.Диска по результату OAuth.
7  *
8  * Живёт в `:data` (а не в `:vault-api`), потому что [VaultRegistration]
9  * намеренно не sealed — конкретные реализации лежат рядом со своим поставщиком.
10  * Слой presentation не раскрывает этот тип, только передаёт его в
11  * `VaultRegistrar.register(...)`.
```

Исходный файл domain-vault/src/main/java/com/github/nullptroma/wallenc/
domain/vault/vaults/yandex/YandexVault.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.vaults.yandex
2
3 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskAuthException
4 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository.YandexDiskRepository
5 import com.github.nullptroma.wallenc.domain.vault.storages.yandex.YandexStorage
6 import com.github.nullptroma.wallenc.domain.datatypes.StorageEncryptionInfo
7 import com.github.nullptroma.wallenc.domain.datatypes.StorageMetaLoadState
8 import com.github.nullptroma.wallenc.domain.interfaces.IStorage
9 import com.github.nullptroma.wallenc.vault.contract.CloudBrand
10 import com.github.nullptroma.wallenc.vault.contract.DescribedVault
11 import com.github.nullptroma.wallenc.vault.contract.VaultDescriptor
12 import kotlinx.coroutines.CoroutineDispatcher
13 import kotlinx.coroutines.CoroutineScope
14 import kotlinx.coroutines.async
15 import kotlinx.coroutines.awaitAll
16 import kotlinx.coroutines.coroutineScope
17 import kotlinx.coroutines.delay
18 import kotlinx.coroutines.flow.MutableStateFlow
19 import kotlinx.coroutines.flow.StateFlow
20 import kotlinx.coroutines.launch
21 import kotlinx.coroutines.sync.Mutex
22 import kotlinx.coroutines.sync.withLock
23 import kotlinx.coroutines.withContext
24 import java.util.UUID
25
26 /**
27  * Удалённый vault Яндекс.Диска: папка приложения `app:/`, внутри — подпапки-UUID как
28  * [YandexStorage].
29  *
30  * [isAvailable] — успешный контакт с Disk API (метаданные диска и загрузка списка
31  * storages).
32  * Пока false, дочерние storages тоже считаются недоступными (см. YandexStorageAccessor).
33  */
34 class YandexVault(
35     override val uuid: UUID,
36     accountEmail: String,
37     private val repo: YandexDiskRepository,
38     private val ioDispatcher: CoroutineDispatcher,
39     private val parentScope: CoroutineScope,
40 ) : DescribedVault {
41
42     override val descriptor: VaultDescriptor = VaultDescriptor.LinkedRemote(
```

```

41         uuid = uuid,
42         brand = CloudBrand.YANDEX,
43         accountDisplayName = accountEmail,
44     )
45
46     private val _vaultReachable = MutableStateFlow(false)
47     override val isAvailable: StateFlow<Boolean> = _vaultReachable
48
49     private val _storages = MutableStateFlow<List<IStorage>>(emptyList())
50     override val storages: StateFlow<List<IStorage>> = _storages
51
52     private val _storagesScanInProgress = MutableStateFlow(false)
53     override val storagesScanInProgress: StateFlow<Boolean> = _storagesScanInProgress
54
55     private val _totalSpace = MutableStateFlow<Long?>(null)
56     override val totalSpace: StateFlow<Long?> = _totalSpace
57
58     private val _availableSpace = MutableStateFlow<Long?>(null)
59     override val availableSpace: StateFlow<Long?> = _availableSpace
60
61     private val refreshMutex = Mutex()
62
63     init {
64         parentScope.launch {
65             runCatching { refreshFromDisk() }
66         }
67     }
68
69     override suspend fun rescanStorages() {
70         refreshMutex.withLock {
71             refreshFromDisk()
72         }
73     }
74
75     private suspend fun refreshFromDisk() {
76         _storagesScanInProgress.value = true
77         _vaultReachable.value = false
78         try {
79             val info = repo.diskInfo()
80             _totalSpace.value = info.totalSpace
81             val used = info.usedSpace ?: 0L
82             val total = info.totalSpace ?: 0L
83             _availableSpace.value = (total - used).coerceAtLeast(0L)
84             _vaultReachable.value = true
85             _storages.value = loadStoragesList()

```

```

86         } catch (_: YandexDiskAuthException) {
87             _vaultReachable.value = false
88             _storages.value = emptyList()
89         } catch (_: Exception) {
90             _vaultReachable.value = false
91             _storages.value = emptyList()
92         } finally {
93             _storagesScanInProgress.value = false
94         }
95     }
96
97     private suspend fun loadStoragesList(): List<IStorage> {
98         val pending = mutableList<YandexStorage>()
99         var offset = 0
100        while (true) {
101            val root = repo.list("app:/", APP_LIST_LIMIT, offset)
102            val items = root.embedded?.items.orEmpty()
103            for (item in items) {
104                if (item.type != "dir") continue
105                val name = item.name ?: continue
106                val storageUuid = runCatching { UUID.fromString(name) }.getOrNull() ?:
continue
107                    pending.add(
108                        YandexStorage(
109                            uuid = storageUuid,
110                            repo = repo,
111                            vaultAvailability = _vaultReachable,
112                            ioDispatcher = ioDispatcher,
113                            accessorScope = parentScope,
114                            reportAuthFailure = {
115                                parentScope.launch { _vaultReachable.value = false }
116                            },
117                        ),
118                    )
119            }
120            if (items.size < APP_LIST_LIMIT) break
121            offset += items.size
122        }
123        if (pending.isEmpty()) return emptyList()
124        return coroutineScope {
125            pending.map { storage ->
126                async(ioDispatcher) { initStorageWithRetry(storage) }
127            }.awaitAll().filterNotNull()
128        }
129    }

```



```

130
131     private suspend fun initStorageWithRetry(storage: YandexStorage): YandexStorage? {
132         for (attempt in 0 until STORAGE_INIT_ATTEMPTS) {
133             if (attempt > 0) {
134                 delay(STORAGE_INIT_RETRY_DELAY_MS * attempt)
135             }
136             if (
137                 runCatching { storage.init() }.isSuccess &&
138                 storage.metaLoadState.value == StorageMetaLoadState.Ready
139             ) {
140                 return storage
141             }
142         }
143         return null
144     }
145
146     override suspend fun createStorage(): IStorage = withContext(ioDispatcher) {
147         val id = UUID.randomUUID()
148         repo.createFolder("app:/$id")
149         val storage = YandexStorage(
150             uuid = id,
151             repo = repo,
152             vaultAvailability = _vaultReachable,
153             ioDispatcher = ioDispatcher,
154             accessorScope = parentScope,
155             reportAuthFailure = {
156                 parentScope.launch { _vaultReachable.value = false }
157             },
158         )
159         storage.init()
160         _storages.value += storage
161         storage
162     }
163
164     override suspend fun createStorage(enc: StorageEncryptionInfo): IStorage {
165         val storage = createStorage()
166         storage.setEncInfo(enc)
167         return storage
168     }
169
170     override suspend fun remove(storage: IStorage) = withContext(ioDispatcher) {
171         if (storage !is YandexStorage) return@withContext
172         repo.delete("app:/${storage.uuid}", permanently = true)
173         _storages.value = _storages.value.filter { it.uuid != storage.uuid }
174     }

```

```
175
176     private companion object {
177         private const val APP_LIST_LIMIT = 1000
178         private const val STORAGE_INIT_ATTEMPTS = 3
179         private const val STORAGE_INIT_RETRY_DELAY_MS = 400L
180     }
181 }
182
```

Исходный файл domain-vault/src/test/java/com/github/nullptroma/wallenc/
domain/vault/errors/VaultThrowableMappingTest.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.errors
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskAuthException
5 import org.junit.Assert.assertEquals
6 import org.junit.Assert.assertTrue
7 import org.junit.Test
8 import okhttp3.ResponseBody.Companion.toResponseBody
9 import retrofit2.HttpException
10 import retrofit2.Response
11 import java.io.FileNotFoundException
12 import java.io.IOException
13 import java.net.SocketTimeoutException
14
15 class VaultThrowableMappingTest {
16
17     @Test
18     fun mapsYandexDiskAuthToAuthFailed() {
19         val mapped = YandexDiskAuthException("unauthorized").toVaultWallencException()
20         assertTrue(mapped is WallencException.Auth.Failed)
21     }
22
23     @Test
24     fun mapsHttpExceptionToNetworkHttpFailed() {
25         val response = Response.error<String>(401, "").toResponseBody(null)
26         val mapped = HttpException(response).toVaultWallencException()
27         assertTrue(mapped is WallencException.Network.HttpFailed)
28         assertEquals(401, (mapped as WallencException.Network.HttpFailed).statusCode)
29     }
30
31     @Test
32     fun mapsMissingOAuthTokenToTokenMissing() {
33         val mapped = IOException("Yandex OAuth token is missing").toVaultWallencException()
34         assertTrue(mapped is WallencException.Auth.TokenMissing)
35     }
36
37     @Test
38     fun mapsSocketTimeoutToOperationTimedOut() {
39         val mapped = SocketTimeoutException("timeout").toVaultWallencException()
40         assertTrue(mapped is WallencException.Network.OperationTimedOut)
41     }
42
```

```
43     @Test
44     fun mapsFileNotFoundToStorageFileNotFound() {
45         val mapped = FileNotFoundException("x").toVaultWallencException()
46         assertTrue(mapped is WallencException.Storage.FileNotFound)
47     }
48
49     @Test
50     fun mapsIllegalStateNotAFile() {
51         val mapped = IllegalStateException("Not a file").toVaultWallencException()
52         assertTrue(mapped is WallencException.Storage.NotAFile)
53     }
54 }
55
```

Исходный файл domain-vault/src/test/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/YandexDiskRepositoryTestFactory.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk
2
3 import com.fasterxml.jackson.module.kotlin.jacksonObjectMapper
4 import okhttp3.OkHttpClient
5 import retrofit2.Retrofit
6 import retrofit2.converter.jackson.JacksonConverterFactory
7 import java.io.IOException
8
9 object YandexDiskRepositoryTestFactory {
10
11     private val jackson = jacksonObjectMapper().findAndRegisterModules()
12
13     fun createApi(
14         baseUrl: String,
15         tokenProvider: () -> String?,
16     ): YandexDiskApi {
17         val client = OkHttpClient.Builder()
18             .addInterceptor { chain ->
19                 val token = tokenProvider()
20                 ?: throw IOException("Yandex OAuth token is missing")
21                 chain.proceed(
22                     chain.request().newBuilder()
23                         .header("Authorization", "OAuth $token")
24                         .build(),
25                 )
26             }
27             .build()
28         return Retrofit.Builder()
29             .baseUrl(baseUrl)
30             .client(client)
31             .addConverterFactory(JacksonConverterFactory.create(jackson))
32             .build()
33             .create(YandexDiskApi::class.java)
34     }
35 }
36
```

Исходный файл domain-vault/src/test/java/com/github/nullptroma/wallenc/
domain/vault/network/yandexdisk/repository/YandexDiskRepositoryTest.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.repository
2
3 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskAuthException
4 import com.github.nullptroma.wallenc.domain.vault.network.yandexdisk.YandexDiskRepositoryTestFactory
5 import kotlinx.coroutines.Dispatchers
6 import kotlinx.coroutines.runBlocking
7 import okhttp3.OkHttpClient
8 import okhttp3.mockwebserver.MockResponse
9 import okhttp3.mockwebserver.MockWebServer
10 import org.junit.After
11 import org.junit.Assert.assertEquals
12 import org.junit.Assert.assertTrue
13 import org.junit.Before
14 import org.junit.Test
15
16 class YandexDiskRepositoryTest {
17
18     private lateinit var server: MockWebServer
19     private lateinit var repository: YandexDiskRepository
20
21     @Before
22     fun setUp() {
23         server = MockWebServer()
24         server.start()
25         val api = YandexDiskRepositoryTestFactory.createApi(server.url("/").toString())
26     { "test-token" }
27         repository = YandexDiskRepository(api, OkHttpClient(), Dispatchers.IO)
28     }
29
30     @After
31     fun tearDown() {
32         server.shutdown()
33     }
34
35     @Test
36     fun diskInfoParsesResponse() = runBlocking {
37         repository.resetCloudApiCallCount()
38         server.enqueue(
39             MockResponse()
40                 .setResponseCode(200)
41                 .setBody("""{"total_space":1000,"used_space":200,"trash_size":0}""")
42                 .addHeader("Content-Type", "application/json"),
43         )
44     }
```

```

42         )
43         val info = repository.diskInfo()
44         assertEquals(1000L, info.totalSpace)
45         assertEquals(200L, info.usedSpace)
46         assertEquals(1L, repository.cloudApiCallCount())
47     }
48
49     @Test
50     fun listReturnsEmptyEmbeddedOn404() = runBlocking {
51         repeat(2) {
52             server.enqueue(MockResponse().setResponseCode(404))
53         }
54         val result = repository.list("disk:/missing", limit = 10, offset = 0)
55         assertTrue(result.embedded?.items?.isEmpty() == true)
56     }
57
58     @Test
59     fun diskInfoThrowsAuthExceptionOn401() = runBlocking {
60         server.enqueue(MockResponse().setResponseCode(401))
61         try {
62             repository.diskInfo()
63             error("Expected YandexDiskAuthException")
64         } catch (_: YandexDiskAuthException) {
65             // expected
66         }
67     }
68 }
69

```

Исходный файл domain-vault/src/test/java/com/github/nullptroma/wallenc/
domain/vault/storages/common/StorageSyncJournalBufferTest.kt

```
1 package com.github.nullptroma.wallenc.domain.vault.storages.common
2
3 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncJournalEntry
4 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncOperation
5 import com.github.nullptroma.wallenc.domain.datatypes.StorageSyncRevision
6 import kotlinx.coroutines.runBlocking
7 import org.junit.Assert.assertEquals
8 import org.junit.Assert.assertTrue
9 import org.junit.Test
10 import java.time.Instant
11 import java.util.concurrent.atomic.AtomicReference
12
13 class StorageSyncJournalBufferTest {
14
15     @Test
16     fun flushRestoresPendingOnWriteFailure() = runBlocking {
17         val disk = AtomicReference(emptyMap<String, StorageSyncJournalEntry>())
18         var shouldFail = true
19         val buffer = StorageSyncJournalBuffer(
20             syncActorId = "actor",
21             originStorageUuid = null,
22             readJournal = { disk.get() },
23             writeJournal = {
24                 if (shouldFail) {
25                     error("disk unavailable")
26                 }
27                 disk.set(it)
28             },
29         )
30         val entry = buffer.buildEntry("/a.txt", StorageSyncOperation.UPSERT, 1L)
31         try {
32             buffer.appendEntry("/a.txt", entry)
33         } catch (_: IllegalStateException) {
34             // expected
35         }
36         shouldFail = false
37         buffer.flushPending()
38         assertTrue(disk.get().containsKey("/a.txt"))
39         assertEquals(1L, disk.get()["/a.txt"]?.revision?.sequence)
40     }
41 }
42
```


ПРИЛОЖЕНИЕ А.7

Модуль :infrastructure-android

Исходный файл infrastructure-android/consumer-rules.pro

