

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

РАЗРАБОТКА МУЛЬТИВАЛЮТНОГО
КРИПТОКОШЕЛЬКА ДЛЯ ПЛАТФОРМЫ
ANDROID С ИСПОЛЬЗОВАНИЕМ
СОВРЕМЕННЫХ МЕТОДОВ ЗАЩИТЫ
КРИПТОГРАФИЧЕСКИХ КЛЮЧЕЙ

Студент В.В. Байков 22.05.2026 Байков Алексей Владимирович
(подпись, дата) (ФИО полностью)

Руководитель А.Н. Токарев 22.05.2026 Токарев А.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр А.С. Бычков 22.05.2026 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой М.А. Митрохин Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от
19 июня 2026 г. № 40)

Секретарь ГЭК Е.И. Гурий Гурий Е.И.
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«07» апреля 2026 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Байков Алексей Владимирович гр. 22ВВП2 факультета вычислительной техники

2. Тема работы: «Разработка мультивалютного криптокошелька для платформы Android с использованием современных методов защиты криптографических ключей»

Тема утверждена приказом ПГУ № 347/о от "07" апреля 2026 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Токарев А.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать некастодиальный криптокошелек для мобильных устройств на ОС Android, соответствующий стандартам безопасности и мультивалютности. Приложение должно позволять пользователям осуществлять исходящие транзакции, отслеживать поступления средств и гибко настраивать пользовательский интерфейс. Реализация графического интерфейса пользователя, управление криптовалютными активами (просмотр баланса, история транзакций, отправка и получение средств), а также взаимодействие с блокчейн-сетями через API-запросы. Обеспечение некастодиального хранения приватных ключей, генерация мнемонической фразы по стандарту BIP39, создание и защита ключей с использованием симметричного шифрования AES-256 в режиме CBC с PKCS5Padding, вектора инициализации (IV) и хранение данных в безопасном хранилище Android.

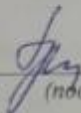
5. Перечень подлежащих разработке вопросов

Требуется провести анализ предметной области и изучить существующие решения в сфере криптокошельков. Спроектировать архитектуру мобильного приложения, определяющую взаимодействие интерфейса, логику управления активами и криптографического модуля. Спроектировать локальную базу данных и разработать программное обеспечение для приёма и передачи данных по API. Выполнить тестирование системы.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	45	07.04.26 – 13.04.26	
2	Анализ темы, аналогов, постановка задачи	24	13.04.26 – 17.04.26	
3	Выбор и настройка инструментов разработки	50	17.04.26 – 24.04.26	
4	Проектирование архитектуры	30	24.04.26 – 28.04.26	
5	Разработка и тестирование программы	151	28.04.26 – 17.05.26	
6	Оформление пояснительной записки	24	17.05.26 – 21.05.26	

Дата выдачи задания "07" апреля 2026 г.

Руководитель бакалаврской работы  Токарев А.Н.
(подпись, дата)

Задание к исполнению принял студент  Байков А.В.
(подпись, дата)

Работу к защите допустить
Декан факультета  17.06.2026 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 231 листов, 39 рисунков, 6 таблиц, 5 источников, 3 приложения.

ANDROID, KOTLIN, JAVA, BTC, ETH, TON, USDT, SOL, ANDROID STUDIO, BIP39, AES-256, RSA

Цель работы – разработать приложение, которое будет соответствовать стандартам безопасности и мультивалютности в формате некастодиального криптокошелька с возможностью осуществления исходящих транзакций, отслеживания поступлений и гибкой настройкой пользовательского интерфейса.

Объектами разработки являются:

- клиентское мобильное приложение для ОС Android, реализующее графический интерфейс пользователя, управление криптовалютными активами (просмотр баланса, история транзакций, отправка и получение средств), а также взаимодействие с блокчейн-сетями через API-запросы;

- криптографический модуль, обеспечивающий некастодиальное хранение приватных ключей: генерацию мнемонической фразы по стандарту BIP39, создание и защиту ключей с использованием симметричного шифрования AES-256 в режиме CBC с PKCS5Padding, вектора инициализации (IV) и хранение зашифрованных данных в безопасном хранилище Android.

В результате проделанной работы разработано приложение, полностью соответствующее поставленной задаче, а именно: реализовано клиентское мобильное приложение для ОС Android, поддерживающее мультивалютность (BTC, ETH, TON, USDT, SOL), возможность осуществления исходящих транзакций, отслеживания поступлений и гибкой настройки пользовательского интерфейса.

					ПГУ 1.090301.29.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Разработка мультивалютного криптокошелька для платформы Android с использованием современных методов защиты криптографических ключей Пояснительная записка	Лит.	Лист	Листов
Разраб.		Бейков А.В.		21.05.24			2	231
Провер.		Токарев А.Н.		21.05.24				
Н. Контр.		Бычков А.С.		21.05.24				
Утверд.		Митрохин		21.05.24				
						ФВТ, каф. ВТ гр. 22ВВП2		

Содержание

Перечень сокращений и обозначений.....	4
Введение	6
1. Разработка мультивалютного криптокошелька для платформы Android с использованием современных методов защиты криптографических ключей.	7
1.1 Анализ предметной области	7
1.1.1 Анализ аналогов.....	8
1.1.2 Постановка задачи	12
1.2 Выбор инструментов разработки	14
1.2.1 Выбор среды разработки клиентской части	14
1.2.2 Выбор среды разработки серверной части	17
1.2.3 Выбор языка программирования	19
1.2.4 Выбор архитектуры приложения	21
1.3 Реализация приложения.....	25
1.3.1 Графический интерфейс	25
1.3.2 Java/Kotlin-классы	28
1.3.3 Структура приложения	34
1.4 Тестирование	38
2 Безопасность жизнедеятельности	65
2.1 Требования к организации рабочего места	65
2.2 Требования к освещению.....	66
2.3 Требования к режиму отдыха	67
Заключение.....	68
Список использованных источников	69
Приложение А – UML-диаграммы.....	70
Приложение Б – Листинг программы	75
Приложение В – Презентация	224

Перечень сокращений и обозначений

В данной работе используется следующие сокращения и обозначения.

ОС – операционная система

ПК – персональный компьютер

AES (Advanced Encryption Standard) – стандарт симметричного шифрования

API (Application Programming Interface) – программный интерфейс приложения

APK (Android Package Kit) – пакет приложений для Android

ART (Android Runtime) – среда выполнения приложений Android

BIP39 (Bitcoin Improvement Proposal 39) – стандарт для генерации мнемонической фразы (сид-фразы)

BTC (Bitcoin) – биткоин, криптовалюта

CBC (Cipher Block Chaining) – режим шифрования сцепления блоков

CPU (Central Processing Unit) – центральный процессор

dApps (Decentralized Applications) – децентрализованные приложения

EIP1559 (Ethereum Improvement Proposal 1559) – предложение по улучшению Ethereum, меняющее механизм комиссий

ERC20 (Ethereum Request for Comments 20) – стандарт токенов в блокчейне Ethereum

ETH (Ethereum) – эфириум, криптовалюта и платформа

GUI (Graphical User Interface) – графический интерфейс пользователя

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

IDE (Integrated Development Environment) – интегрированная среда разработки

IOS (iPhone Operating System) – операционная система для устройств Apple

IV (Initialization Vector) – вектор инициализации (в криптографии)

JSON (JavaScript Object Notation) – текстовый формат обмена данными

JVM (Java Virtual Machine) – виртуальная машина Java

MVVM (Model-View-ViewModel) – архитектурный паттерн для пользовательских интерфейсов

					ПГУ 1.090301.29.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

PIN (Personal Identification Number) – персональный идентификационный номер

QR (Quick Response) – быстрый отклик (тип матричного штрихкода)

RAM (Random Access Memory) – оперативная память

RSA (Rivest–Shamir–Adleman) – криптографический алгоритм с открытым ключом

SDK (Software Development Kit) – комплект для разработки программного обеспечения

SHA (Secure Hash Algorithm) – алгоритм безопасного хеширования

SOL (Solana) – криптовалюта и блокчейн-платформа

TON (The Open Network) – блокчейн-платформа и криптовалюта

UI (User Interface) – пользовательский интерфейс

UML (Unified Modeling Language) – унифицированный язык моделирования

USDT (Tether USD) – стейблкоин, привязанный к доллару США

WORA (Write Once, Run Anywhere) – принцип «напиши однажды, запускай где угодно»

XML (Extensible Markup Language) – расширяемый язык разметки

					ПГУ 1.090301.29.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В условиях стремительной цифровизации мировой экономики и повсеместного распространения мобильных технологий, децентрализованные цифровые активы, известные как криптовалюты, заняли прочное место в структуре личных финансов миллионов людей по всему миру. Биткоин, Эфириум и стейблкоины, такие как USDT, перестали быть инструментом исключительно технических энтузиастов, трансформировавшись в полноценный класс инвестиционных и расчетных средств. По данным аналитических агентств, число пользователей криптовалют в 2026 году превысило отметку в 700 миллионов человек, при этом более двух третей всех операций инициируется с портативных устройств под управлением операционной системы Android. В отличие от традиционных банковских сервисов, где ответственность за сохранность средств лежит на финансовом учреждении, особенностью криптовалютной экосистемы является принцип некастодиальности: владелец актива самостоятельно несет полную ответственность за хранение и защиту приватных криптографических ключей. Компрометация или утрата ключа влечет за собой безвозвратную потерю средств без возможности апелляции к третьей стороне.