```
1  # Consumer ProGuard rules for infrastructure-android (empty).  
2
```

Исходный файл infrastructure-android/
src/androidTest/java/com/github/nullptroma/wallenc/infrastructure/android/db/app/
repository/YandexAccountRepositoryTest.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.repository
2
3 import androidx.room.Room
4 import androidx.test.core.app.ApplicationProvider
5 import androidx.test.ext.junit.runners.AndroidJUnit4
6 import com.github.nullptroma.wallenc.domain.vault.model.YandexAccount
7 import com.github.nullptroma.wallenc.infrastructure.android.db.app.AppDb
8 import kotlinx.coroutines.Dispatchers
9 import kotlinx.coroutines.runBlocking
10 import org.junit.After
11 import org.junit.Assert.assertEquals
12 import org.junit.Assert.assertNull
13 import org.junit.Before
14 import org.junit.Test
15 import org.junit.runner.RunWith
16
17 @RunWith(AndroidJUnit4::class)
18 class YandexAccountRepositoryTest {
19
20     private lateinit var db: AppDb
21     private lateinit var repository: YandexAccountRepository
22
23     @Before
24     fun setUp() {
25         val context = ApplicationProvider.getApplicationContext<android.content.Context>()
26         db = Room.inMemoryDatabaseBuilder(context, AppDb::class.java).build()
27         repository = YandexAccountRepository(
28             dao = db.yandexAccountDao,
29             ioDispatcher = Dispatchers.IO,
30         )
31     }
32
33     @After
34     fun tearDown() {
35         db.close()
36     }
37
38     @Test
39     fun insertAndLoadByVaultUuid() = runBlocking {
40         val account = YandexAccount(
41             vaultUuid = "vault-1",
```

```

42         yandexUserId = "user-1",
43         email = "test@yandex.ru",
44         oauthToken = "token-abc",
45     )
46     repository.insert(account)
47     val loaded = repository.getByVaultUuid("vault-1")
48     assertEquals(account, loaded)
49 }
50
51 @Test
52 fun updateCredentialsChangesToken() = runBlocking {
53     repository.insert(
54         YandexAccount(
55             vaultUuid = "vault-2",
56             yandexUserId = "user-2",
57             email = "old@yandex.ru",
58             oauthToken = "old-token",
59         ),
60     )
61     repository.updateCredentials("vault-2", "new@yandex.ru", "new-token")
62     val loaded = repository.getByVaultUuid("vault-2")
63     assertEquals("new@yandex.ru", loaded?.email)
64     assertEquals("new-token", loaded?.oauthToken)
65 }
66
67 @Test
68 fun deleteByVaultUuidRemovesRow() = runBlocking {
69     repository.insert(
70         YandexAccount(
71             vaultUuid = "vault-3",
72             yandexUserId = "user-3",
73             email = "x@yandex.ru",
74             oauthToken = "t",
75         ),
76     )
77     repository.deleteByVaultUuid("vault-3")
78     assertNull(repository.getByVaultUuid("vault-3"))
79 }
80 }
81

```

Исходный файл infrastructure-android/src/main/AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest />
3
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/RoomFactory.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db
2
3 import android.content.Context
4 import androidx.room.Room
5 import com.github.nullptroma.wallenc.infrastructure.android.db.app.AppDb
6
7 class RoomFactory(private val context: Context) {
8     fun buildAppDb(): AppDb {
9         val room = Room.databaseBuilder(
10             context, AppDb::class.java, "app-db"
11         ).fallbackToDestructiveMigration(dropAllTables = true).build()
12         return room
13     }
14 }
15
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/AppDb.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app
2
3 import androidx.room.Database
4 import androidx.room.RoomDatabase
5 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageKeyMapDao
6 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageMetaInfoDao
7 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageSyncGroupDao
8 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.YandexAccountDao
9 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbStorageKeyMap
10 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbStorageMetaInfo
11 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbStorageSyncGroup
12 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbYandexAccount
13
14 interface IAppDb {
15     val storageKeyMapDao: StorageKeyMapDao
16     val storageMetaInfoDao: StorageMetaInfoDao
17     val storageSyncGroupDao: StorageSyncGroupDao
18     val yandexAccountDao: YandexAccountDao
19 }
20
21 @Database(
22     entities = [DbStorageKeyMap::class, DbStorageMetaInfo::class, DbYandexAccount::class,
23     DbStorageSyncGroup::class],
24     version = 5,
25     exportSchema = false
26 )
27 abstract class AppDb : IAppDb, RoomDatabase() {
28     abstract override val storageKeyMapDao: StorageKeyMapDao
29     abstract override val storageMetaInfoDao: StorageMetaInfoDao
30     abstract override val storageSyncGroupDao: StorageSyncGroupDao
31     abstract override val yandexAccountDao: YandexAccountDao
32 }
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/dao/StorageKeyMapDao.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.dao
2
3 import androidx.room.Dao
4 import androidx.room.Delete
5 import androidx.room.Insert
6 import androidx.room.OnConflictStrategy
7 import androidx.room.Query
8 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbStorageKeyMap
9
10 @Dao
11 interface StorageKeyMapDao {
12     @Insert(onConflict = OnConflictStrategy.REPLACE)
13     suspend fun add(vararg keymaps: DbStorageKeyMap)
14
15     @Query("SELECT * FROM storage_key_maps")
16     suspend fun getAll(): List<DbStorageKeyMap>
17
18     @Delete
19     suspend fun delete(vararg keymaps: DbStorageKeyMap)
20 }
21
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/dao/StorageMetaInfoDao.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.dao
2
3 import androidx.room.Dao
4 import androidx.room.Delete
5 import androidx.room.Insert
6 import androidx.room.OnConflictStrategy
7 import androidx.room.Query
8 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbStorageMetaInfo
9 import kotlinx.coroutines.flow.Flow
10 import java.util.UUID
11
12 @Dao
13 interface StorageMetaInfoDao {
14     @Insert(onConflict = OnConflictStrategy.REPLACE)
15     suspend fun add(metaInfo: DbStorageMetaInfo)
16
17     @Query("SELECT * FROM storage_meta_infos")
18     suspend fun getAll(): List<DbStorageMetaInfo>
19
20     @Query("SELECT * FROM storage_meta_infos")
21     fun getAllFlow(): Flow<List<DbStorageMetaInfo>>
22
23     @Query("SELECT * FROM storage_meta_infos WHERE uuid == :uuid")
24     fun getMetaInfoFlow(uuid: UUID): Flow<DbStorageMetaInfo>
25
26     @Query("SELECT * FROM storage_meta_infos WHERE uuid == :uuid")
27     suspend fun getMetaInfo(uuid: UUID): DbStorageMetaInfo?
28
29     @Delete
30     suspend fun delete(metaInfo: DbStorageMetaInfo)
31
32     @Query("DELETE FROM storage_meta_infos WHERE uuid == :uuid")
33     suspend fun delete(uuid: UUID)
34 }
35
```


Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/dao/StorageSyncGroupDao.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.dao
2
3 import androidx.room.Dao
4 import androidx.room.Insert
5 import androidx.room.OnConflictStrategy
6 import androidx.room.Query
7 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbStorageSyncGroup
8
9 @Dao
10 interface StorageSyncGroupDao {
11     @Query("SELECT * FROM storage_sync_groups")
12     suspend fun getAll(): List<DbStorageSyncGroup>
13
14     @Insert(onConflict = OnConflictStrategy.REPLACE)
15     suspend fun upsert(group: DbStorageSyncGroup)
16
17     @Query("DELETE FROM storage_sync_groups WHERE group_id = :groupId")
18     suspend fun deleteById(groupId: String)
19 }
20
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/dao/YandexAccountDao.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.dao
2
3 import androidx.room.Dao
4 import androidx.room.Insert
5 import androidx.room.Query
6 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbYandexAccount
7 import kotlinx.coroutines.flow.Flow
8
9 @Dao
10 interface YandexAccountDao {
11     @Query("SELECT * FROM yandex_accounts WHERE yandexUserId = :id LIMIT 1")
12     suspend fun getByYandexUserId(id: String): DbYandexAccount?
13
14     @Query("SELECT * FROM yandex_accounts WHERE vaultUuid = :vaultUuid LIMIT 1")
15     suspend fun getByVaultUuid(vaultUuid: String): DbYandexAccount?
16
17     @Insert
18     suspend fun insert(account: DbYandexAccount)
19
20     @Query(
21         "UPDATE yandex_accounts SET oauthToken = :token, email = :email WHERE vaultUuid = :vaultUuid",
22     )
23     suspend fun updateCredentials(vaultUuid: String, email: String, token: String)
24
25     @Query("SELECT * FROM yandex_accounts ORDER BY email COLLATE NOCASE ASC")
26     fun observeAll(): Flow<List<DbYandexAccount>>
27
28     @Query("DELETE FROM yandex_accounts WHERE vaultUuid = :vaultUuid")
29     suspend fun deleteByVaultUuid(vaultUuid: String)
30 }
31
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/model/DbStorageKeyMap.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.model
2
3 import androidx.room.ColumnInfo
4 import androidx.room.Entity
5 import androidx.room.PrimaryKey
6 import com.github.nullptroma.wallenc.domain.vault.model.StorageKeyMap
7 import com.github.nullptroma.wallenc.domain.datatypes.EncryptKey
8 import java.util.UUID
9
10 @Entity(tableName = "storage_key_maps")
11 data class DbStorageKeyMap(
12     @PrimaryKey
13     @ColumnInfo(name = "source_uuid") val sourceUuid: UUID,
14     @ColumnInfo(name = "key") val key: ByteArray
15 ) {
16     fun toModel(): StorageKeyMap {
17         return StorageKeyMap(
18             sourceUuid = sourceUuid,
19             key = EncryptKey(key)
20         )
21     }
22
23     override fun equals(other: Any?): Boolean {
24         if (this === other) return true
25         if (javaClass != other?.javaClass) return false
26
27         other as DbStorageKeyMap
28
29         if (sourceUuid != other.sourceUuid) return false
30         if (!key.contentEquals(other.key)) return false
31
32         return true
33     }
34
35     override fun hashCode(): Int {
36         var result = sourceUuid.hashCode()
37         result = 31 * result + key.contentHashCode()
38         return result
39     }
40
41     companion object {
42         fun fromModel(keymap: StorageKeyMap): DbStorageKeyMap {
```

```
43         return DbStorageKeyMap(  
44             sourceUuid = keymap.sourceUuid,  
45             key = keymap.key.bytes  
46         )  
47     }  
48 }  
49 }  
50
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/model/DbStorageMetaInfo.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.model
2
3 import androidx.room.ColumnInfo
4 import androidx.room.Entity
5 import androidx.room.PrimaryKey
6 import java.util.UUID
7
8 @Entity(tableName = "storage_meta_infos")
9 data class DbStorageMetaInfo(
10     @PrimaryKey @ColumnInfo(name = "uuid") val uuid: UUID,
11     @ColumnInfo(name = "meta_info") val metaInfoJson: String
12 )
13
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/model/DbStorageSyncGroup.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.model
2
3 import androidx.room.ColumnInfo
4 import androidx.room.Entity
5 import androidx.room.PrimaryKey
6
7 @Entity(tableName = "storage_sync_groups")
8 data class DbStorageSyncGroup(
9     @PrimaryKey @ColumnInfo(name = "group_id") val id: String,
10     @ColumnInfo(name = "storage_uuids_csv") val storageUuidsCsv: String,
11     @ColumnInfo(name = "encryption_kind") val encryptionKind: String,
12     @ColumnInfo(name = "encryption_secret") val encryptionSecret: String?,
13 )
14
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/
wallenc/infrastructure/android/db/app/model/DbYandexAccount.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.model
2
3 import androidx.room.Entity
4 import androidx.room.Index
5 import androidx.room.PrimaryKey
6
7 @Entity(
8     tableName = "yandex_accounts",
9     indices = [Index(value = ["yandexUserId"], unique = true)],
10 )
11 data class DbYandexAccount(
12     @PrimaryKey val vaultUuid: String,
13     val yandexUserId: String,
14     val email: String,
15     val oauthToken: String,
16 )
17
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/wallenc/infrastructure/android/db/app/repository/StorageKeyMapRepository.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.repository
2
3 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageKeyMapDao
4 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbStorageKeyMap
5 import com.github.nullptroma.wallenc.domain.vault.model.StorageKeyMap
6 import com.github.nullptroma.wallenc.domain.vault.ports.StorageKeyMapStore
7 import kotlinx.coroutines.CoroutineDispatcher
8 import kotlinx.coroutines.withContext
9
10 class StorageKeyMapRepository(
11     private val dao: StorageKeyMapDao,
12     private val ioDispatcher: CoroutineDispatcher
13 ) : StorageKeyMapStore {
14     override suspend fun getAll() = withContext(ioDispatcher) { dao.getAll().map
15     { it.toModel() } }
16     override suspend fun add(value: StorageKeyMap) = withContext(ioDispatcher) {
17         val dbModel = DbStorageKeyMap.fromModel(value)
18         dao.add(dbModel)
19     }
20     suspend fun add(vararg keymaps: StorageKeyMap) = withContext(ioDispatcher) {
21         val dbModels = keymaps.map { DbStorageKeyMap.fromModel(it) }
22         dao.add(*dbModels.toTypedArray())
23     }
24
25     override suspend fun delete(vararg values: StorageKeyMap) = withContext(ioDispatcher) {
26         val dbModels = values.map { DbStorageKeyMap.fromModel(it) }
27         dao.delete(*dbModels.toTypedArray())
28     }
29 }
30
```


Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/wallenc/infrastructure/android/db/app/repository/StorageSyncGroupRepository.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.repository
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IStorageSyncGroupStore
4 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroup
5 import com.github.nullptroma.wallenc.domain.interfaces.StorageSyncGroupEncryptionKind
6 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.StorageSyncGroupDao
7 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbStorageSyncGroup
8 import com.github.nullptroma.wallenc.domain.vault.utils.IProvider
9 import kotlinx.coroutines.CoroutineDispatcher
10 import kotlinx.coroutines.withContext
11 import java.util.UUID
12
13 class StorageSyncGroupRepository(
14     private val dao: StorageSyncGroupDao,
15     private val ioDispatcher: CoroutineDispatcher,
16 ) : IStorageSyncGroupStore {
17     private val allGroupsProvider: IProvider<List<StorageSyncGroup>> = object :
18         IProvider<List<StorageSyncGroup>> {
19         override suspend fun get(): List<StorageSyncGroup> =
20             dao.getAll().mapNotNull(::toDomain)
21     }
22
23     override suspend fun set(value: List<StorageSyncGroup>) {
24         val desiredIds = value.map { it.id }.toSet()
25         val current = dao.getAll()
26         current
27             .asSequence()
28             .filter { it.id !in desiredIds }
29             .forEach { dao.deleteById(it.id) }
30         value.forEach { group ->
31             dao.upsert(toDb(group))
32         }
33     }
34
35     override suspend fun clear() {
36         dao.getAll().forEach { dao.deleteById(it.id) }
37     }
38
39     override suspend fun getGroups(): List<StorageSyncGroup> = withContext(ioDispatcher) {
40         allGroupsProvider.get().orEmpty()
41     }
42
43     override suspend fun putGroup(group: StorageSyncGroup) = withContext(ioDispatcher) {
```

```

42         val all = allGroupsProvider.get().orEmpty().toMutableList()
43         val idx = all.indexOfFirst { it.id == group.id }
44         if (idx >= 0) {
45             all[idx] = group
46         } else {
47             all.add(group)
48         }
49         allGroupsProvider.set(all)
50     }
51
52     override suspend fun removeGroup(groupId: String) = withContext(ioDispatcher) {
53         val all = allGroupsProvider.get().orEmpty().filterNot { it.id == groupId }
54         allGroupsProvider.set(all)
55     }
56
57     private fun toDb(group: StorageSyncGroup): DbStorageSyncGroup = DbStorageSyncGroup(
58         id = group.id,
59         storageUuidsCsv = group.storageUuids.joinToString(",") { it.toString() },
60         encryptionKind = group.encryptionKind.name,
61         encryptionSecret = group.encryptionSecret,
62     )
63
64     private fun toDomain(db: DbStorageSyncGroup): StorageSyncGroup? {
65         val kind = runCatching {
66             StorageSyncGroupEncryptionKind.valueOf(db.encryptionKind)
67         }.getOrNull {
68             StorageSyncGroupEncryptionKind.UNSET
69         }
70         if (db.id.isBlank()) {
71             return null
72         }
73         val uuids = db.storageUuidsCsv
74             .split(",")
75             .mapNotNull { token ->
76                 val value = token.trim()
77                 if (value.isBlank()) {
78                     null
79                 } else {
80                     runCatching { UUID.fromString(value) }.getOrNull()
81                 }
82             }
83             .toSet()
84         return StorageSyncGroup(
85             id = db.id,
86             storageUuids = uuids,

```

```
87         encryptionKind = kind,
88         encryptionSecret = db.encryptionSecret?.takeIf { it.isNotBlank() },
89     )
90 }
91 }
92
```

Исходный файл infrastructure-android/src/main/java/com/github/nullptroma/wallenc/infrastructure/android/db/app/repository/YandexAccountRepository.kt

```
1 package com.github.nullptroma.wallenc.infrastructure.android.db.app.repository
2
3 import com.github.nullptroma.wallenc.infrastructure.android.db.app.dao.YandexAccountDao
4 import com.github.nullptroma.wallenc.infrastructure.android.db.app.model.DbYandexAccount
5 import com.github.nullptroma.wallenc.domain.vault.model.YandexAccount
6 import com.github.nullptroma.wallenc.domain.vault.ports.YandexAccountStore
7 import kotlinx.coroutines.CoroutineDispatcher
8 import kotlinx.coroutines.flow.Flow
9 import kotlinx.coroutines.flow.map
10 import kotlinx.coroutines.withContext
11
12 class YandexAccountRepository(
13     private val dao: YandexAccountDao,
14     private val ioDispatcher: CoroutineDispatcher
15 ) : YandexAccountStore {
16     override fun observeAll(): Flow<List<YandexAccount>> = dao.observeAll().map { rows ->
17         rows.map { it.toModel() }
18     }
19
20     override suspend fun getByYandexUserId(id: String): YandexAccount? =
21     withContext(ioDispatcher) {
22         dao.getByYandexUserId(id)?.toModel()
23     }
24
25     override suspend fun getByVaultUuid(vaultUuid: String): YandexAccount? =
26     withContext(ioDispatcher) {
27         dao.getByVaultUuid(vaultUuid)?.toModel()
28     }
29
30     override suspend fun insert(account: YandexAccount) = withContext(ioDispatcher) {
31         dao.insert(fromModel(account))
32     }
33
34     override suspend fun updateCredentials(vaultUuid: String, email: String, token: String)
35     =
36     withContext(ioDispatcher) {
37         dao.updateCredentials(vaultUuid, email, token)
38     }
39
40     override suspend fun deleteByVaultUuid(vaultUuid: String) = withContext(ioDispatcher) {
41         dao.deleteByVaultUuid(vaultUuid)
42     }
43 }
```

```

41     private fun DbYandexAccount.toModel(): YandexAccount =
42         YandexAccount(
43             vaultUuid = vaultUuid,
44             yandexUserId = yandexUserId,
45             email = email,
46             oauthToken = oauthToken,
47         )
48
49     private fun fromModel(model: YandexAccount): DbYandexAccount =
50         DbYandexAccount(
51             vaultUuid = model.vaultUuid,
52             yandexUserId = model.yandexUserId,
53             email = model.email,
54             oauthToken = model.oauthToken,
55         )
56 }
57

```

ПРИЛОЖЕНИЕ А.8

Модуль :vault-contracts

Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/CloudBrand.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 /**
4  * Поддерживаемые облачные провайдеры удалённых vault'ов.
5  *
6  * Бренд «локального» устройства тут не указывается — для дискриминации UI
7  * используется [VaultDescriptor.LocalDevice] vs [VaultDescriptor.LinkedRemote].
8  */
9 enum class CloudBrand {
10     YANDEX,
11 }
12
```

Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/DescribedVault.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IVault
4
5 /**
6  * Vault, дополнительно знающий свою категорию для UI/маршрутизации.
7  *
8  * Все реализации vault'ов в `:data` обязаны быть [DescribedVault] –
9  * это единственный способ для presentation/app узнать, что это за vault,
10  * не имея зависимости от `:data`.
11  */
12 interface DescribedVault : IVault {
13     val descriptor: VaultDescriptor
14 }
15
```

Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/RemoteVaultAuthenticator.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 /**
4  * Запуск OAuth-сценария привязки удалённого хранилища для конкретного [CloudBrand].
5  *
6  * Реализация в `:app` (привязана к Activity). Слой presentation вызывает [beginLink]
7  * из контекста Compose-экрана и получает [VaultLinkOutcome] асинхронно.
8  *
9  * Намеренно императивный API через колбэк, чтобы корректно встроиться
10 * в `ActivityResultLauncher` Yandex Auth SDK без удержания Activity синглтоном.
11 */
12 fun interface RemoteVaultAuthenticator {
13     /**
14      * Начать сценарий линка для бренда [brand].
15      *
16      * Если бренд не поддерживается — реализация должна синхронно вызвать
17      * `onResult(VaultLinkOutcome.Failed(...))`, не бросая исключение.
18      */
19     fun beginLink(brand: CloudBrand, onResult: (VaultLinkOutcome) -> Unit)
20 }
21
```


Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/VaultDescriptor.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IVaultInfo
4 import java.util.UUID
5
6 /**
7  * Категория vault'a для UI и ветвления – sealed-иерархия в одном модуле,
8  * чтобы `when` был exhaustive.
9  *
10 * `domain` про эти подтипы ничего не знает; внешнее кольцо (`:vault-api`)
11 * расширяет [IVaultInfo] (только `uuid`) до структуры с дополнительной
12 * категоризацией.
13 */
14 sealed interface VaultDescriptor : IVaultInfo {
15     /** Vault, физически живущий на устройстве; без привязки к облачному аккаунту. */
16     data class LocalDevice(
17         override val uuid: UUID,
18     ) : VaultDescriptor
19
20     /** Vault, привязанный к облачному аккаунту через OAuth. */
21     data class LinkedRemote(
22         override val uuid: UUID,
23         val brand: CloudBrand,
24         val accountDisplayName: String,
25     ) : VaultDescriptor
26 }
27
```

Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/VaultLinkFailure.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 enum class VaultLinkFailure {
4     UnsupportedBrand,
5     NotRegistered,
6     AuthError,
7     Unknown,
8 }
9
```

Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/VaultLinkOutcome.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 /**
4  * Результат сценария OAuth-линка нового удалённого хранилища.
5  *
6  * Слой presentation сводит это к: успех → отдать `registration` в
7  * [VaultRegistrar.register];
8  * cancel → ничего не делать; failure → показать сообщение.
9  */
10 sealed interface VaultLinkOutcome {
11     /** Пользователь успешно вошёл; в [registration] лежат данные для регистрации. */
12     data class Success(val registration: VaultRegistration) : VaultLinkOutcome
13
14     /** Пользователь отменил вход. */
15     data object Cancelled : VaultLinkOutcome
16
17     /** Ошибка SDK/сети/сервера; [reason] маппится в UI через [VaultLinkFailure]. */
18     data class Failed(val reason: VaultLinkFailure) : VaultLinkOutcome
19 }
```

Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/VaultRegistrar.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 import java.util.UUID
4
5 /**
6  * Контракт регистрации/удаления удалённого vault'a.
7  * Реализуется тем же объектом, что и `domain.IVaultsManager`.
8  */
9 interface VaultRegistrar {
10     /**
11      * Зарегистрировать новый удалённый vault по «полезной нагрузке» из
12      * [VaultLinkOutcome.Success]. Если тип [registration] неизвестен реализации,
13      * бросает [IllegalArgumentException].
14      */
15     suspend fun register(registration: VaultRegistration)
16
17     /** Удалить ранее зарегистрированный vault по идентификатору. */
18     suspend fun unregister(vaultUuid: UUID)
19
20     /**
21      * Повторная попытка инициализации/подключения удалённого vault.
22      * Возвращает false, если vault не найден или не поддерживает retry.
23      */
24     suspend fun retry(vaultUuid: UUID): Boolean
25 }
26
```

Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/VaultRegistration.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 /**
4  * Маркер «полезной нагрузки» для регистрации удалённого хранилища через [VaultRegistrar].
5  *
6  * Намеренно НЕ sealed: конкретные реализации (`YandexRegistration`, ...) живут в `:data`
7  * рядом с соответствующими реализациями vault, чтобы не дробить модуль без нужды.
8  * Цена — отсутствие exhaustive-when через границу модуля; лечится fail-fast веткой
9  * `else` в `VaultsManager.register(...)` .
10 *
11 * Код приложения (presentation) не раскрывает этот тип — только передаёт экземпляр
12 * из [VaultLinkOutcome.Success] в [VaultRegistrar.register].
13 */
14 interface VaultRegistration
15
```

Исходный файл vault-contracts/src/main/java/com/github/nullptroma/wallenc/
vault/contract/Vaults.kt

```
1 package com.github.nullptroma.wallenc.vault.contract
2
3 import com.github.nullptroma.wallenc.domain.interfaces.IVault
4
5 /**
6  * Хелперы фильтрации списка vault'ов по [VaultDescriptor].
7  *
8  * Все реализации vault'а обязаны быть [DescribedVault];
9  * [described] отбрасывает те, что не соответствуют контракту (на случай
10 * вспомогательных тест-двойников и т.п.).
11 */
12 fun List<IVault>.described(): List<DescribedVault> = filterIsInstance<DescribedVault>()
13
14 val List<DescribedVault>.locals: List<DescribedVault>
15     get() = filter { it.descriptor is VaultDescriptor.LocalDevice }
16
17 val List<DescribedVault>.remotes: List<DescribedVault>
18     get() = filter { it.descriptor is VaultDescriptor.LinkedRemote }
19
```

ПРИЛОЖЕНИЕ А.9

Модуль :task-runtime

Исходный файл task-runtime/src/main/java/com/github/nullptroma/wallenc/task/runtime/TaskOrchestrator.kt

```
1 package com.github.nullptroma.wallenc.task.runtime
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.domain.errors.toWallencException
5 import com.github.nullptroma.wallenc.domain.tasks.ITaskOrchestrator
6 import com.github.nullptroma.wallenc.domain.tasks.PipelineState
7 import com.github.nullptroma.wallenc.domain.tasks.PipelineTask
8 import com.github.nullptroma.wallenc.domain.tasks.PipelineWork
9 import com.github.nullptroma.wallenc.domain.tasks.TaskContext
10 import com.github.nullptroma.wallenc.domain.tasks.TaskForegroundItem
11 import com.github.nullptroma.wallenc.domain.tasks.TaskForegroundUiState
12 import com.github.nullptroma.wallenc.domain.tasks.TaskId
13 import com.github.nullptroma.wallenc.domain.tasks.TaskLogKey
14 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLevel
15 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLine
16 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
17 import com.github.nullptroma.wallenc.domain.tasks.TaskProgress
18 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
19 import kotlinx.coroutines.CancellationException
20 import kotlinx.coroutines.CoroutineDispatcher
21 import kotlinx.coroutines.CoroutineScope
22 import kotlinx.coroutines.Job
23 import kotlinx.coroutines.SupervisorJob
24 import kotlinx.coroutines.delay
25 import kotlinx.coroutines.coroutineContext
26 import kotlinx.coroutines.ensureActive
27 import kotlinx.coroutines.flow.MutableStateFlow
28 import kotlinx.coroutines.flow.StateFlow
29 import kotlinx.coroutines.flow.asStateFlow
30 import kotlinx.coroutines.launch
31 import java.util.Collections
32 import java.util.UUID
33 import java.util.concurrent.ConcurrentHashMap
34
35 class TaskOrchestrator(
36     private val ioDispatcher: CoroutineDispatcher,
37 ) : ITaskOrchestrator {
38
```

```

39     private val pipelineSupervisor = SupervisorJob()
40     private val scope = CoroutineScope(pipelineSupervisor + ioDispatcher)
41
42     private val tasksById =
43         Collections.synchronizedMap(linkedMapOf<TaskId, PipelineTask>())
44
45     private val cancelRequested = ConcurrentHashMap<TaskId, Boolean>()
46     private val runningJobs = ConcurrentHashMap<TaskId, Job>()
47     private val delayedForegroundJobs = ConcurrentHashMap<TaskId, Job>()
48     private val visibleForegroundTaskIds =
49         Collections.synchronizedSet(linkedSetOf<TaskId>())
50
51     private val _pipelineState = MutableStateFlow(PipelineState(emptyList(), emptySet()))
52     override val pipelineState: StateFlow<PipelineState> = _pipelineState.asStateFlow()
53
54     private val logLock = Any()
55     private val logBuffer = ArrayDeque<TaskLogLine>(MAX_LOG_LINES + 1)
56     private val _logLines = MutableStateFlow<List<TaskLogLine>>(emptyList())
57     override val logLines: StateFlow<List<TaskLogLine>> = _logLines.asStateFlow()
58
59     private val _foregroundUi =
60         MutableStateFlow<TaskForegroundUiState>(TaskForegroundUiState.Hidden)
61     override val foregroundUi: StateFlow<TaskForegroundUiState> =
62         _foregroundUi.asStateFlow()
63
64     private fun onRunningProgress(taskId: TaskId, progress: TaskProgress) {
65         replaceTask(taskId) { it.copy(state = TaskRunState.Running(progress)) }
66         emitState()
67         emitForegroundUiState()
68     }
69
70     override fun enqueue(
71         title: String,
72         dispatcher: CoroutineDispatcher,
73         work: PipelineWork,
74         busyStorageUuid: UUID?,
75         locksVaultStorageList: Boolean,
76     ): TaskId {
77         val id = TaskId()
78         val task = PipelineTask(
79             id = id,
80             title = title,
81             enqueuedAtMs = System.currentTimeMillis(),
82             dispatcher = dispatcher,
83             state = TaskRunState.Queued,
84             busyStorageUuid = busyStorageUuid,

```



```

82         locksVaultStorageList = locksVaultStorageList,
83     )
84     synchronized(tasksById) {
85         tasksById[id] = task
86     }
87     emitState()
88     launchTask(id, work)
89     return id
90 }
91
92 override fun cancel(taskId: TaskId): Boolean {
93     val exists = synchronized(tasksById) { tasksById.containsKey(taskId) }
94     if (!exists) return false
95     cancelRequested[taskId] = true
96     runningJobs[taskId]?.cancel()
97     delayedForegroundJobs.remove(taskId)?.cancel()
98     return true
99 }
100
101 override fun cancelAll() {
102     val ids = synchronized(tasksById) { tasksById.keys.toList() }
103     for (id in ids) {
104         cancelRequested[id] = true
105         runningJobs[id]?.cancel()
106         delayedForegroundJobs.remove(id)?.cancel()
107     }
108 }
109
110 override fun appendPipelineLog(level: TaskLogLevel, key: TaskLogKey) {
111     appendLogLine(level, message = "", logKey = key)
112 }
113
114 private fun replaceTask(id: TaskId, fn: (PipelineTask) -> PipelineTask) {
115     synchronized(tasksById) {
116         val cur = tasksById[id] ?: return
117         tasksById[id] = fn(cur)
118     }
119 }
120
121 private fun emitState() {
122     val snapshot = synchronized(tasksById) {
123         tasksById.values.toList()
124     }
125     val running = snapshot
126     .asSequence()

```