Основная цель данной выпускной квалификационной работы заключается в разработке некастодиального мультивалютного криптокошелька для платформы Android, реализующего современные методы защиты криптографических ключей.

Результатом выполнения выпускной квалификационной работы станет готовый к установке программный продукт в формате APK, отвечающий актуальным требованиям в области мобильной разработки и информационной безопасности. Практическая значимость работы заключается в создании инструмента, который позволит пользователям безопасно и удобно управлять личными цифровыми активами в условиях растущей сложности и волатильности криптовалютного рынка.

					ПГУ 1.090301.29.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1. Разработка мультивалютного криптокошелька для платформы Android с использованием современных методов защиты криптографических ключей.

1.1 Анализ предметной области

Ключевым трендом последних лет является смещение пользовательской активности в мобильный сегмент: более 65% всех операций с цифровыми активами инициируется с портативных устройств под управлением операционных систем Android и IOS.

Разработка Android-приложения для управления криптовалютами активами является сложной междисциплинарной задачей, лежащей на стыке финансовых технологий (FinTech), информационной безопасности и проектирования мобильных интерфейсов. В отличие от традиционных банковских приложений, где ответственность за сохранность средств несет финансовое учреждение, криптокошелек работает по принципу некастодиальности (Self-Custody): приватные ключи, являющиеся единственным доказательством права собственности на активы в блокчейне, генерируются и хранятся исключительно на устройстве пользователя. Утрата или компрометация ключа влечет за собой безвозвратную потерю средств без возможности апелляции к третьей стороне.

При проведении анализа предметной области можно выделить несколько критически значимых аспектов, которые определяют качество и востребованность подобного программного продукта:

– функциональность и мультивалютность. Приложение должно обеспечивать полноценный цикл работы с несколькими независимыми блокчейн-сетями. На текущий момент де-факто стандартом является поддержка Биткоина (BTC) как основного средства сбережения, Эфириума (ETH) как платформы для смарт-контрактов и токенов стандарта ERC-20, в частности стейблкоина USDT, обеспечивающего связь с фиатной экономикой.

					ПГУ 1.090301.29.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

– функционал должен включать генерацию адресов, формирование и подпись транзакций с учетом актуальных механизмов расчета комиссий (например, EIP-1559 для сети Ethereum), а также отображение детальной истории операций с указанием статуса подтверждения в сети;

– безопасность и управление доступом. Учитывая необратимость транзакций в блокчейне, защита ключевой информации выходит на первый план. Приложение должно реализовывать многоуровневую систему локальной защиты: шифрование приватных ключей и seed на внутреннем хранилище устройства, запрет на автоматическое облачное резервирование конфиденциальных данных, обязательную аутентификацию пользователя с использованием PIN-кода и биометрических данных;

– удобство использования и эргономика. Финансовые приложения требуют интуитивно понятного интерфейса, минимизирующего риск пользовательской ошибки. Важными элементами являются: интеграция сканера QR-кода для безопасного ввода адреса получателя (исключение атак подмены буфера обмена), режим скрытия баланса (замена сумм символами «?») для обеспечения приватности в публичных местах), а также гибкие настройки визуального оформления (тема, цвет акцентных элементов и текста);

– интернационализация. Криптовалютный рынок глобален, и успешное приложение должно быть ориентировано на широкую международную аудиторию. Поддержка нескольких языковых локалей (русский, английский, китайский, хинди, испанский) является обязательным требованием для обеспечения конкурентоспособности продукта.

Таким образом, Android-приложение «BETUS» представляет собой высокотехнологичный программный продукт, призванный обеспечить безопасное, удобное и персонализированное управление цифровыми активами в условиях растущей сложности и волатильности криптовалютного рынка.

1.1.1 Анализ аналогов

Для определения ниши разрабатываемого приложения и обоснования его

					ПГУ 1.090301.29.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

конкурентных преимуществ был проведён сравнительный анализ трёх наиболее репрезентативных некастодиальных криптовалютных кошельков:

- Metamask;
- Phantom;
- Tonkeeper.

Рассмотрим приложение – MetaMask (рисунок 1).

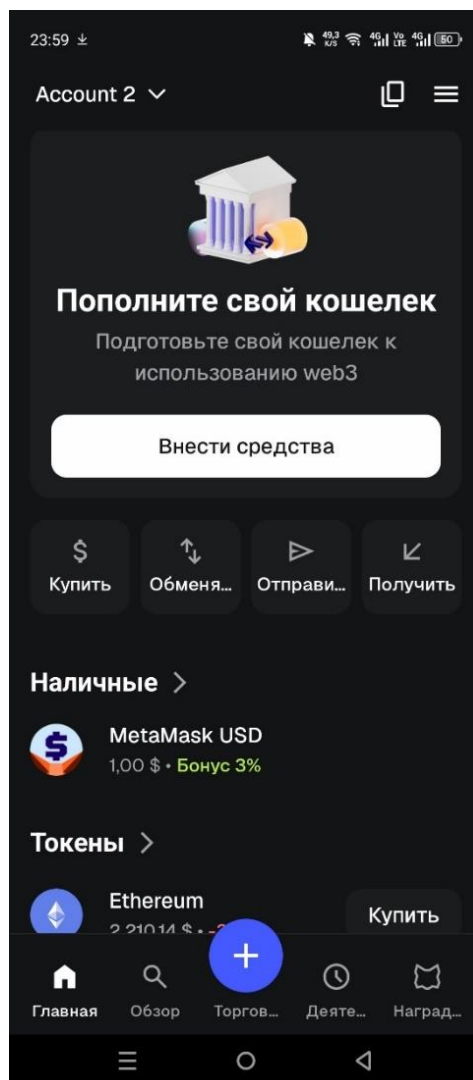


Рисунок 1 – Интерфейс приложения «MetaMask»

Рассмотрим плюсы и минусы данного приложения.

Плюсы:

- полноценная поддержка Ethereum и токенов стандарта ERC-20;
- встроенный браузер для децентрализованных приложений (dApps);

					ПГУ 1.090301.29.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

– генерация и хранение seed (BIP39) и приватных ключей с использованием локального шифрования;

– аутентификация по биометрии.

Минусы:

– отсутствие нативной поддержки Биткоина;

– ограниченная персонализация интерфейса;

– перегруженность интерфейса;

– зависимость от инфраструктуры Infura.

Рассмотрим приложение – Phantom (рисунок 2).

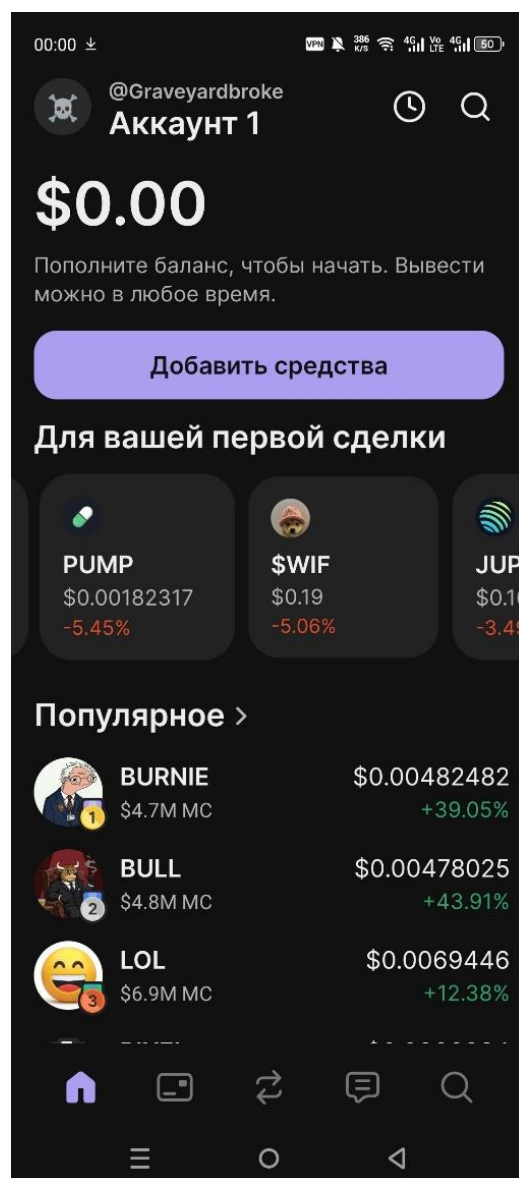


Рисунок 2 – Интерфейс приложения «Phantom»

					ПГУ 1.090301.29.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

Рассмотрим плюсы и минусы данного приложения.

Плюсы:

- бесшовная работа с нативным токеном SOL и стандартами SPL;
- встроенный обмен токенов и стейкинг SOL;
- поддержка аппаратных кошельков Ledger;
- высокая производительность и отзывчивость интерфейса.