```

127         .filter { it.state is TaskRunState.Running }
128         .map { it.id }
129         .toSet()
130     _pipelineState.value = PipelineState(
131         tasks = snapshot,
132         runningTaskIds = running,
133     )
134 }
135
136 private fun emitForegroundUiState() {
137     val visibleItems = synchronized(tasksById) {
138         tasksById.values
139             .filter { visibleForegroundTaskIds.contains(it.id) && it.state is
TaskRunState.Running }
140             .map {
141                 TaskForegroundItem(
142                     taskId = it.id,
143                     title = it.title,
144                     progress = (it.state as TaskRunState.Running).progress,
145                 )
146             }
147     }
148     _foregroundUi.value = if (visibleItems.isEmpty()) {
149         TaskForegroundUiState.Hidden
150     } else {
151         TaskForegroundUiState.Visible(visibleItems)
152     }
153 }
154
155 private fun appendLogLine(level: TaskLogLevel, message: String, logKey: TaskLogKey? =
null) {
156     val line = TaskLogLine(
157         timestampMs = System.currentTimeMillis(),
158         level = level,
159         message = message,
160         logKey = logKey,
161     )
162     synchronized(logLock) {
163         if (logBuffer.size >= MAX_LOG_LINES) {
164             logBuffer.removeFirst()
165         }
166         logBuffer.addLast(line)
167         _logLines.value = logBuffer.toList()
168     }
169 }
170

```

```

171     private fun launchTask(taskId: TaskId, work: PipelineWork) {
172         replaceTask(taskId) { it.copy(state = TaskRunState.Running(null)) }
173         emitState()
174
175         val showForegroundJob = scope.launch {
176             delay(FOREGROUND_DELAY_MS)
177             val shouldShow = synchronized(tasksById) {
178                 val task = tasksById[taskId] ?: return@synchronized false
179                 task.state is TaskRunState.Running
180             }
181             if (shouldShow) {
182                 visibleForegroundTaskIds.add(taskId)
183                 emitForegroundUiState()
184             }
185         }
186         delayedForegroundJobs[taskId] = showForegroundJob
187
188         val dispatcher = synchronized(tasksById) { tasksById[taskId]?.dispatcher } ?:
ioDispatcher
189         val runJob = scope.launch(dispatcher) {
190             val ctx = TaskContextImpl(
191                 taskId = taskId,
192                 onRunningProgress = { p -> onRunningProgress(taskId, p) },
193                 appendLog = { level, msg, key -> appendLogLine(level, msg, key) },
194             )
195             try {
196                 if (cancelRequested[taskId] == true) {
197                     throw CancellationException("Task cancelled before start")
198                 }
199                 work.run(ctx)
200                 replaceTask(taskId) { it.copy(state = TaskRunState.Completed) }
201             } catch (_: CancellationException) {
202                 replaceTask(taskId) { it.copy(state = TaskRunState.Cancelled) }
203             } catch (e: TaskFailedException) {
204                 replaceTask(taskId) {
205                     it.copy(state = TaskRunState.Failed(e.error))
206                 }
207             } catch (e: Exception) {
208                 replaceTask(taskId) {
209                     it.copy(state = TaskRunState.Failed(e.toWallenceException()))
210                 }
211             } finally {
212                 cancelRequested.remove(taskId)
213                 runningJobs.remove(taskId)
214                 delayedForegroundJobs.remove(taskId)?.cancel()

```

```

215         visibleForegroundTaskIds.remove(taskId)
216         emitState()
217         emitForegroundUiState()
218     }
219 }
220 runningJobs[taskId] = runJob
221 }
222
223 private class TaskContextImpl(
224     override val taskId: TaskId,
225     private val onRunningProgress: (TaskProgress) -> Unit,
226     private val appendLog: (TaskLogLevel, String, TaskLogKey?) -> Unit,
227 ) : TaskContext {
228     override suspend fun reportProgress(fraction: Float?, label: TaskProgressLabel?) {
229         onRunningProgress(TaskProgress(fraction, label))
230     }
231
232     override fun log(level: TaskLogLevel, message: String) {
233         appendLog(level, message, null)
234     }
235
236     override fun log(level: TaskLogLevel, key: TaskLogKey) {
237         appendLog(level, "", key)
238     }
239
240     override fun fail(error: WallencException): Nothing {
241         throw TaskFailedException(error)
242     }
243
244     override suspend fun ensureNotCancelled() {
245         coroutineContext.ensureActive()
246     }
247 }
248
249 private class TaskFailedException(val error: WallencException) : RuntimeException()
250
251 companion object {
252     private const val MAX_LOG_LINES = 500
253     private const val FOREGROUND_DELAY_MS = 1_000L
254 }
255 }
256

```

Исходный файл task-runtime/src/test/java/com/github/nullptroma/wallenc/task/
runtime/TaskOrchestratorTest.kt

```
1 package com.github.nullptroma.wallenc.task.runtime
2
3 import com.github.nullptroma.wallenc.domain.errors.WallencException
4 import com.github.nullptroma.wallenc.domain.tasks.TaskLogLevel
5 import com.github.nullptroma.wallenc.domain.tasks.TaskProgressLabel
6 import com.github.nullptroma.wallenc.domain.tasks.TaskRunState
7 import kotlinx.coroutines.ExperimentalCoroutinesApi
8 import kotlinx.coroutines.test.StandardTestDispatcher
9 import kotlinx.coroutines.test.advanceTimeBy
10 import kotlinx.coroutines.test.advanceUntilIdle
11 import kotlinx.coroutines.test.runTest
12 import org.junit.Assert.assertTrue
13 import org.junit.Test
14
15 @OptIn(ExperimentalCoroutinesApi::class)
16 class TaskOrchestratorTest {
17
18     private val dispatcher = StandardTestDispatcher()
19
20     @Test
21     fun enqueueCompletesTask() = runTest(dispatcher) {
22         val orchestrator = TaskOrchestrator(dispatcher)
23         val id = orchestrator.enqueue(
24             title = "Test",
25             dispatcher = dispatcher,
26             work = { ctx ->
27                 ctx.reportProgress(0.5f, TaskProgressLabel.SyncPreparing(1))
28             },
29         )
30         advanceUntilIdle()
31         val task = orchestrator.pipelineState.value.tasks.first { it.id == id }
32         assertTrue(task.state is TaskRunState.Completed)
33     }
34
35     @Test
36     fun cancelAllMarksRunningTaskCancelled() = runTest(dispatcher) {
37         val orchestrator = TaskOrchestrator(dispatcher)
38         val id = orchestrator.enqueue(
39             title = "Long",
40             dispatcher = dispatcher,
41             work = { ctx ->
42                 ctx.reportProgress(null, null)
43             }
44         )
45     }
```

```

43         kotlinx.coroutines.delay(60_000)
44     },
45 )
46 advanceTimeBy(1)
47 orchestrator.cancelAll()
48 advanceUntilIdle()
49 val task = orchestrator.pipelineState.value.tasks.first { it.id == id }
50 assertTrue(task.state is TaskRunState.Cancelled)
51 }
52
53 @Test
54 fun cancelMarksTaskCancelled() = runTest(dispatcher) {
55     val orchestrator = TaskOrchestrator(dispatcher)
56     val id = orchestrator.enqueue(
57         title = "Long",
58         dispatcher = dispatcher,
59         work = { ctx ->
60             ctx.reportProgress(null, null)
61             kotlinx.coroutines.delay(60_000)
62         },
63     )
64     advanceTimeBy(1)
65     orchestrator.cancel(id)
66     advanceUntilIdle()
67     val task = orchestrator.pipelineState.value.tasks.first { it.id == id }
68     assertTrue(task.state is TaskRunState.Cancelled)
69 }
70
71 @Test
72 fun failRecordsFailedState() = runTest(dispatcher) {
73     val orchestrator = TaskOrchestrator(dispatcher)
74     val id = orchestrator.enqueue(
75         title = "Fail",
76         dispatcher = dispatcher,
77         work = { ctx ->
78             ctx.fail(WallencException.Storage.FileNotFound())
79         },
80     )
81     advanceUntilIdle()
82     val task = orchestrator.pipelineState.value.tasks.first { it.id == id }
83     val failed = task.state as TaskRunState.Failed
84     assertTrue(failed.error is WallencException.Storage.FileNotFound)
85 }
86
87 @Test

```

```

88     fun progressUpdatesRunningState() = runTest(dispatcher) {
89         val orchestrator = TaskOrchestrator(dispatcher)
90         val id = orchestrator.enqueue(
91             title = "Progress",
92             dispatcher = dispatcher,
93             work = { ctx ->
94                 ctx.reportProgress(0.25f, TaskProgressLabel.SyncCompleted)
95             },
96         )
97         advanceUntilIdle()
98         val task = orchestrator.pipelineState.value.tasks.first { it.id == id }
99         assertTrue(task.state is TaskRunState.Completed)
100     }
101
102     @Test
103     fun logAppendsLine() = runTest(dispatcher) {
104         val orchestrator = TaskOrchestrator(dispatcher)
105         orchestrator.enqueue(
106             title = "Log",
107             dispatcher = dispatcher,
108             work = { ctx ->
109                 ctx.log(TaskLogLevel.Info, "hello")
110             },
111         )
112         advanceUntilIdle()
113         assertTrue(orchestrator.logLines.value.any { it.message == "hello" })
114     }
115 }
116

```

ПРИЛОЖЕНИЕ А.10

Модуль :other

Исходный файл gradle/gradle-daemon-jvm.properties

```
1  #This file is generated by updateDaemonJvm
2  toolchainVersion=21
3
```


ПРИЛОЖЕНИЕ Б

Программная документация

ПРИЛОЖЕНИЕ Б.1

Обзор программного продукта Wallenc

Wallenc — мобильное приложение для Android: VaultsManager объединяет vault (один локальный и удалённые по OAuth), внутри каждого vault пользователь управляет storage с файлами и клиентским шифрованием. Продукт не использует собственный сервер; взаимодействие с облаком выполняется через API внешних провайдеров после OAuth.

ПРИЛОЖЕНИЕ Б.2

Техническое задание (выдержка)

Наименование: мобильное приложение для защищённого хранения пользовательских данных (Wallenc).

Основание для разработки: производственная практика, задание ООО НМФ «Нейротех», направление 09.03.04.

Назначение: обеспечение конфиденциального хранения пользовательских данных на недоверенных хранилищах.

Требования к функциям: см. табл. 2 (глава 1).

Требования к надёжности: устойчивость к прерыванию операций шифрования; восстановление метаданных из Room.

Стадии разработки: аналитический этап; реализация ядра; тестирование; оформление документации.

Порядок контроля: модульные и ручные испытания по программе (ниже).

ПРИЛОЖЕНИЕ Б.3

Программа и методика испытаний