Минусы:

- моноэкосистемность;
- сложности интеграции с другими сетями;
- отсутствие тонкой настройки приватности.

Рассмотрим приложение – Tonkeeper (рисунок 3).

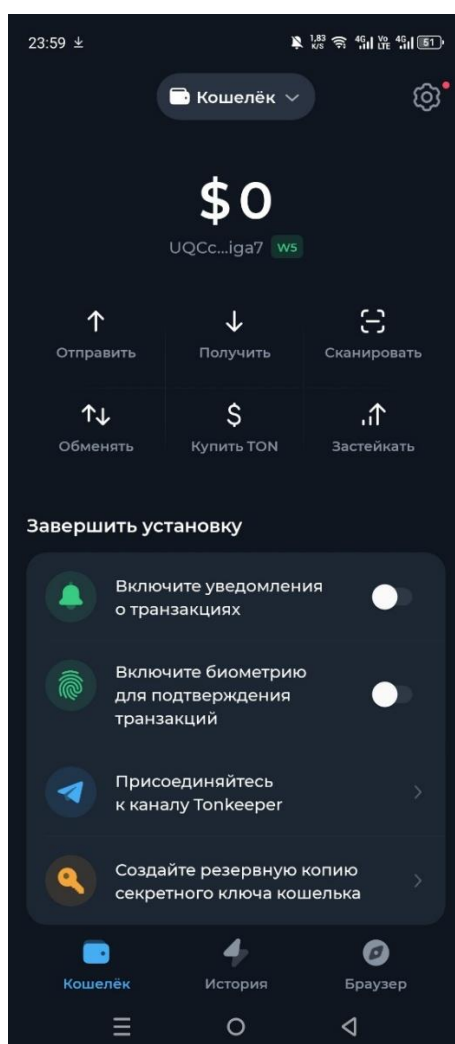


Рисунок 3 – Интерфейс приложения «Tonkeeper»

					ПГУ 1.090301.29.001 ПЗ	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

Рассмотрим плюсы и минусы данного приложения.

Плюсы:

- работа с нативной монетой TON и токенами стандарта Jettons (включая USDT на сети TON);
- интеграция с сервисами Telegram (TON DNS, TON Storage, фрагменты анонимных номеров);
- низкие комиссии за транзакции внутри сети TON;
- поддержка биометрической аутентификации.

Минусы:

- отсутствие поддержки классических криптоактивов;
- техническая сложность кроссчейн-интеграции.

1.1.2 Постановка задачи

Целью выпускной квалификационной работы является разработка Android-приложения «BETUS», предназначенного для безопасного управления цифровыми активами в сетях Bitcoin, Ethereum и токенах стандарта ERC-20 (USDT).

Приложение должно представлять собой некастодиальное программное решение, в котором приватные ключи генерируются, хранятся в зашифрованном виде и обрабатываются исключительно на устройстве пользователя без передачи на внешние серверы. Основной упор при проектировании делается на обеспечение информационной безопасности, удобство пользовательского интерфейса и возможность тонкой персонализации внешнего вида и поведения приложения.

Функциональные требования приложения:

- возможность создания нового кошелька или привязка существующего по seed фразе;
- реализация безопасного хранения и защиты конфиденциальных данных;
- реализация функционала отправки и получения криптовалюты с использованием QR-кода;
- возможность просмотра истории транзакций и ее очищение;

					ПГУ 1.090301.29.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

- обеспечение приватности отображения данных (скрытие финансовых показателей);
- поддержка мультиязычности (интернационализация);
- наличие инструментов управления учетной записью и безопасностью (кастомизация внешнего вида приложения и смена PIN-кода);
- приложение должно обеспечивать корректную работу с базовыми криптоактивами (BTC, ETH, USDT);
- отображение актуальной стоимости активов с использованием API запросов с разбивкой по токенам и общей суммой;
- наличие сервера для централизованного управления API запросами.

На рисунке 4 представлена диаграмма использования приложения.

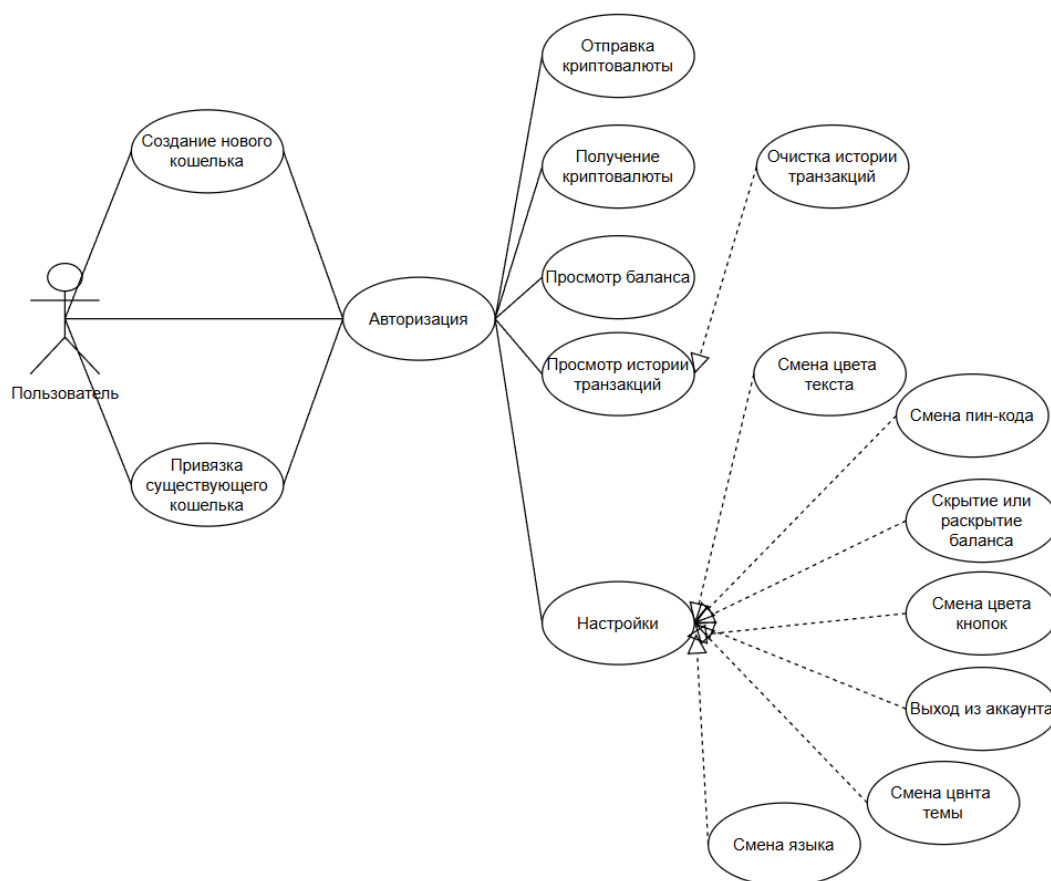


Рисунок 4 – UML-диаграмма прецедентов

Технический требования приложения:

- наличие интернета для осуществления отправки токенов, отображения

актуальной стоимости активов и отображения новых транзакций;

- мобильное устройство с операционной системой Android версии не ниже 6.0 (API 23);
- наличие сканера отпечатка пальца для аутентификации;
- наличие камеры (опционально) для осуществления транзакций с помощью QR кода;
- наличие 2 ГБ оперативной памяти (RAM) и около 100 МБ свободного места во внутреннем хранилище для кэширования данных;
- выданные разрешения для камеры.

1.2 Выбор инструментов разработки

В данном разделе будут рассмотрены и обоснованы основные инструменты разработки, использованные при реализации программного комплекса «Мультивалютный криптокошелек BETUS». Комплекс состоит из двух компонентов: клиентского Android-приложения и вспомогательного серверного модуля для агрегации данных о рыночных курсах. Для каждого компонента был проведён сравнительный анализ доступных сред разработки и выбран оптимальный инструментарий.

1.2.1 Выбор среды разработки клиентской части

Ключевым этапом перед началом реализации мобильного приложения является выбор интегрированной среды разработки (IDE). Были рассмотрены три наиболее распространённые среды, применяемые для создания Android-приложений: Microsoft Visual Studio, IntelliJ IDEA и Android Studio.

1.2.1.1 Microsoft Visual Studio

Microsoft Visual Studio – это многофункциональная интегрированная среда разработки корпорации Microsoft, предназначенная для создания, отладки и развёртывания программного обеспечения на различных языках программирования и платформах. Благодаря расширению Xamarin среда

позволяет вести кросс-платформенную мобильную разработку, включая сборку под Android [1].

Преимущества Microsoft Visual Studio:

- поддержка широкого спектра языков программирования (C#, C++, F#, JavaScript и др.) и платформ;
- глубокая интеграция с облачными сервисами Microsoft Azure и системой контроля версий Azure DevOps;
- мощные инструменты профилирования и отладки кода, включая анализ производительности и потребления памяти.