Испытания проводятся на устройстве или эмуляторе Android. Матрица сценариев — табл. 9. Критерий приёмки: отсутствие блокирующих дефектов по сценариям Т-1–Т-6.

ПРИЛОЖЕНИЕ Б.4

Отчёт о результатах испытаний

Результаты приведены в табл. 13 (глава 5). Модульные тесты криптографии — пройдены.

ПРИЛОЖЕНИЕ Б.5

Руководство пользователя

ПРИЛОЖЕНИЕ Б.5.1

Установка

Установите APK сборки debug/release, полученной от разработчика, или соберите проект из репозитория GitLab ЮФУ. Разрешите доступ к файловой системе при запросе системы.

ПРИЛОЖЕНИЕ Б.5.2

Первый запуск и storage в локальном vault

- а) Откройте приложение Wallenc.
- б) На экране «локальные vault» (список storage в единственном LocalVault) нажмите «+» для создания storage.
- в) Укажите имя storage и подтвердите создание.

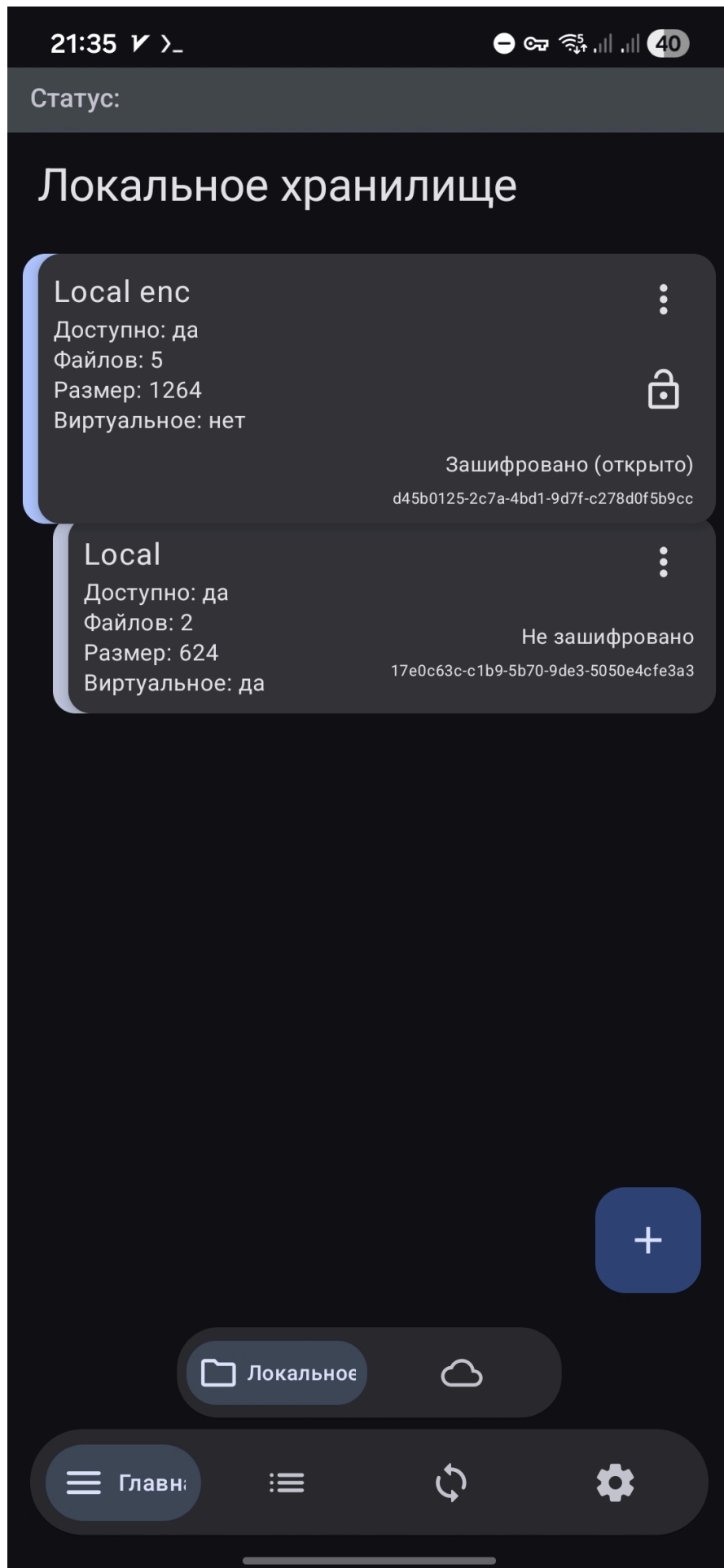


Рисунок Б.1 — Экран списка storage (локальный vault)

ПРИЛОЖЕНИЕ Б.5.3

Шифрование storage

- а) Выберите storage в списке.
- б) Выберите действие «Включить шифрование».
- в) Введите пароль (мастер-ключ) и подтвердите. **Важно:** без пароля восстановление невозможно.

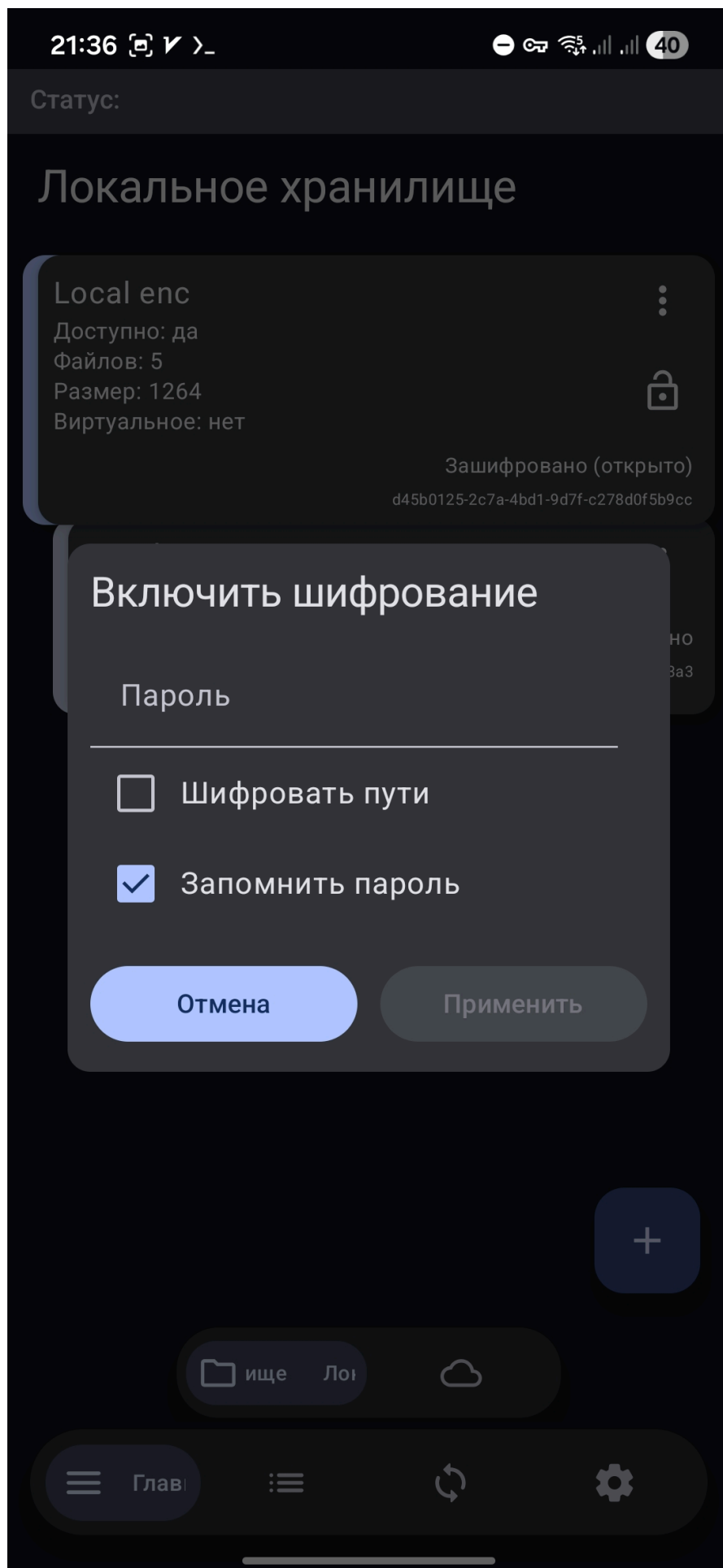


Рисунок Б.2 — Диалог включения шифрования
745

ПРИЛОЖЕНИЕ Б.5.4