Недостатки:

- высокие системные требования, что может привести к снижению производительности при работе с эмуляторами Android на компьютерах средней мощности;
- сложность настройки окружения для нативной Android-разработки на Java/Kotlin по сравнению со специализированными средами;
- ограниченная функциональность бесплатной версии (Community Edition), а также необходимость приобретения платной лицензии для доступа к расширенным корпоративным возможностям.

1.2.1.2 IntelliJ IDEA

IntelliJ IDEA – это интеллектуальная среда разработки от компании JetBrains, ориентированная преимущественно на языки Java и Kotlin. Она является базовой платформой для Android Studio и широко известна своими возможностями статического анализа и рефакторинга кода.

Преимущества IntelliJ IDEA:

- превосходная поддержка языков Java и Kotlin, включая интеллектуальное автодополнение, анализ «на лету» и безопасный рефакторинг;
- гибкая система плагинов, позволяющая расширить функциональность среды под конкретные задачи;
- единая экосистема с другими продуктами JetBrains, что удобно при работе

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		15

над многомодульными проектами.

Недостатки:

- высокие системные требования, аналогичные Android Studio, что может затруднить работу на устаревшем оборудовании;
- избыточный функционал и сложный интерфейс для начинающих разработчиков; настройка окружения для Android-разработки требует установки и конфигурации дополнительных плагинов;
- полнофункциональная версия Ultimate является платной, а бесплатная Community Edition не включает в себя специализированные инструменты для Android-разработки, присутствующие в Android Studio.

1.2.1.3 Android Studio

Android Studio – это официальная интегрированная среда разработки для платформы Android, созданная компанией Google на основе открытой версии IntelliJ IDEA. Она предоставляет полный набор инструментов, необходимых для проектирования, кодирования, отладки и тестирования мобильных приложений.

Преимущества:

- полная и бесшовная интеграция с Android SDK, эмуляторами устройств и утилитами командной строки (ADB, Gradle);
- встроенный редактор макетов (Layout Editor) с поддержкой визуального конструирования интерфейсов (Drag-and-Drop) и предпросмотра на различных конфигурациях экранов;
- мощный профилировщик производительности (CPU Profiler, Memory Profiler), необходимый для оптимизации криптографических операций и сетевых запросов;
- нативная поддержка системы сборки Gradle, что упрощает управление сторонними библиотеками (Web3j, BitcoinJ, BouncyCastle), используемыми в проекте;
- регулярные обновления от Google, обеспечивающие совместимость с новейшими версиями API.

Недостатки:

- высокое потребление оперативной памяти и ресурсов процессора, особенно при запуске нескольких эмуляторов одновременно;
- ориентация исключительно на платформу Android и ограниченная поддержка других языков и технологий за пределами мобильной экосистемы.

В качестве среды разработки клиентской части проекта была выбрана Android Studio. Данный выбор обусловлен необходимостью использования специализированных инструментов для работы с Android SDK, встроенной поддержкой языка Kotlin (на котором написан проект) и удобным управлением зависимостями через Gradle, что критически важно при интеграции множества криптографических и сетевых библиотек [3].

1.2.2 Выбор среды разработки серверной части

Серверная часть приложения представляет собой легковесный модуль на языке Java, предназначенный для агрегации данных о рыночных курсах и кэширования информации, получаемой от публичных API. Для его реализации были рассмотрены две популярные бесплатные среды: Eclipse IDE и Apache NetBeans [2].

1.2.2.1 Eclipse IDE

Eclipse – это модульная среда разработки с открытым исходным кодом, исторически являющаяся одной из первых полноценных платформ для Java-разработки. Она обладает гибкой архитектурой, основанной на плагинах OSGi.

Преимущества:

- широкая распространённость и огромное количество обучающих материалов и плагинов в открытом доступе;
- относительно низкое потребление системных ресурсов по сравнению с аналогами;
- бесплатное распространение и отсутствие лицензионных ограничений.

Недостатки:

- устаревший и перегруженный интерфейс, требующий привыкания и

дополнительной настройки рабочих перспектив;

- менее интеллектуальное автодополнение и статический анализ кода по сравнению с продуктами JetBrains и NetBeans;
- сложности с настройкой сборки и запуска проектов без использования Maven/Gradle (ручное управление classpath).

1.2.2.2 Apache NetBeans

Apache NetBeans – это свободная интегрированная среда разработки, поддерживаемая фондом Apache Software Foundation. Она изначально создавалась как инструмент для Java-разработки и предоставляет мощные средства для создания десктопных и веб-приложений.

Преимущества:

- встроенная поддержка системы сборки Maven и Ant «из коробки», что упрощает управление зависимостями серверного модуля;
- высокое качество редактора кода Java с функциями рефакторинга, поиска использования и навигации по иерархии классов;
- удобный визуальный конструктор Swing GUI (Matisse), который может быть использован при необходимости создания административной панели мониторинга;
- полностью бесплатная лицензия и стабильная работа на различных операционных системах.

Недостатки:

- меньшая популярность в корпоративной среде по сравнению с IntelliJ IDEA Ultimate;
- медленнее внедряет поддержку новейших версий спецификаций Jakarta EE по сравнению с конкурентами.

В качестве среды разработки серверной части был выбран Apache NetBeans. Данный выбор обоснован его простотой настройки для чистого Java SE проекта, отличной интеграцией с Maven (используется для управления зависимостями JSON-парсеров и HTTP-клиентов) и отсутствием необходимости приобретения

					ПГУ 1.090301.29.001 ПЗ	Лист
						18
Изм.	Лист	№ докум.	Подпись	Дата		

коммерческой лицензии.

1.2.3 Выбор языка программирования

В рамках данного проекта разработка велась для двух компонентов: клиентского Android-приложения и серверного модуля агрегации данных. Для каждого из них были проанализированы два ключевых языка, применяемых в экосистеме Java Virtual Machine (JVM): Java и Kotlin.

1.2.3.1 Java

Java – это объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems (ныне Oracle). С момента появления платформы Android и до 2019 года Java являлся основным языком для написания мобильных приложений под управление данной ОС. Благодаря многолетней истории использования, язык обладает развитой экосистемой, огромным количеством проверенных временем библиотек и одним из самых больших сообществ разработчиков в мире.

Преимущества языка Java:

- огромный объём накопленных знаний, учебных материалов и решений типовых проблем позволяет значительно ускорить процесс разработки и отладки кода;
- богатая экосистема библиотек. Для Java написано колоссальное количество open-source фреймворков и утилит, включая криптографические провайдеры (Bouncy Castle) и инструменты для работы с блокчейном (Web3j, BitcoinJ), активно используемые в настоящем проекте;
- высокая производительность и стабильность. Благодаря JIT-компиляции в виртуальной машине ART (Android Runtime) и зрелости компилятора, Java-код демонстрирует стабильную и высокую скорость выполнения, что критически важно для вычислительно ёмких операций, таких как подпись транзакций и генерация криптографических ключей;
- кроссплатформенность. Принцип «Write Once, Run Anywhere» (WORA) позволяет использовать один и тот же Java-код как на серверной стороне (в среде

					ПГУ 1.090301.29.001 ПЗ	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

JVM), так и в клиентском приложении, что было использовано при разработке серверного модуля на NetBeans.

Недостатки языка Java:

- многословность (Boilerplate Code). По сравнению с современными языками, синтаксис Java требует написания большого количества шаблонного кода для реализации простых конструкций (геттеры, сеттеры, конструкторы), что увеличивает объём исходного файла и усложняет его восприятие;
- проблема нулевых ссылок (NullPointerException). Отсутствие встроенной на уровне языка защиты от null-значений является частой причиной крашей приложений во время выполнения, что требует от разработчика дополнительных проверок и осторожности.

1.2.3.2 Kotlin

Kotlin – это современный статически типизированный язык программирования, разработанный компанией JetBrains. Начиная с конференции Google I/O 2019, Kotlin объявлен приоритетным языком для разработки Android-приложений (Kotlin-first). Он полностью совместим с Java на уровне байт-кода и призван устранить фундаментальные недостатки предшественника, обеспечивая более лаконичный и безопасный синтаксис.

Преимущества языка Kotlin:

- лаконичность синтаксиса. Код на Kotlin в среднем на 30-40% короче эквивалентного Java-кода, что повышает читаемость и снижает вероятность ошибок. В проекте это активно применяется при описании UI-логики в Activity и Fragment;
- встроенная защита от NullPointerException. Система типов Kotlin разделяет nullable и non-nullable переменные на уровне компиляции, что практически полностью исключает риск возникновения ошибки доступа к нулевой ссылке в рантайме;
- полная совместимость с Java (Interoperability). Kotlin позволяет беспрепятственно вызывать Java-код и использовать все существующие

библиотеки без дополнительных обёрток. В рамках проекта на Kotlin написаны новые модули (например, логика управления балансом и настройки интерфейса), в то время как Java-код отвечает за взаимодействие с тяжелыми криптографическими библиотеками;

- поддержка корутин (Coroutines). Наличие встроенного в язык механизма асинхронного программирования позволяет писать неблокирующий код в последовательном (императивном) стиле, что упрощает выполнение сетевых запросов к Etherscan и Blockstream без блокировки пользовательского интерфейса.