Открытие и закрытие зашифрованного vault

- а) Для зашифрованного vault выберите «Открыть».
- б) Введите пароль. При успехе содержимое доступно для просмотра.
- в) Используйте «Закрыть» для блокировки.

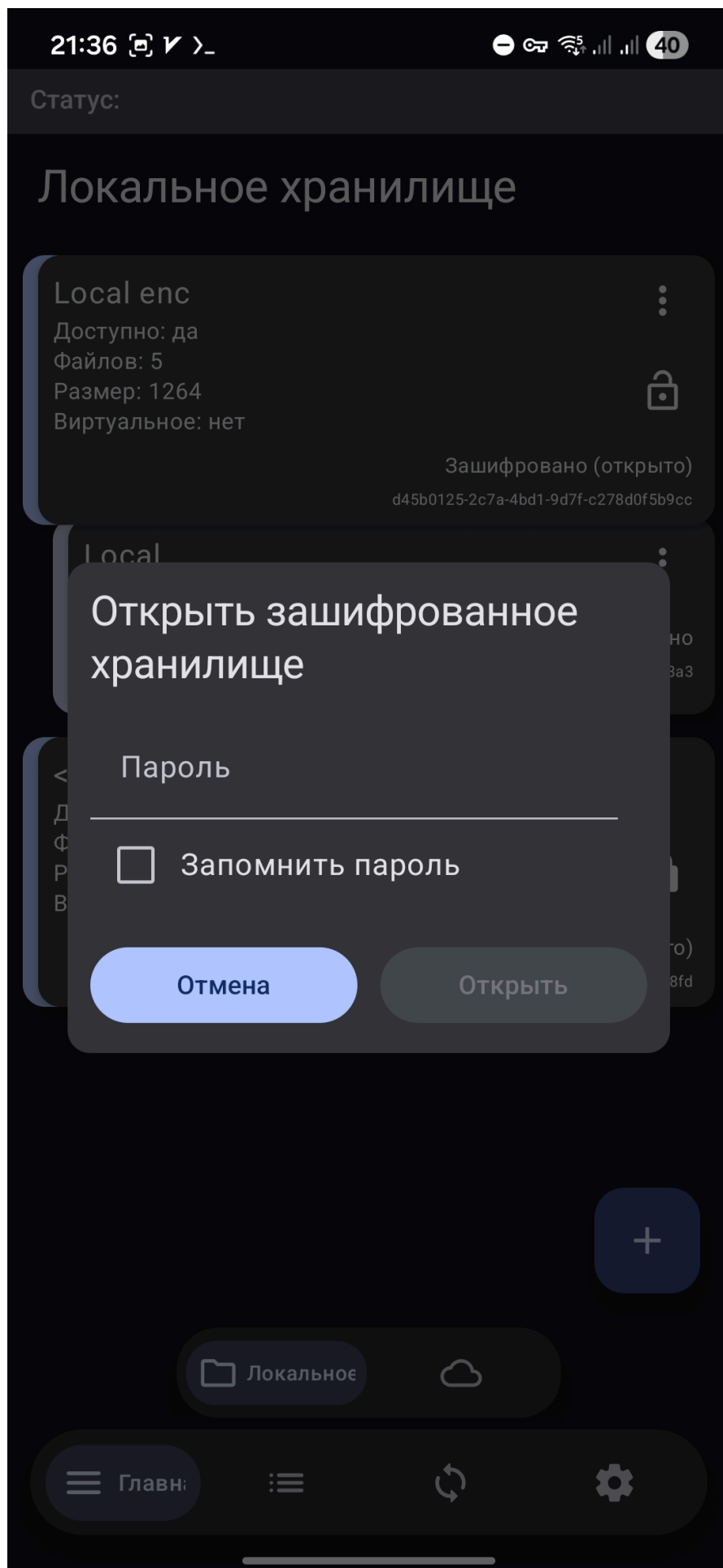


Рисунок Б.3 — Диалог открытия и закрытия
747

ПРИЛОЖЕНИЕ Б.5.5

Переименование и удаление

Долгое нажатие или меню vault → «Переименовать» / «Удалить».
Подтвердите действие в диалоге.

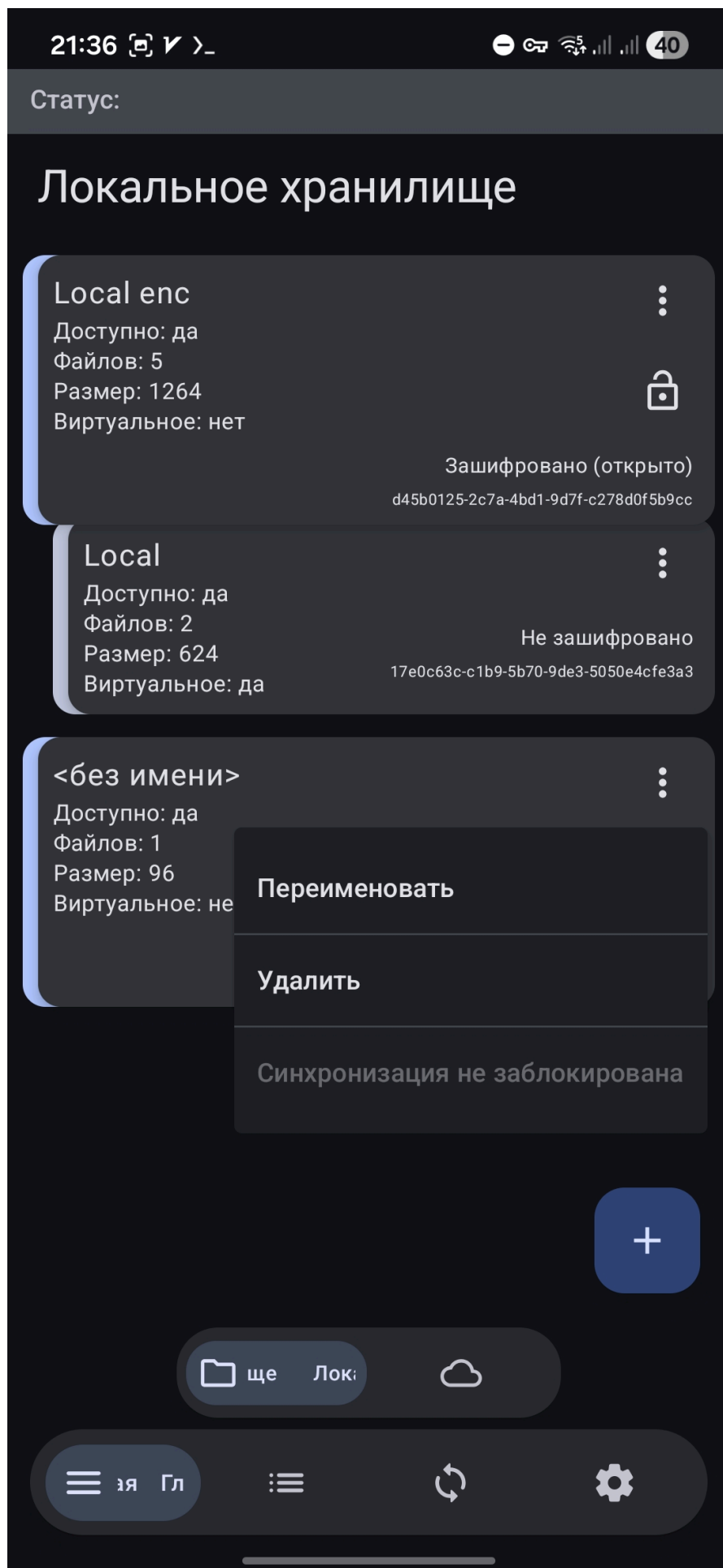


Рисунок Б.4 — Диалог переименования и удаления
749

ПРИЛОЖЕНИЕ Б.5.6

Удалённые vault и Яндекс

- а) Перейдите на экран удалённых vault.
- б) Нажмите «+» → выберите авторизацию Яндекс.
- в) Пройдите OAuth в браузере/встроенном окне.
- г) После успеха добавьте удалённое хранилище.

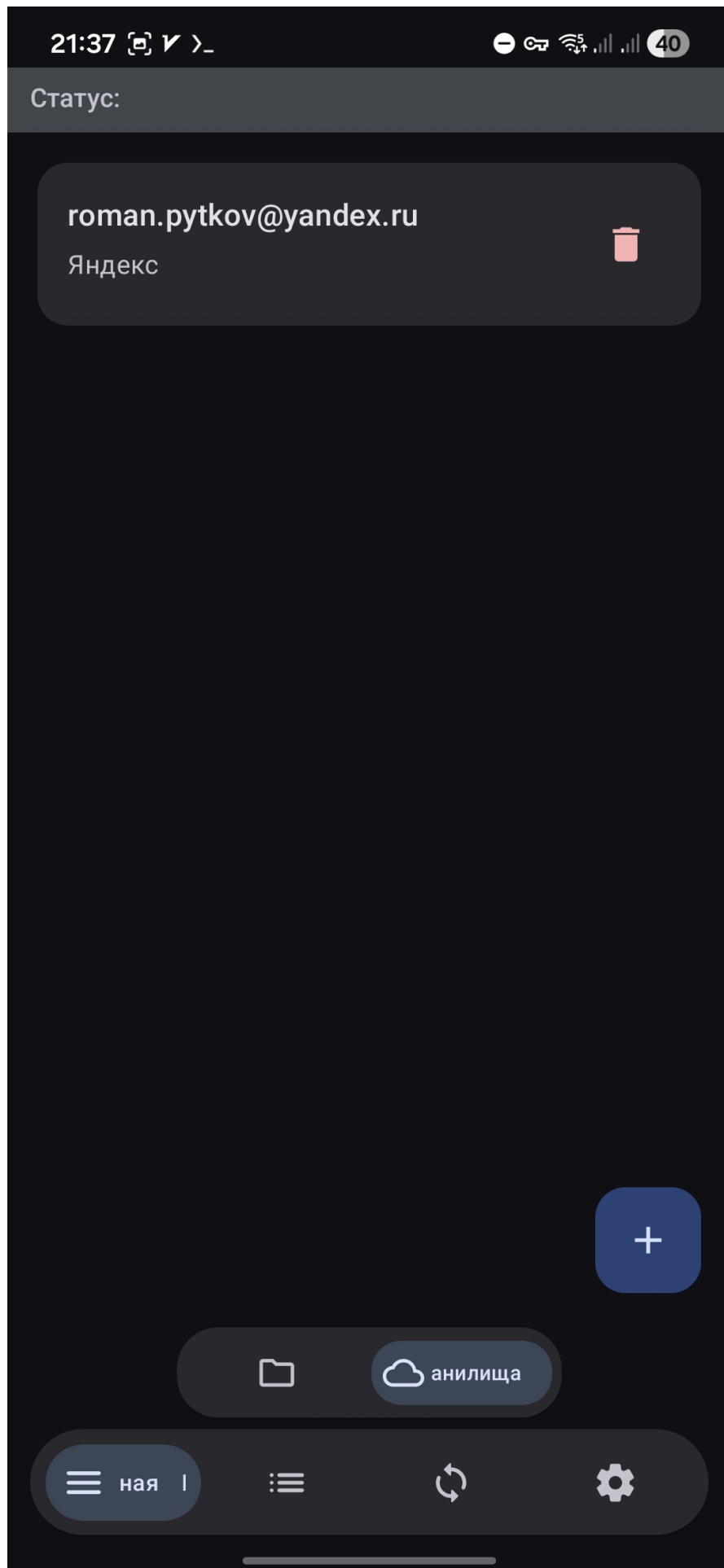


Рисунок Б.5 — Экран удалённых vault
751

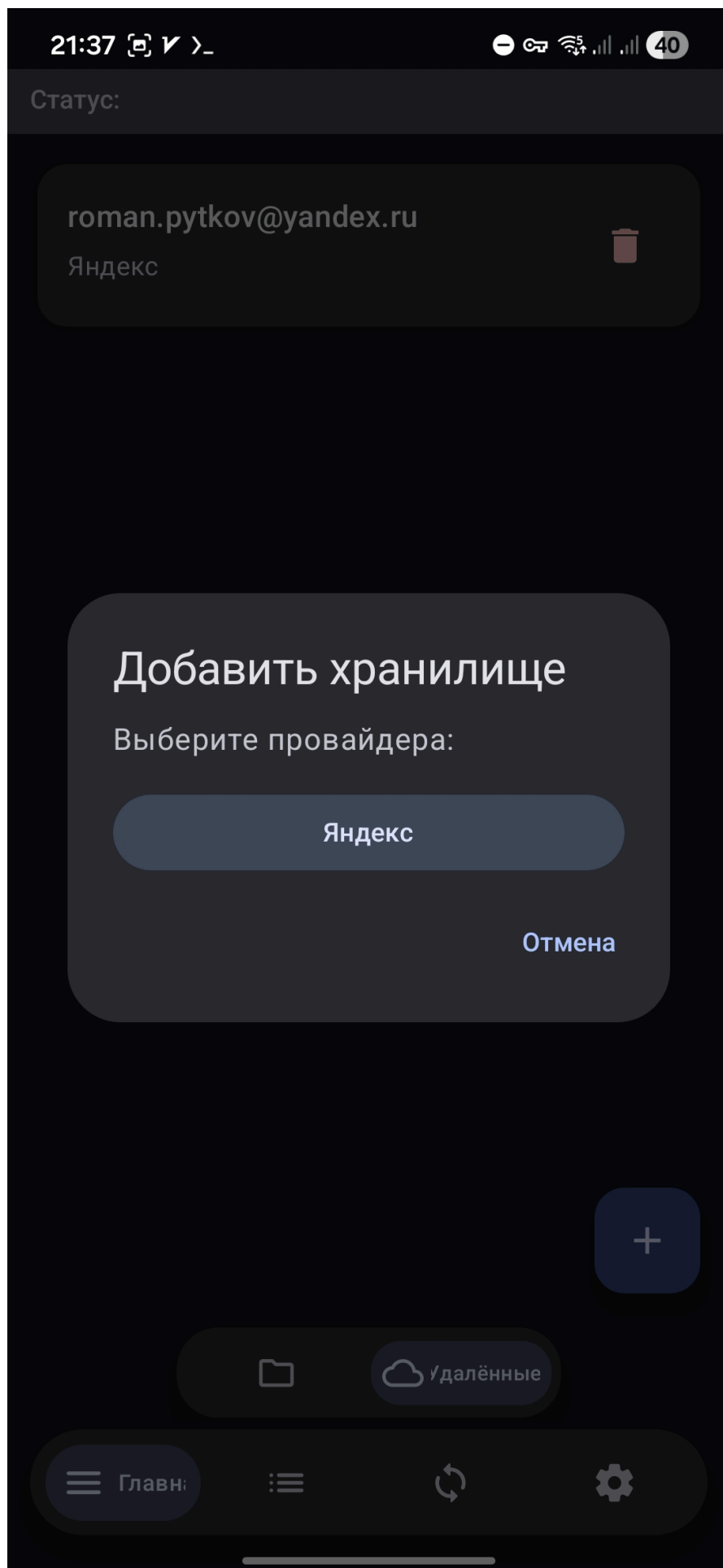


Рисунок Б.6 — OAuth Яндекс
752

ПРИЛОЖЕНИЕ Б.5.7

Секреты и 2FA внутри storage

- а) Откройте storage (после создания или из списка).
- б) На экране storage перейдите в разделы «Секреты» и «2FA».

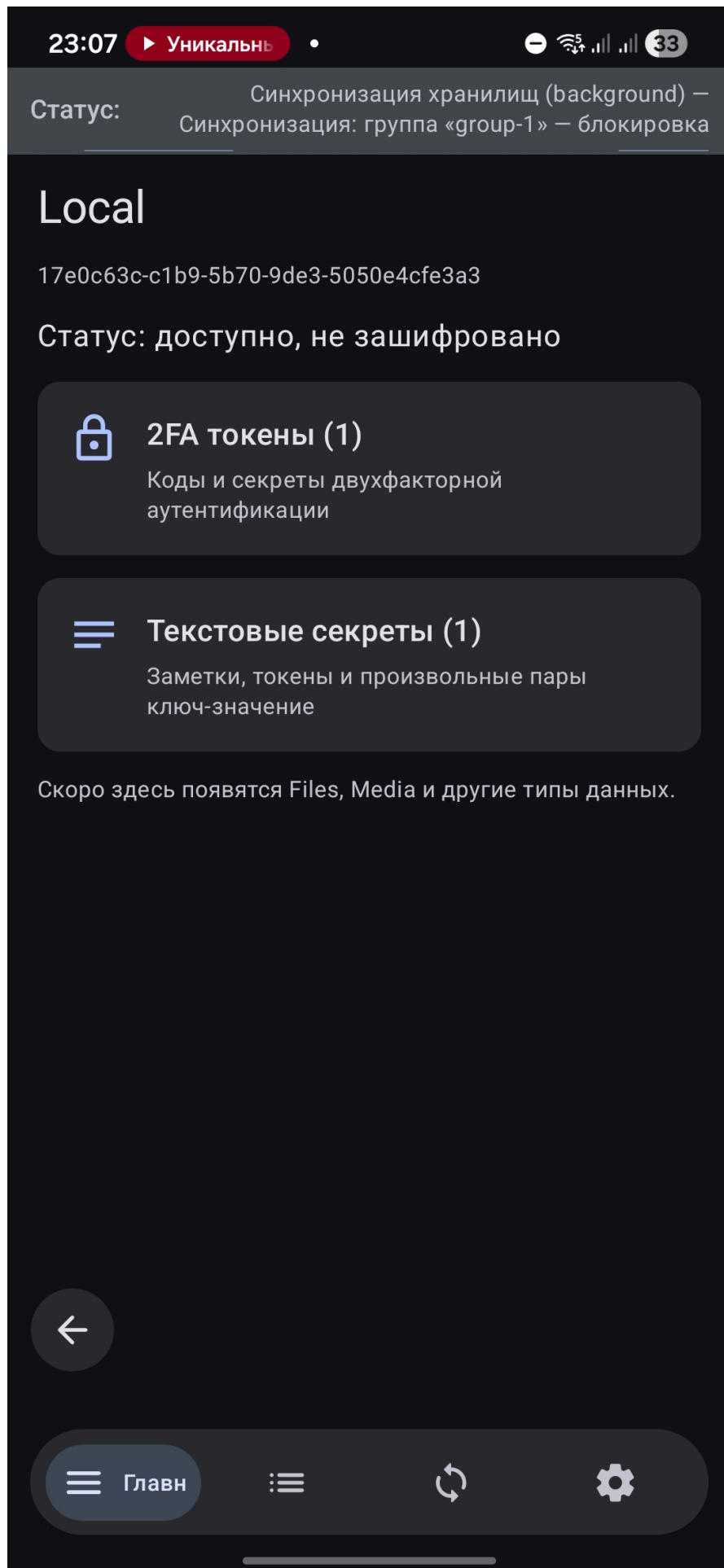


Рисунок Б.7 — Экран storage: секреты и 2FA
754