Недостатки языка Kotlin:

- скорость полной сборки. Компиляция Kotlin-проекта может выполняться немного дольше по сравнению с чистым Java-проектом аналогичного размера;

- меньший объём документации для узкоспециализированных задач.

Несмотря на стремительный рост популярности, в некоторых нишевых библиотеках (особенно старых версий криптографических фреймворков) примеры использования всё ещё преобладают на Java, что может потребовать дополнительного времени на адаптацию.

1.2.4 Выбор архитектуры приложения

Выбор архитектуры – это важный и необходимый этап разработки каждого приложения. С её помощью определяется структура, взаимодействие между компонентами приложения, а также общая управляемость проекта. При разработке мультивалютного криптокошелька была выбрана архитектура MVVM (Model-View-ViewModel), которая является одним из наиболее рекомендуемых и широко применяемых архитектурных шаблонов для платформы Android. Данный шаблон разделяет приложение на три основных компонента: модель, представление и модель представления [4].

Model (Модель) – это слой, отвечающий за представление данных и бизнес-логику приложения. В контексте разработанного кошелька модель включает в себя:

- менеджеры и репозитории, такие как BalanceManager,

TransactionHistoryManager,
BitcoinTransactionChecker;

EthereumTransactionChecker

и

- источники данных: публичные API блокчейн-обозревателей (Blockstream, Etherscan) для получения информации о транзакциях и собственный легковесный сервер для получения актуальных рыночных курсов;
- локальные хранилища: файловая система внутренней памяти устройства для зашифрованного хранения приватных ключей и seed, а также SharedPreferences для сохранения пользовательских настроек (тема, язык, цвета интерфейса) и служебных флагов.

Модель полностью независима от представления и не содержит ссылок на UI-компоненты. Её главной задачей является предоставление чистых, готовых к отображению данных через наблюдаемые объекты (LiveData, StateFlow) и выполнение операций по изменению состояния (например, подпись и отправка транзакции, шифрование ключей). Благодаря такой изоляции модель может быть легко протестирована и повторно использована в различных частях приложения.

View (Представление) – отвечает за отображение данных пользователю и обработку пользовательского ввода. В разработанном приложении слой представления реализован в виде набора Activity (например, MainWindow, HistoryAction, SettingsActivity) и кастомных элементов интерфейса. Представление подписывается на наблюдаемые данные, предоставляемые ViewModel, и автоматически обновляет пользовательский интерфейс при их изменении. В задачи представления также входит:

- отображение балансов, истории транзакций и QR-кодов;
- перехват действий пользователя (нажатия кнопок, ввод PIN-кода, сканирование QR-кодов) и передача команд во ViewModel;
- управление визуальными состояниями (индикаторы загрузки, сообщения об ошибках).

Ключевой особенностью представления является то, что оно не содержит бизнес-логики и не принимает решений о том, как обрабатывать данные. Вся

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

логика навигации и форматирования значений вынесена либо в ViewModel, либо в утилитарные классы, что делает слой представления максимально тонким и легким для модификации.

ViewModel (Модель представления) – является связующим звеном между моделью и представлением. Это компонент, который хранит и управляет данными, необходимыми для отображения на экране, и переживает изменения конфигурации устройства (например, поворот экрана). ViewModel получает запросы от представления (например, «обновить баланс» или «отправить транзакцию»), обращается к соответствующим компонентам модели для выполнения операции и предоставляет результат обратно представлению в виде реактивных потоков данных. Важно отметить, что ViewModel не содержит ссылок на конкретные View или Activity, что обеспечивает её независимость от жизненного цикла UI и предотвращает утечки памяти.

Взаимодействие между компонентами MVVM организовано следующим образом:

- представление (Activity) подписывается на LiveData или StateFlow во ViewModel;
- пользователь совершает действие;
- представление вызывает соответствующий метод ViewModel;
- ViewModel запускает асинхронную операцию, делегируя выполнение слою модели;
- модель выполняет сетевой запрос или криптографическую операцию, сохраняет результат в локальное хранилище (файл истории или SharedPreferences) и возвращает данные во ViewModel;
- ViewModel обновляет наблюдаемые поля данных;
- представление автоматически получает уведомление об изменении и перерисовывает интерфейс.

Преимущества использования архитектурного шаблона MVVM для разработанного криптокошелька включают:

– чёткое разделение ответственности. Код, отвечающий за шифрование ключей и сетевое взаимодействие, изолирован в модели; логика подготовки данных для UI – во ViewModel; а отрисовка – в представлении;

– устойчивость к изменениям конфигурации. Благодаря тому, что ViewModel не уничтожается при повороте экрана, данные о балансе и истории транзакций сохраняются без необходимости повторных запросов к сети;

– удобство тестирования. Бизнес-логика, инкапсулированная во ViewModel и модели, может быть протестирована с помощью модульных тестов (JUnit) без запуска эмулятора Android;

– снижение риска утечек памяти. Отсутствие прямых ссылок из ViewModel на Activity исключает случайное удержание уничтоженных UI-компонентов в памяти;

– поддержка реактивного программирования. Использование LiveData и корутин Kotlin позволяет строить асинхронные цепочки обработки данных, что особенно важно при работе с сетевыми запросами к блокчейн-API, время отклика которых может быть непредсказуемым.

Выбор в пользу архитектуры MVVM обусловлен не только её статусом рекомендуемого подхода для современной Android-разработки, но и спецификой реализуемого проекта. Некастодиальный характер кошелька требует максимальной надёжности и предсказуемости поведения приложения, а чёткое разделение слоёв позволяет минимизировать вероятность ошибок, связанных с нарушением жизненного цикла компонентов или некорректным управлением состоянием.

Схема паттерна архитектуры MVVM представлена на рисунке 5.

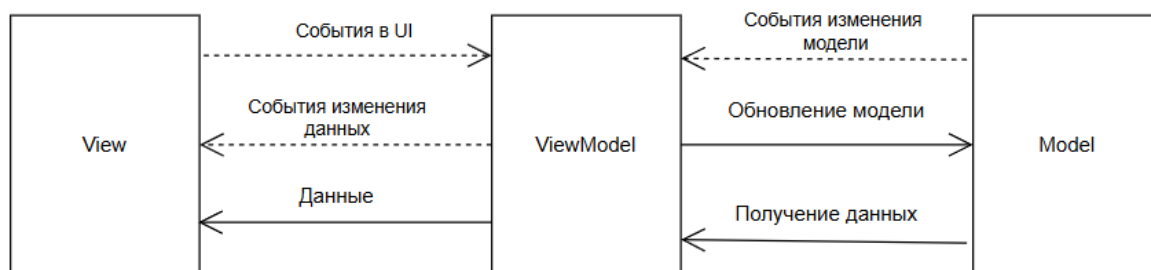


Рисунок 5 – Архитектура MVVM

1.3 Реализация приложения

1.3.1 Графический интерфейс

Для того, чтобы приложение смогло привлечь потенциальных пользователей, оно должно обладать приятным дизайном и понятным интерфейсом.

1.3.1.1 Layout

В Android Studio для разработки интерфейса используется редактор компоновки Layout Editor, который задействует XML файлы. Для реализации дизайна интерфейса страниц и всплывающих окон, используются стили, которые хранятся в папке с ресурсами – layout.

Рассмотрим назначение всех стилей:

- activity_asset_set.xml – содержит список токенов для совершения исходящей транзакции;
- activity_asset_get.xml – содержит список токенов для совершения входящей транзакции;
- activity_button_color.xml – содержит список цветов для выбора цвета кнопок;
- activity_connection_old_wallet.xml – содержит 24 поля для ввода seed для подключения уже существующего кошелька;
- activity_create_new_wallet_stage1.xml – содержит инструкцию по созданию нового кошелька;

- activity_create_or_connect_wallet.xml – содержит стартовое окно приложения с выбором подключения существующего кошелька или созданием нового;
- activity_create_pin_code.xml – содержит окно с созданием PIN-кода и подключением биометрии;
- activity_enter_pin_code.xml – содержит окно с вводом PIN-кода и биометрии;
- activity_get.xml – содержит qr-код для получения выбранного токена;
- activity_history_action.xml – содержит список всех входящих и исходящих транзакций и кнопку по ее очищению;
- activity_just_send.xml – содержит поле для ввода адреса кошелька получателя;
- activity_language_selection.xml – содержит список языков (системный, русский, английский, испанский, китайский, хинди) для применения нужного;
- activity_main_window.xml – содержит главное окно приложения, которое отображает: общую и индивидуальную сумму токенов в долларовом эквиваленте, список токенов с их актуальным количеством, адрес каждого токена, переход в активность с настройками, вводом адреса получателя, выбором токена для получения, сканером qr-кодов и списком транзакций.
- activity_seed_phrase.xml – содержит демонстрацию seed для ее сохранения;
- activity_seed_phrase2.xml – содержит 4 поля для проверки запоминания seed;
- activity_send_by_scanner.xml – содержит сканер qr-кодов;
- activity_settings.xml – содержит окно с настройками с выбором перехода к смене языка, цветы темы, текста, кнопок, скрытию баланса, смене PIN-кода и выходу из аккаунта;
- activity_text_color.xml – содержит список цветов для смены цвета текста;

– activity_themes_setting.xml – содержит список цветов для смены цвета темы приложения;

– activity_transaction_amount.xml – содержит поле для ввода количества токенов для исходящей транзакции и отображение имеющихся токенов на балансе;

– bottom_sheet_exit.xml – содержит предупреждение о безвозвратном выходе из аккаунта с возможностью подтвердить или отменить выход.

1.3.1.2 Anim

Была создана отдельная папка с анимированными стилями, которая хранит стиль pulse.xml – содержит анимацию при сохранении и проверки PIN-кода.

1.3.1.3 Font

Для осуществления подчеркивания некоторых слов или предложений используются несколько видов шрифтов, которые хранятся в каталоге font.

Рассмотрим список используемых шрифтов:

- almendra_display.xml;
- atomic_age.xml;
- delius_unicafe_bold.xml;
- lemon.xml.

1.3.1.4 Values

В папке Values хранятся цвета, используемые в приложении, и сертификаты шрифтов описанных в п. 1.3.1.3

Содержимое папки Values:

- colors.xml – содержит наименования цветов используемых в приложении;
- font-certs – содержит сертификаты используемых шрифтов в приложении;
- preloaded_fonts.xml – содержит наименования шрифтов используемых в приложении.

1.3.1.5 Strings

Для использования интернационального использования приложения была добавлена поддержка самых популярных языков.

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		27

Рассмотрим содержимое папки Strings:

- strings.xml – содержит поддержку русского языка;
- strings.xml (en) – содержит поддержку английского языка;
- strings.xml (es) – содержит поддержку испанского языка;
- strings.xml (hi) – содержит поддержку хинди (индийский язык);
- strings.xml (zh) – содержит поддержку китайского (упрощенного) языка.

1.3.2 Java/Kotlin-классы

В этом разделе будут рассмотрены Java/Kotlin-классы приложения, их функции и назначения.

Рассмотрим Java-классы, использующие Activity:

- AssetSelectionGet – это Activity, которая передает идентификатор токена для генерации qr-кода и адрес выбранного токена;
- AssetSelectionSet – это Activity, которая проверяет является ли валидным адрес, который ввел пользователь для осуществления исходящей транзакции;
- BalanceManager – это класс, который управляет балансами и адресами криптовалютных токенов, храня их в виде пар "ключ-значение";
- BiometricAuthManager – отвечает за биометрическую аутентификацию (сканер отпечатка пальца);
- BitcoinManager – это объект, который управляет Bitcoin-кошельком, обеспечивая генерацию адресов по стандарту BIP84, получение баланса и UTXO, а также создание и подписание транзакций с динамическим расчётом комиссии;
- BitcoinTransactionChecker – это класс, который проверяет входящие транзакции в блокчейн Bitcoin для указанного адреса, запрашивая API и анализируя выходы транзакций, найдя новую транзакцию, он сохраняет её в историю;
- ButtonColorActivity – это Activity, в которой пользователь может выбрать подходящий для него цвет кнопок;
- ButtonsColorManager – это класс, который: сохраняет выбранный цвет (по

идентификатору) в SharedPreferences и применяет сохранённый цвет ко всем переданным кнопкам. При отсутствии сохранённого значения используется цвет по умолчанию;

- ConfirmPinCode – это Activity, которая реализует экран подтверждения PIN-кода, где пользователь вводит 4 цифры для проверки ранее сохранённого кода и может использовать биометрическую аутентификацию;

- ConnectionOldWalletActivity – это Activity, которая позволяет восстановить существующий кошелёк, вводя 24-словную seed, после чего проверяет её контрольную сумму и генерирует адреса для Bitcoin, Ethereum, Solana и TON;

- CreateNewWalletStage1 – это Activity, в которой пользователь знакомится с правилами работы с seed и подтверждает сохранение сид-фразы с помощью трёх чекбоксов;

- CreateOrConnectWallet – это стартовая Activity, которая отображается, если пользователь впервые открывает приложение или осуществил выход из аккаунта. Пользователь может выбрать подключить существующий кошелек или сгенерировать новый;

- CreatePinCode – это Activity, реализует экран создания PIN-кода, где пользователь дважды вводит 4-значный код для подтверждения, после чего проверяется наличие интернет-соединения. При отсутствии связи отображается предупреждение с анимацией пульсации на кружках PIN-кода и выполняется периодическая проверка каждые 2 секунды, а после восстановления соединения код сохраняется в SharedPreferences и выполняется переход на главный экран;

- CrypthoHelper – этот класс, который предоставляет утилиты для симметричного шифрования AES-256 в режиме CBC с PKCS5Padding: генерирует случайный ключ, шифрует текст (добавляя случайный IV в начало зашифрованных данных) и расшифровывает обратно. IV генерируется случайно для каждого шифрования и сохраняется вместе с зашифрованным текстом, что обеспечивает корректное расшифрование без необходимости передавать IV

					ПГУ 1.090301.29.001 ПЗ	Лист
						29
Изм.	Лист	№ докум.	Подпись	Дата		

отдельно;

- DoubleBlackClickManager – этот класс реализует механизм "двойного нажатия назад для выхода": при первом нажатии показывается Toast-сообщение, а если повторное нажатие происходит в течение 2 секунд, активность завершается;

- EnterPinCode – это Activity, которая отвечает за аутентификацию с помощью ввода PIN-кода или биометрии, при успехе переходит в главное окно приложения;

- EthereumManager – этот объект управляет токенами в блокчейнт Ethereum, обеспечивая генерацию адресов по стандарту BIP44, получение балансов ETH и ERC20 через Web3j, а также отправку транзакций с динамическим расчётом комиссии и ожиданием подтверждения;

- EthereumTransactionChecker – это класс, который проверяет входящие транзакции ETH и ERC20-токенов для указанного адреса через Etherscan API, сохраняя обнаруженные транзакции в историю с помощью TransactionHistoryManager;

- GetActivity – это Activity, которая отображает экран для получения криптовалюты, где на основе выбранного токена (BTC, ETH, TON, USDT, SOL) извлекает соответствующий адрес из BalanceManager и генерирует QR-код с цветом фона, характерным для данного токена. Также генерируется предупреждение о осуществление перевода на этот адрес только для выбранного токена;

- HDWalletManager – этот объект управляет созданием и хранением HD-кошелька: генерирует мнемоническую фразу (12 или 24 слова), преобразует её в seed и создаёт мастер-ключ для BIP32 деривации. Он также обеспечивает безопасное сохранение seed в зашифрованном виде через AndroidKeyStore (AES + RSA) и его последующее извлечение [5];

- HistoryAction – этот класс отображает историю транзакций для всех поддерживаемых токенов, читая данные из файла через TransactionHistoryManager и динамически формируя UI-блоки с иконками, суммами, адресами и датами;

					ПГУ 1.090301.29.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

– JustSend – это Activity, которая предоставляет экран для ручного ввода адреса получателя криптовалюты. При нажатии кнопки он проверяет, что адрес не пустой и содержит не менее 10 символов;

– LanguageSelectoin – это Activity, в которой пользователь может выбрать интересующий его язык, поддерживаются: русский, английский, испанский, хинди и китайский языки;

– LocalHelper – этот класс позволяет динамически изменять язык приложения: он принимает код языка (например, "en", "ru") или значение "system" для использования системной локали, после чего устанавливает соответствующую локаль и обновляет конфигурацию ресурсов контекста;

– MainWindow – это главная Activity криптовалютного кошелька, отображающим балансы и текущие цены для BTC, ETH, TON, SOL и USDT с возможностью раскрытия адресов токенов. Он управляет периодическим обновлением балансов через RPC-запросы и цен через API, обрабатывает нажатия для отправки, получения, сканирования QR-кодов, просмотра истории и настроек, а также проверяет наличие интернет-соединения;

– PressEffectHelper - этот класс обеспечивает тактильный визуальный отклик при нажатии на различные UI-элементы (кнопки, ImageButton, кастомные GradientDrawable) путём временного изменения цвета фона на 100 миллисекунд;

– PriceManager – этот класс управляет получением и кэшированием балансов и цен криптовалют (BTC, ETH, TON, SOL, USDT) через запросы к локальному серверу, сохраняя данные в SharedPreferences для офлайн-доступа;

– QRCodeGenerator – этот класс генерирует QR-код для заданного адреса токена с указанным цветом фона и названием токена, создавая итоговое изображение, состоящее из QR-кода сверху и текстовой подписи снизу;

– SeedPhraseActivity – это Activity, которая отображает seed из 24 слов, разделяя её на две колонки по 12 слов в каждой, для того чтобы пользователь мог ее сохранить для дальнейшего использования кошелька на новом устройстве или при восстановлении доступа;