- в) В разделе 2FA добавьте TOTP-токен; на экране отображается сгенерированный код.

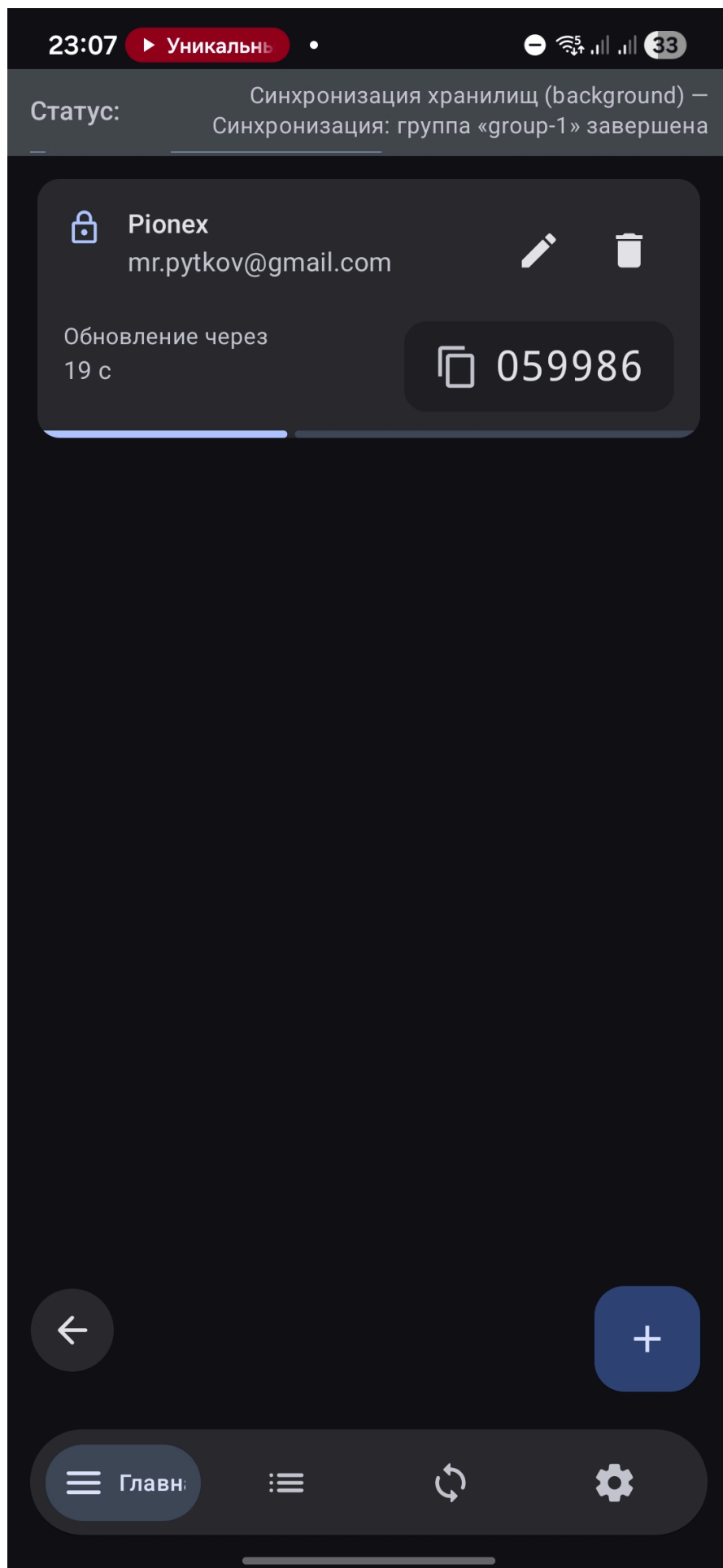


Рисунок Б.8 — Экран 2ФА с одним токеном
756

ПРИЛОЖЕНИЕ Б.5.8

Фоновые задачи

На экране задач отображаются операции шифрования и синхронизации. Уведомления информируют о завершении (см. рис. 12 и 13 в гл. 5).

ПРИЛОЖЕНИЕ В

Скриншоты пользовательского интерфейса

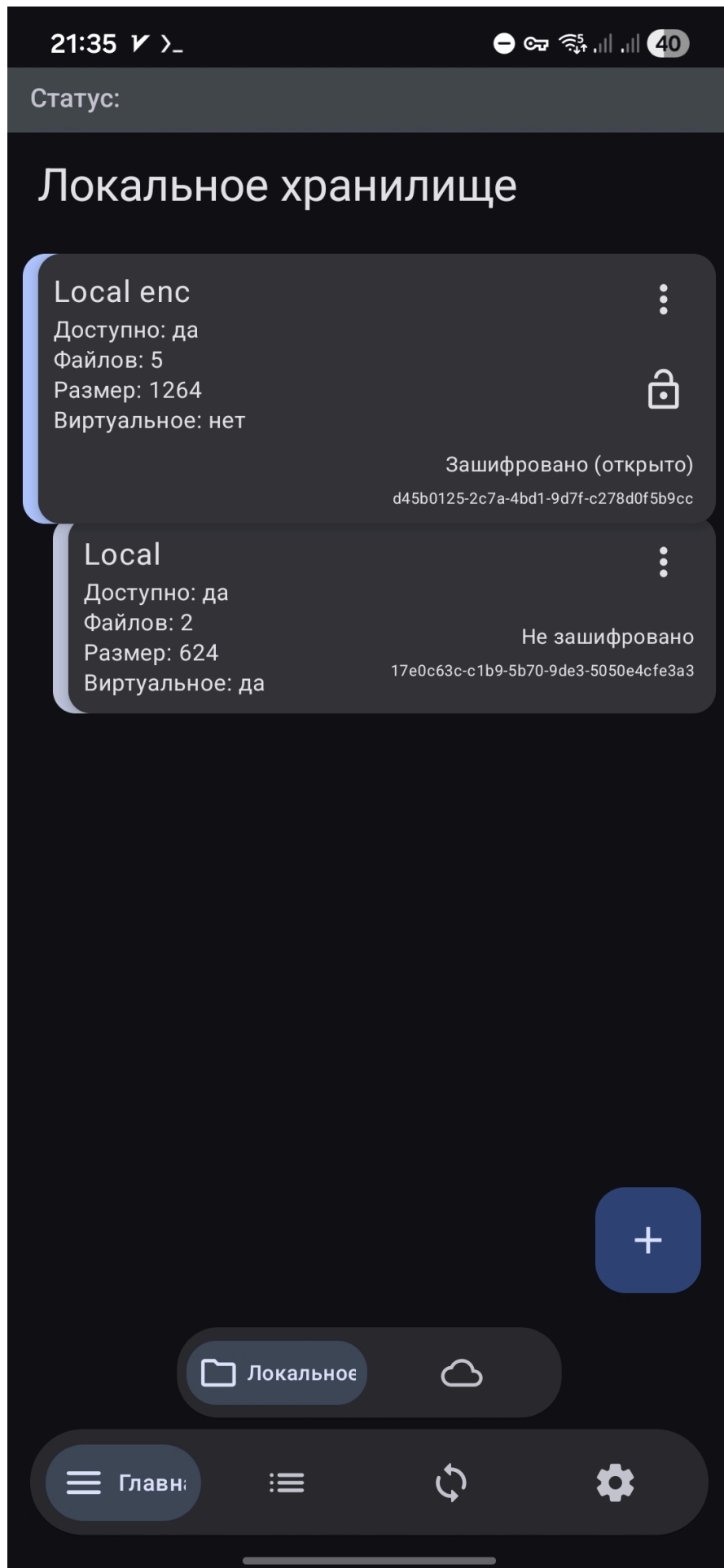


Рисунок В.1 — Локальные vault
759

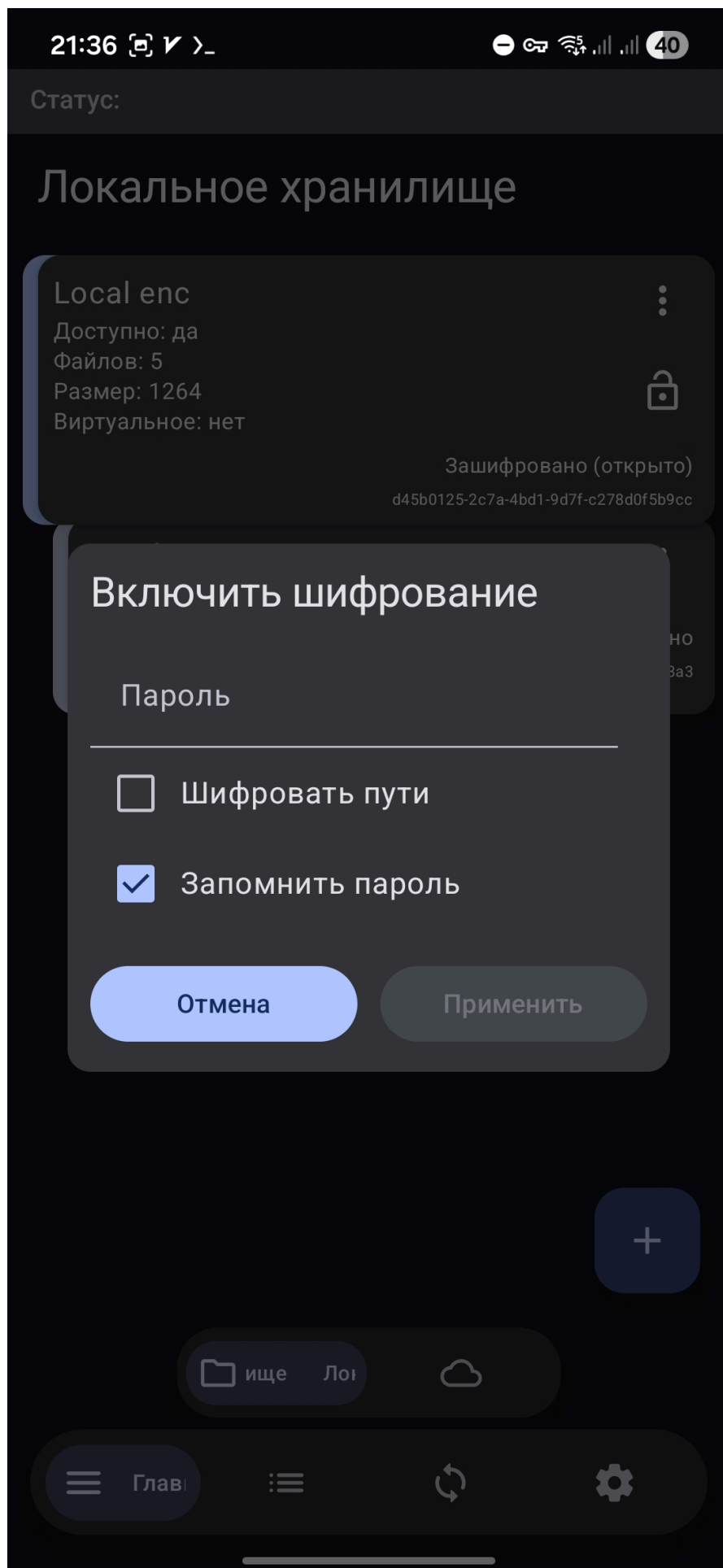


Рисунок В.2 — Диалог шифрования
760

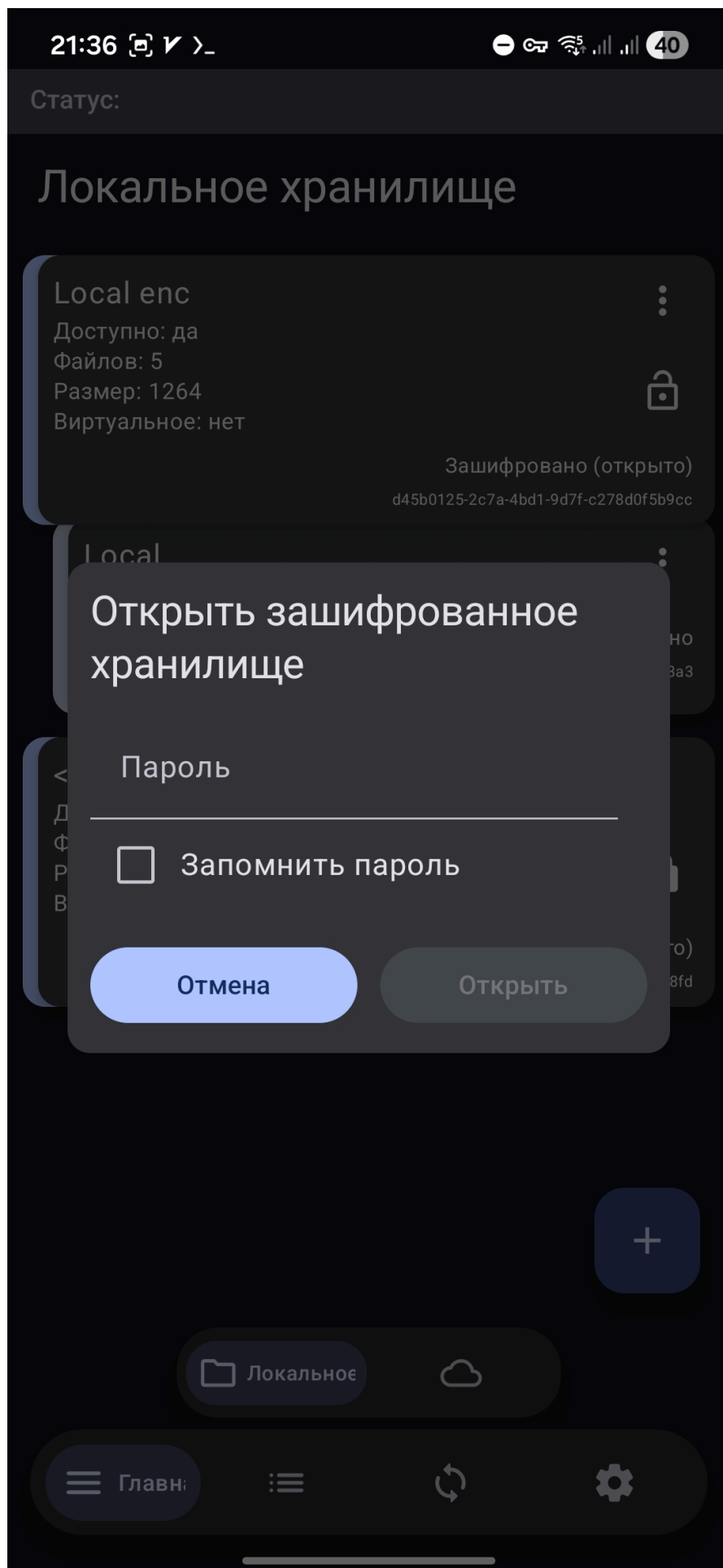


Рисунок В.3 — Открытие и закрытие vault
761

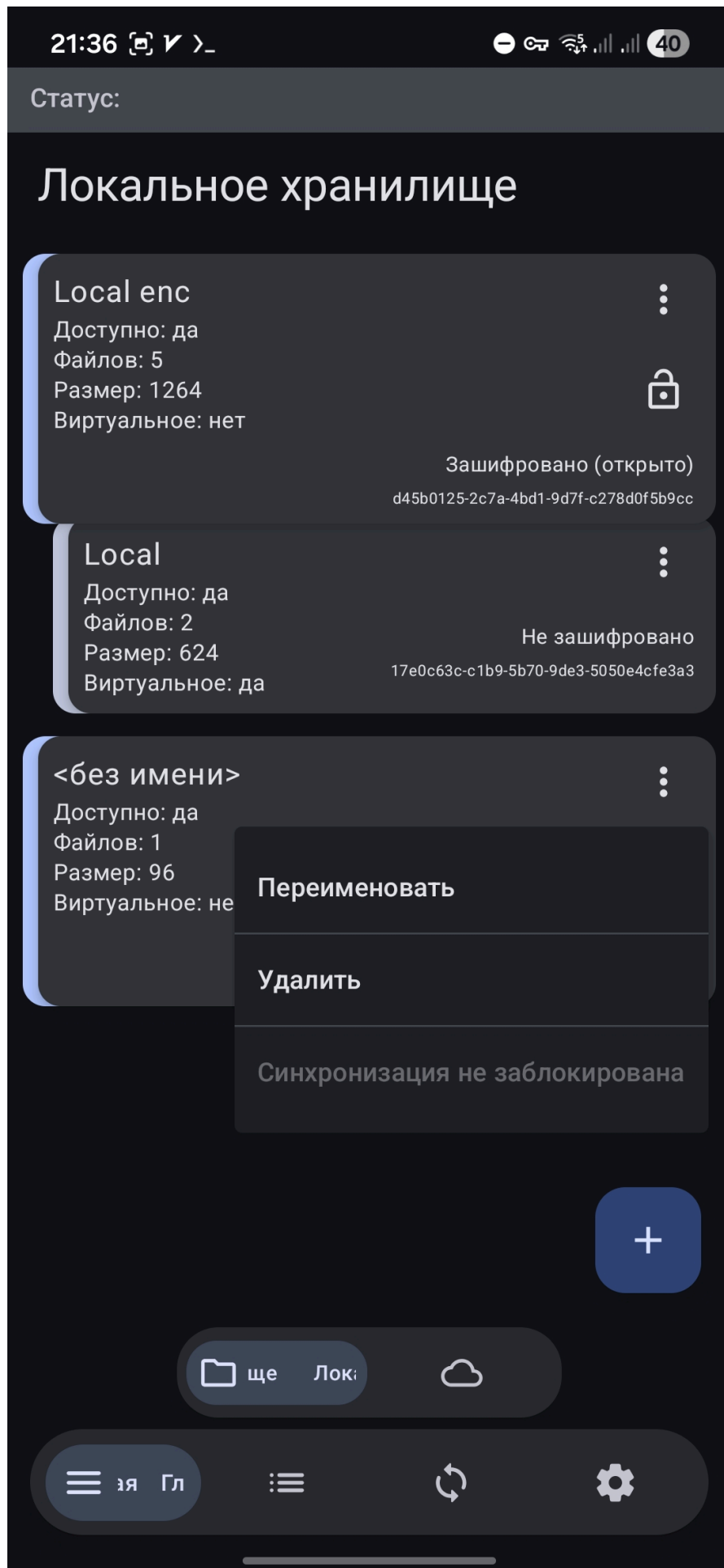


Рисунок В.4 — Переименование и удаление
762



Рисунок В.5 — Удалённые vault
763

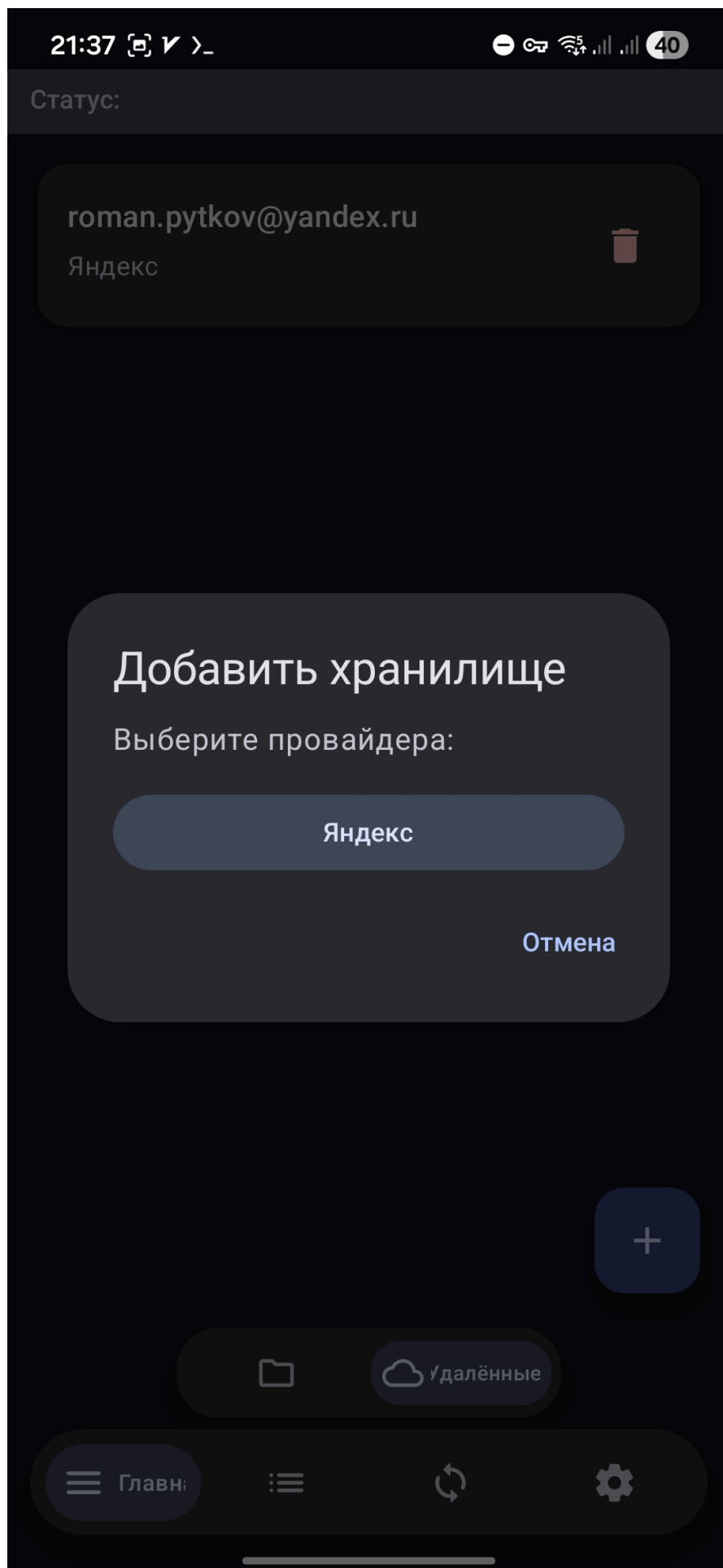


Рисунок В.6 — OAuth Яндекс
764

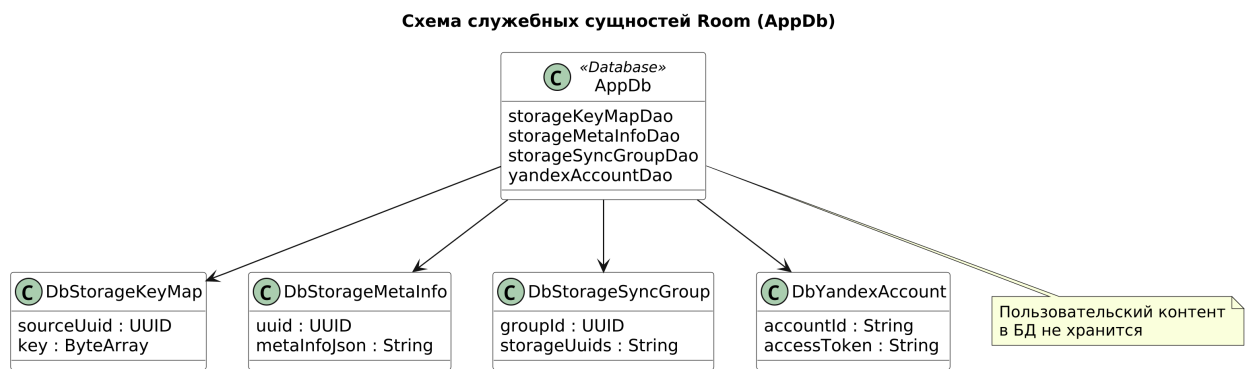


Рисунок В.7 — Схема Room

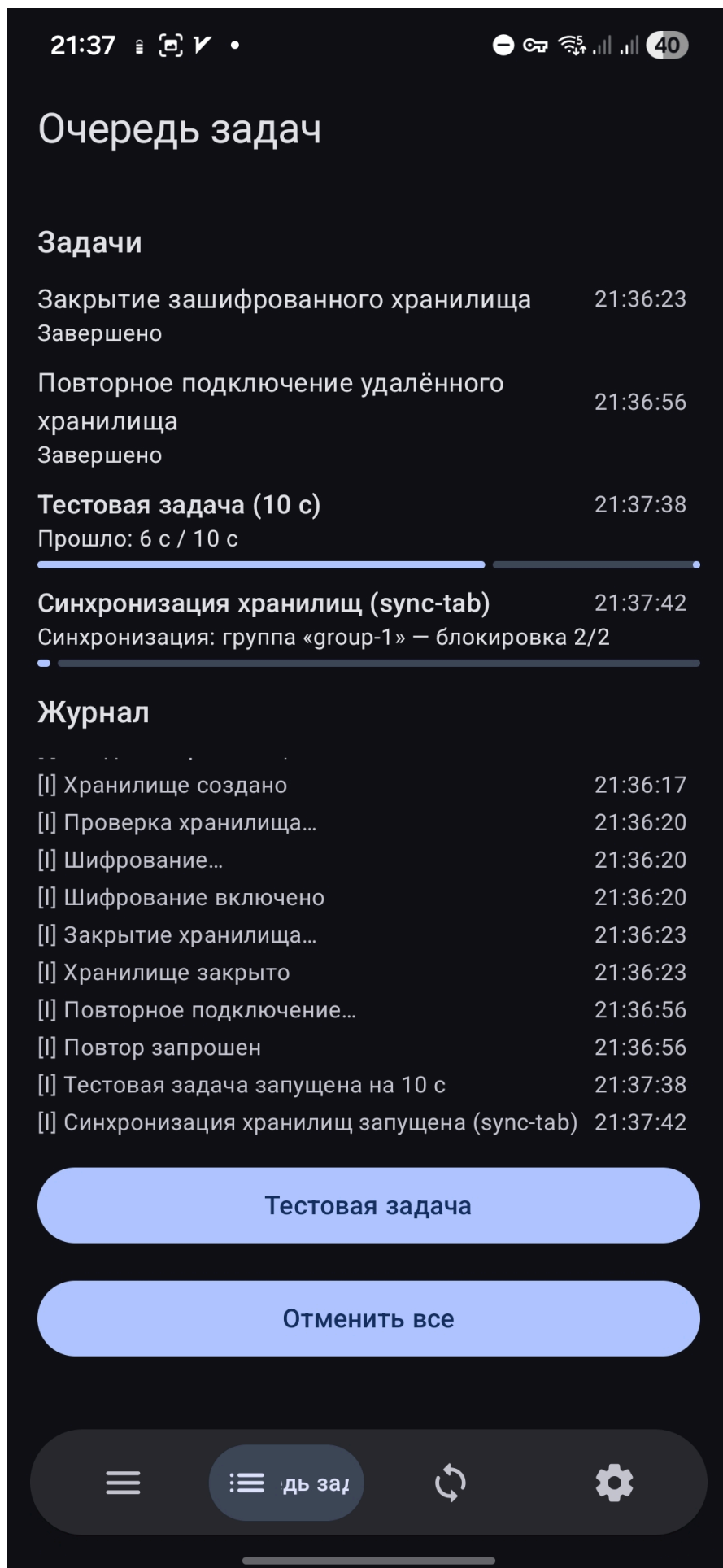


Рисунок В.8 — Экран задач
766

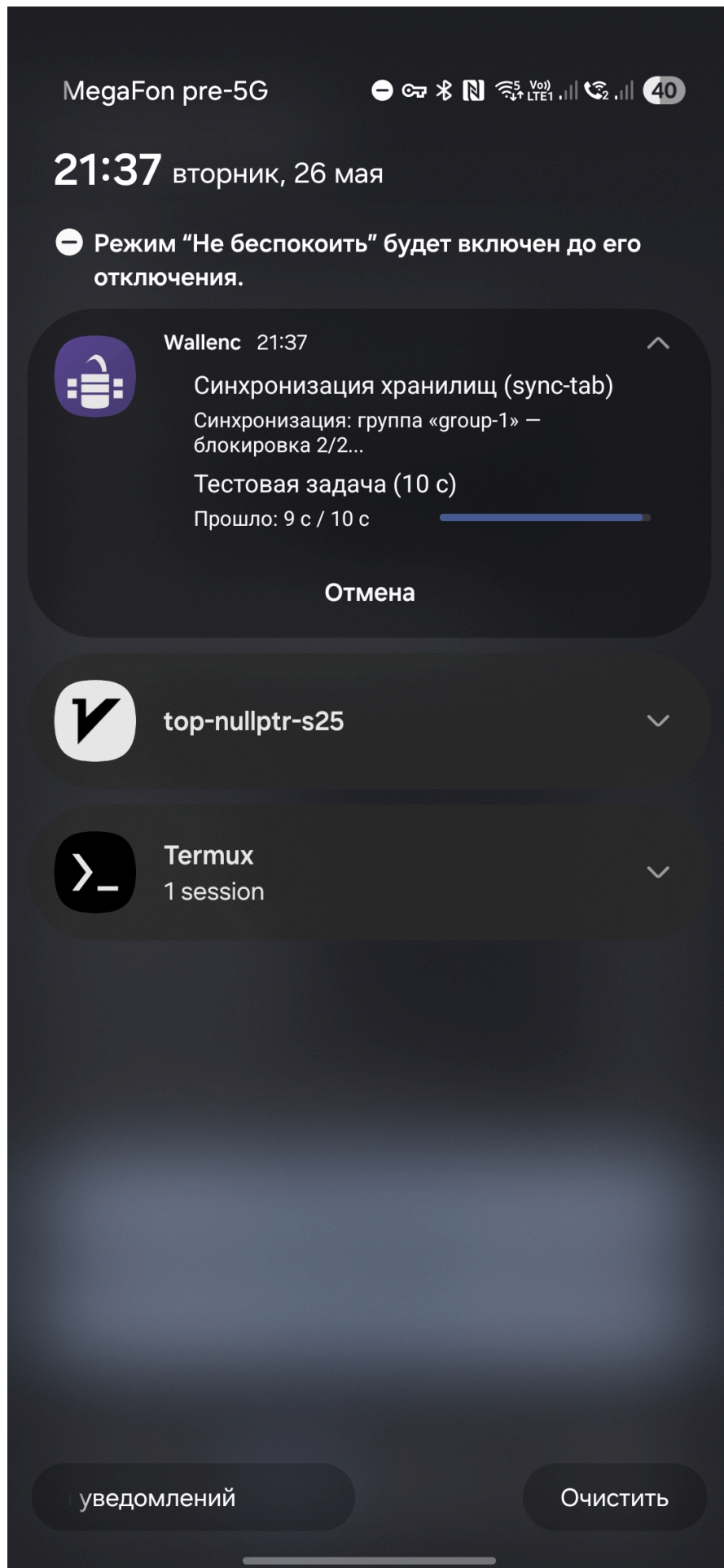


Рисунок В.9 — Уведомление о задачах
767

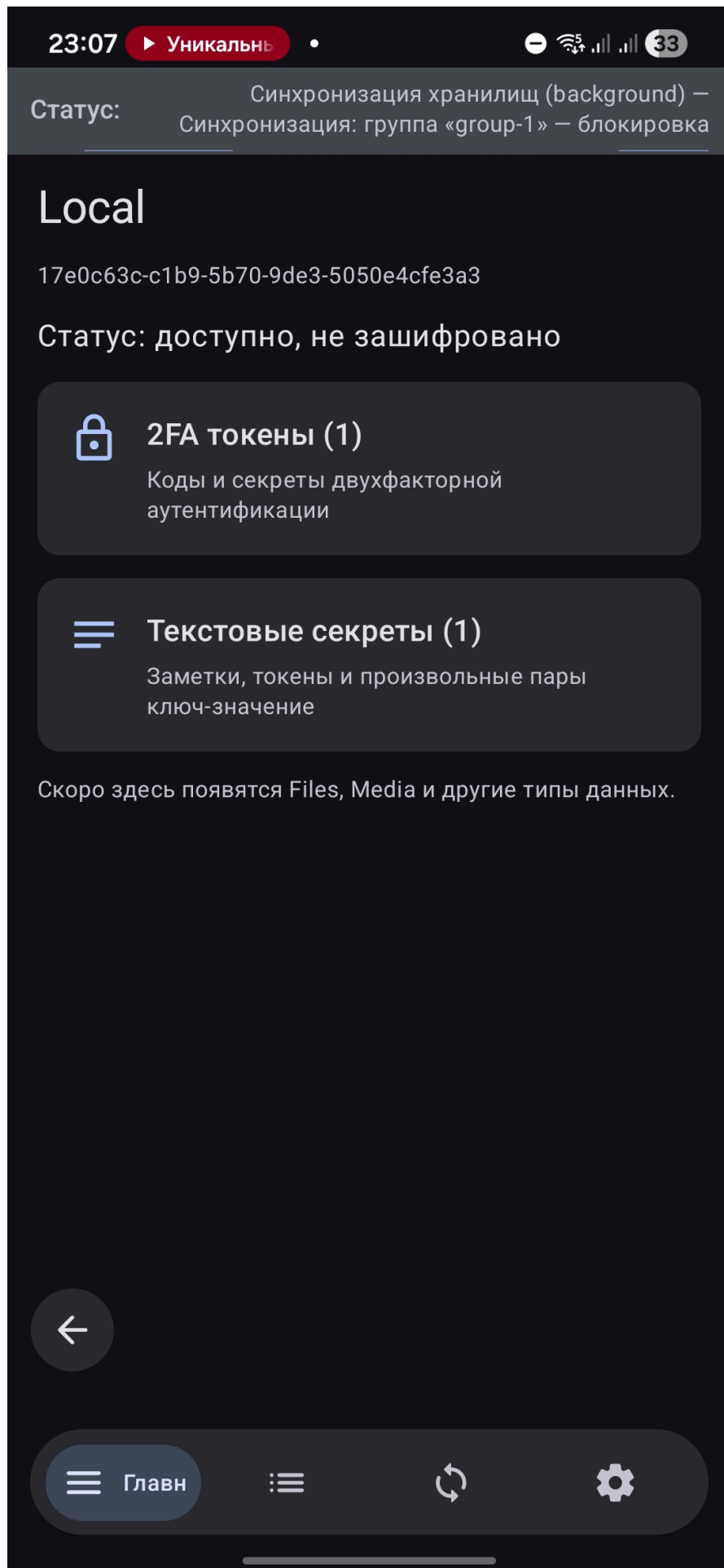


Рисунок В.10 — Экран storage: секреты и 2FA
768

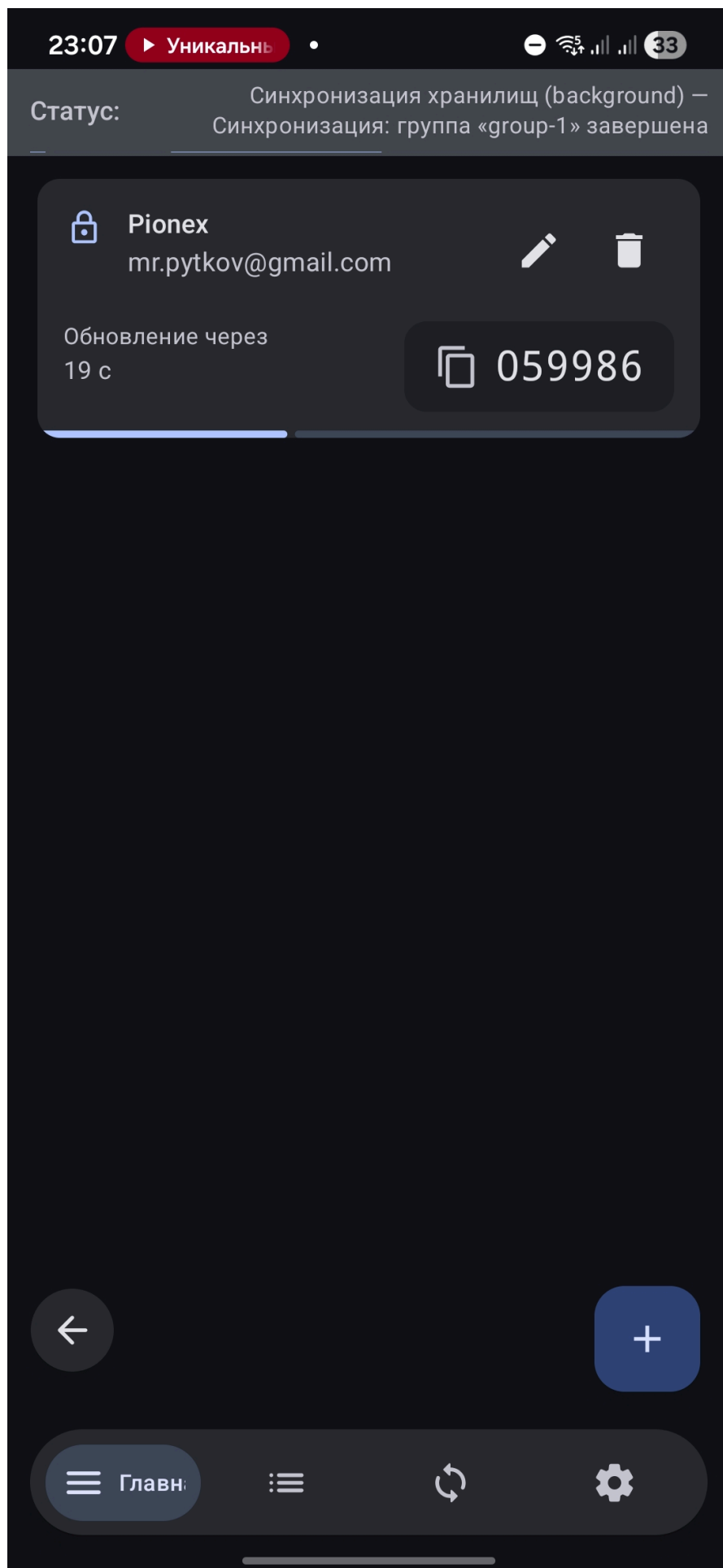


Рисунок В.11 — Экран 2FA с токеном
769