- SeedPhraseActivity2 – это Activity, которая обеспечивает проверку правильности сохранённой seed: он случайным образом выбирает 4 слова из 24, просит пользователя ввести их в соответствующие поля и сравнивает введённые значения с оригинальными. При успешной проверке выполняется переход к экрану создания PIN-кода (CreatePinCode), при ошибке – вибрация и уведомление;
- SendByScanner – этот класс обрабатывает результат сканирования QR-кода;
- SettingsActivty – это Activity, которая представляет экран настроек, позволяющий изменять язык, тему, цвет текста и кнопок, сбрасывать PIN-код, скрывать или отображать общий баланс (с подтверждением по PIN-коду) и выходить из аккаунта. При выходе он очищает все пользовательские данные и возвращает приложение к начальному экрану создания или подключения кошелька;
- SolanaManager – этот объект управляет токеном Solana: генерирует адрес из seed (через SHA-256 деривацию) и получает баланс;
- SolanaTransactionChecker – этот класс отслеживает входящие транзакции SOL для указанного адреса, сравнивая текущий баланс с сохранённым в SharedPreferences и сохраняя разницу в историю через TransactionHistoryManager;
- SystemBarColorHelper – этот класс управляет цветом системных панелей (статус-бара и навигационной панели), применяя сохранённый цвет темы и при необходимости затемняя его для светлых оттенков;
- TextColorActivity – это Activity, которая позволяет пользователю сменить цвет текста из 15 предложенных вариантов. Данный функционал необходим для кастомизация приложения, чтобы пользователь мог настроить интерфейс в зависимости от своих потребностей цветовой палитры;
- TextColorManager – этот класс управляет цветом текста в приложении: сохраняет выбранный цвет в SharedPreferences и применяет его ко всем указанным TextView;
- ThemeManager – этот класс управляет цветом фона темы приложения:

сохраняет выбранный пользователем цвет в SharedPreferences и применяет его к указанному корневому макету во всех окнах приложения за исключением стартовых при создании нового кошелька или импорте существующего;

– ThemesSetting – это Activity, которая позволяет пользователю сменить цвет темы (заднего фона) из 15 предложенных вариантов. Данный функционал необходим для кастомизация приложения, чтобы пользователь мог настроить интерфейс в зависимости от своих потребностей цветовой палитры;

– TransactionAmount – это Activity, которая отвечает за экран ввода суммы отправляемой криптовалюты после выбора адреса получателя, отображая доступный баланс токена. При подтверждении отправки он загружает мнемоническую фразу из защищённого хранилища, выполняет транзакцию через соответствующий менеджер (BitcoinManager, EthereumManager), сохраняет запись в историю и обновляет локальные кэши балансов;

– TransactionHistoryManager – Этот класс управляет историей транзакций, сохраняя записи о переводе или получении криптовалюты в текстовый файл во внешнем хранилище приложения. В нем учитывается время осуществления транзакции, при получении транзакции адрес отправителя, а при исходящей адрес получателя, тип токена и количество токенов;

– VibrationHelper – этот класс инициирует тактильную обратную связь (вибрацию) устройства на короткое время (100 мс) в ответ на действия пользователя, такие как ошибка ввода PIN-кода или некорректные данные.

Структура всех классов приложения представлена на рисунке 6.

приложения была реализована локальная база данных.

Таблица 1 – Атрибуты сущности WalletData

Поле	Тип данных	Описание
eth_balance	string	Количество токенов ETH
eth_price	float	Стоимость 1 токена ETH
ton_price	float	Стоимость 1 токена TON
ton_balance	string	Количество токенов TON
btc_balance	string	Количество токенов BTC
btc_price	float	Стоимость 1 токена BTC
sol_balance	string	Количество токенов SOL
sol_price	float	Стоимость 1 токена SOL
usdt_balance	string	Количество токенов USDT (ERC-20)

Таблица 2 – Атрибуты сущности MyPrefs

Поле	Тип данных	Описание
ton_address	string	Публичный ключ TON
bitcoin_address	string	Публичный ключ BTC
user_pin	string	PIN-код пользователя
encrypted_aes_key	string	Ключ шифрования
solana_address	string	Публичный ключ SOL
ethereum_address	string	Публичный ключ ETH
encrypted_seed	string	Зашифрованная seed

Таблица 3 – Атрибуты сущности UserElement

Поле	Тип данных	Описание
color_theme	string	Цвет темы приложения
color_text	string	Цвет текста приложения
color_button	string	Цвет кнопок приложения
language	string	Язык приложения

Таблица 4 – Атрибуты сущности bts_txs

Поле	Тип данных	Описание
BTC	boolean[]	Информация о транзакции (адрес контрагента, дата и время)

Таблица 5 – Атрибуты сущности eth_txs

Поле	Тип данных	Описание
ETH	boolean[]	Информация о транзакции (адрес контрагента, дата и время)

Таблица 6 – Атрибуты сущности usdy_txs

Поле	Тип данных	Описание
USDT	boolean[]	Информация о транзакции (адрес контрагента, дата и время)

Диаграмма последовательности действий представлена на рисунке 8.

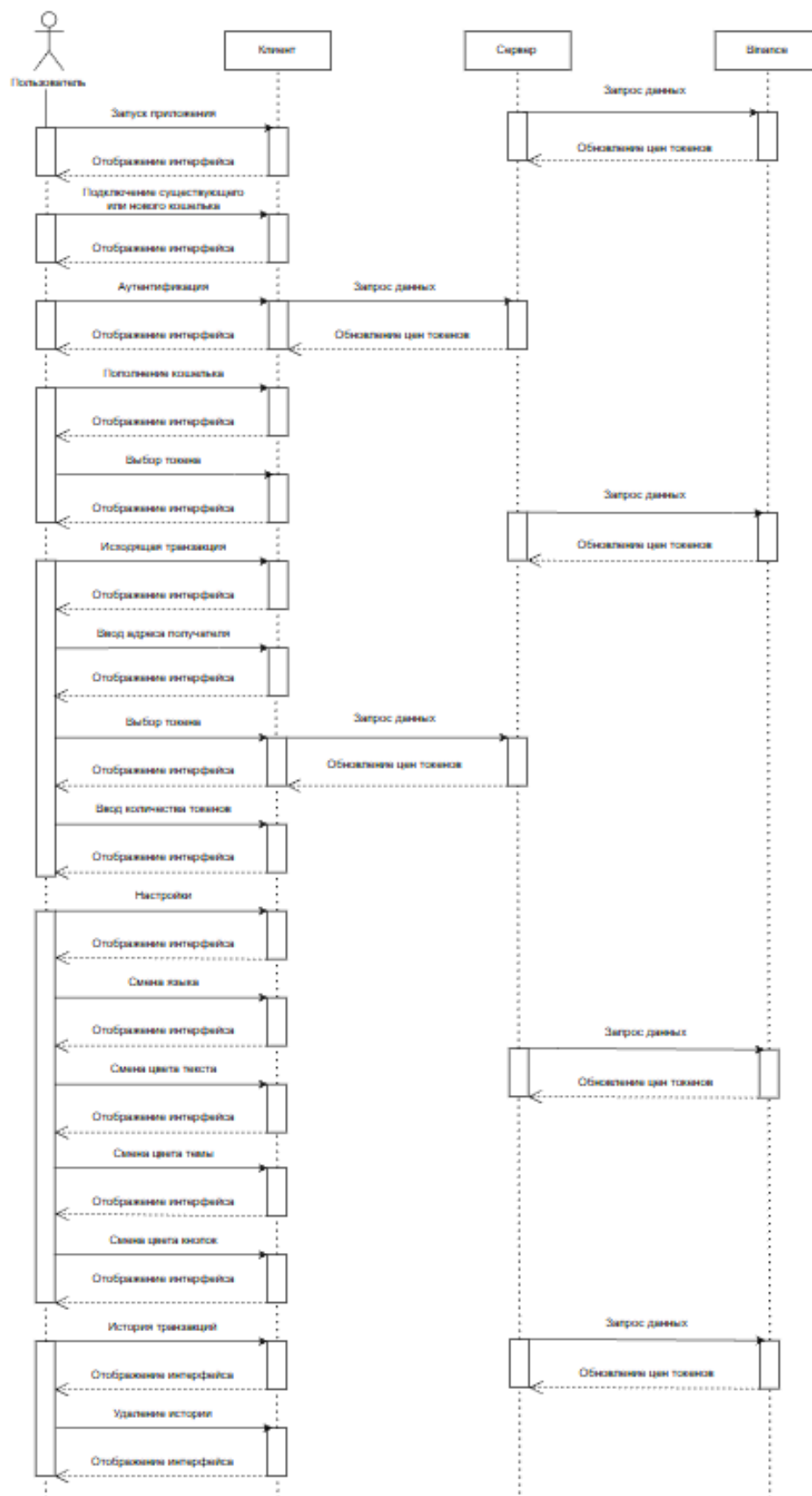


Рисунок 8 – UML-диаграмма последовательности

1.4 Тестирование

Чтобы запустить приложение нужно нажать на его иконку (рисунок 9).



Рисунок 9 – Иконка приложения

Для демонстрации первичного запуска необходимо выйти из текущего аккаунта, чтобы пройти путь с самого начала (рисунок 10).

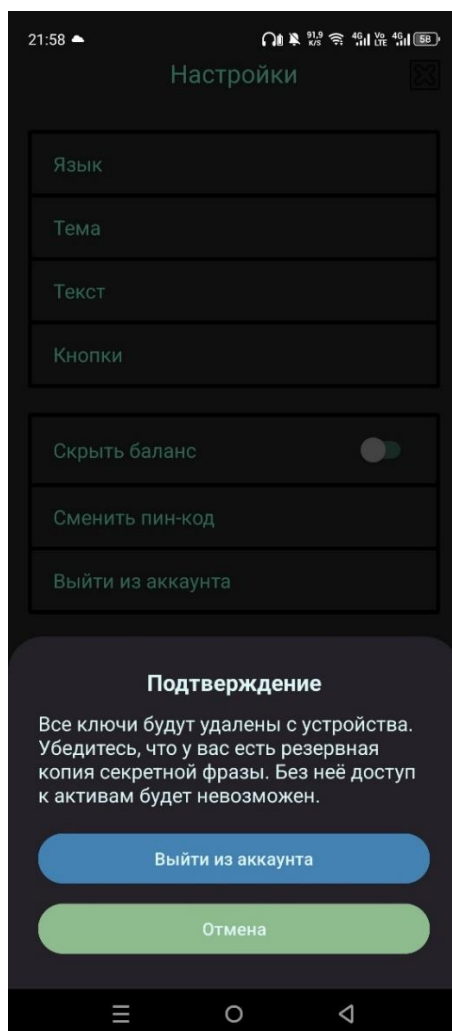


Рисунок 10 – Выход из текущего аккаунта

					ПГУ 1.090301.29.001 ПЗ	Лист
						38
Изм.	Лист	№ докум.	Подпись	Дата		

При первичном запуске приложения клиенту предлагается подключить существующий кошелек или создать новый (рисунок 11).

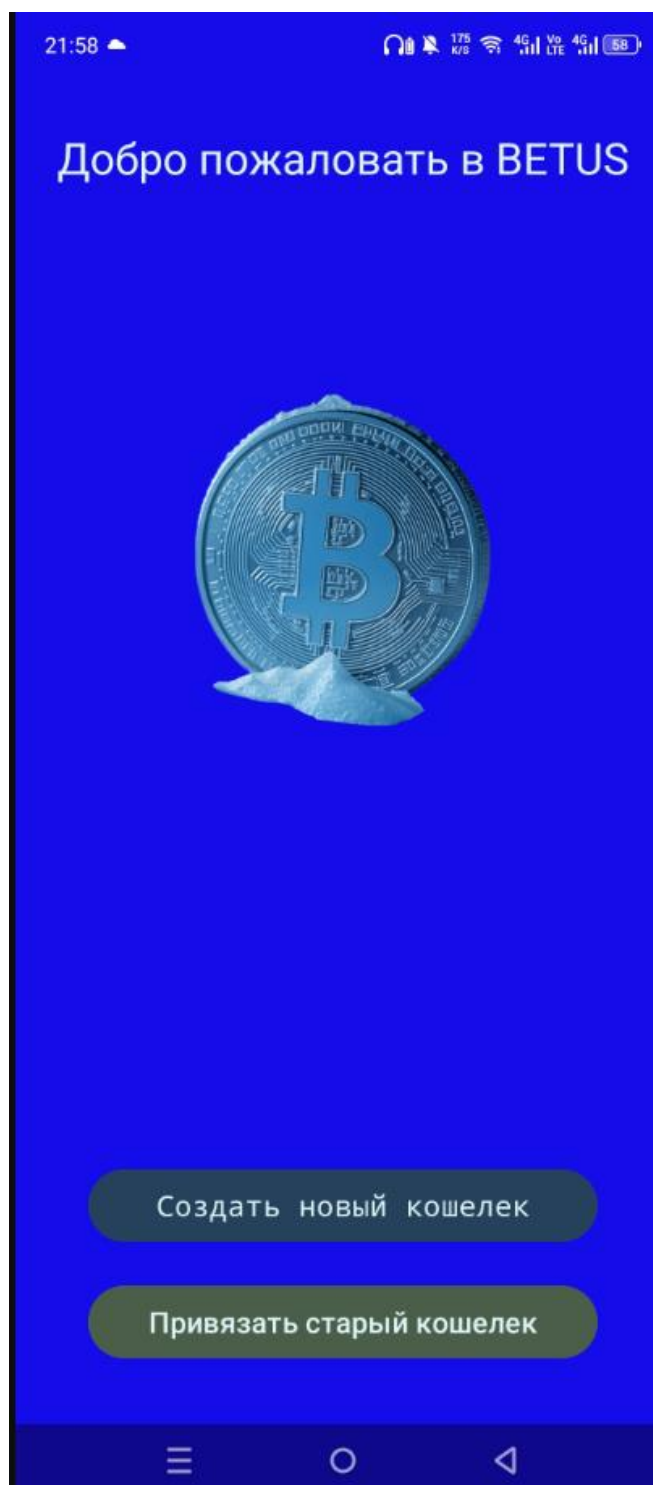


Рисунок 11 – Стартовое окно

При создании нового кошелька пользователь должен ознакомиться с правилами работы с seed, для этого он должен заполнить 3 чек-бокса (рисунок 12).

					ПГУ 1.090301.29.001 ПЗ	Лист
						39
Изм.	Лист	№ докум.	Подпись	Дата		

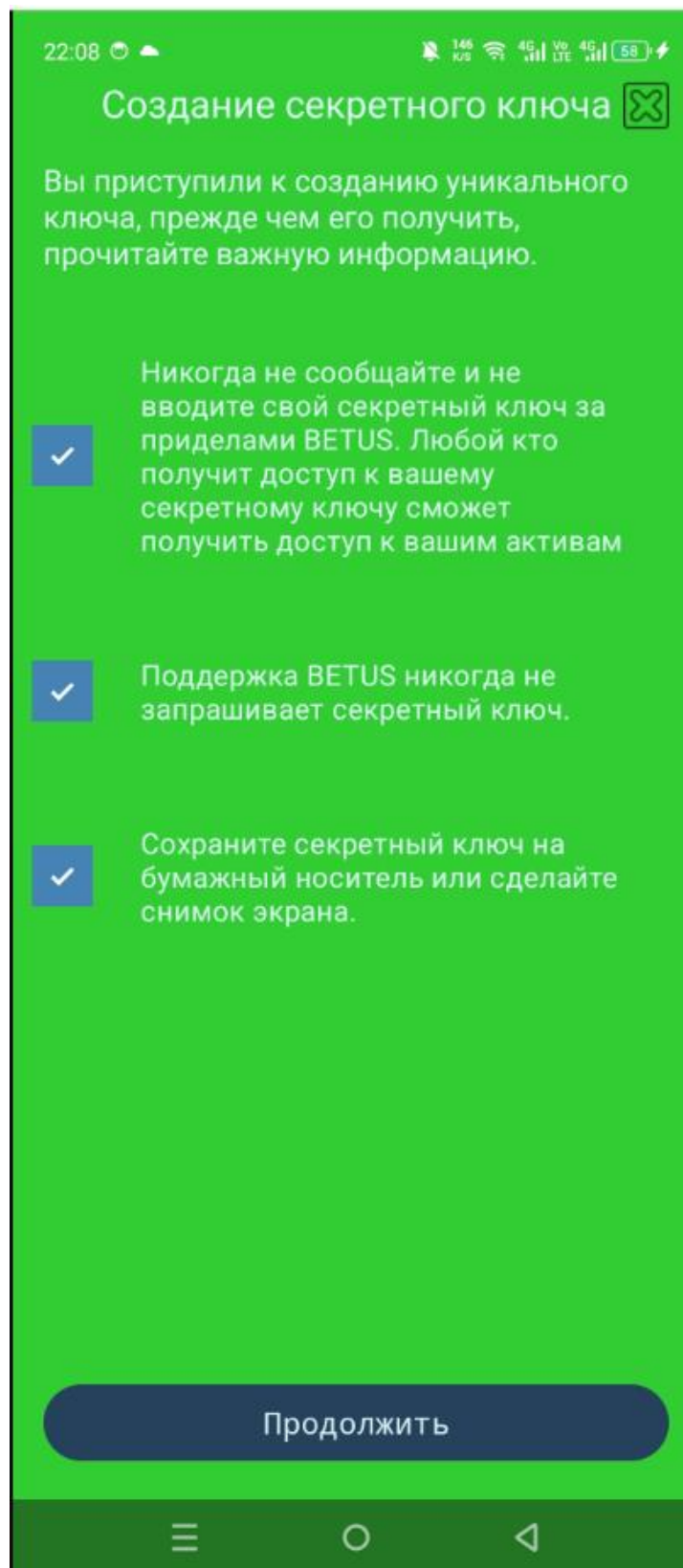


Рисунок 12 – Знакомство с правилами работы с seed

Далее пользователю необходимо сохранить сгенерированную seed, чтобы в будущем у него была возможность восстановить данные (рисунок 13).

					ПГУ 1.090301.29.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		



Рисунок 13 – Сгенерированная seed

Чтобы обезопасить пользователя от необдуманных поступков необходимо проверить сохранил ли он свою seed, для этого он должен ввести 4 случайных слова из своей seed (рисунок 14).

					ПГУ 1.090301.29.001 ПЗ	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

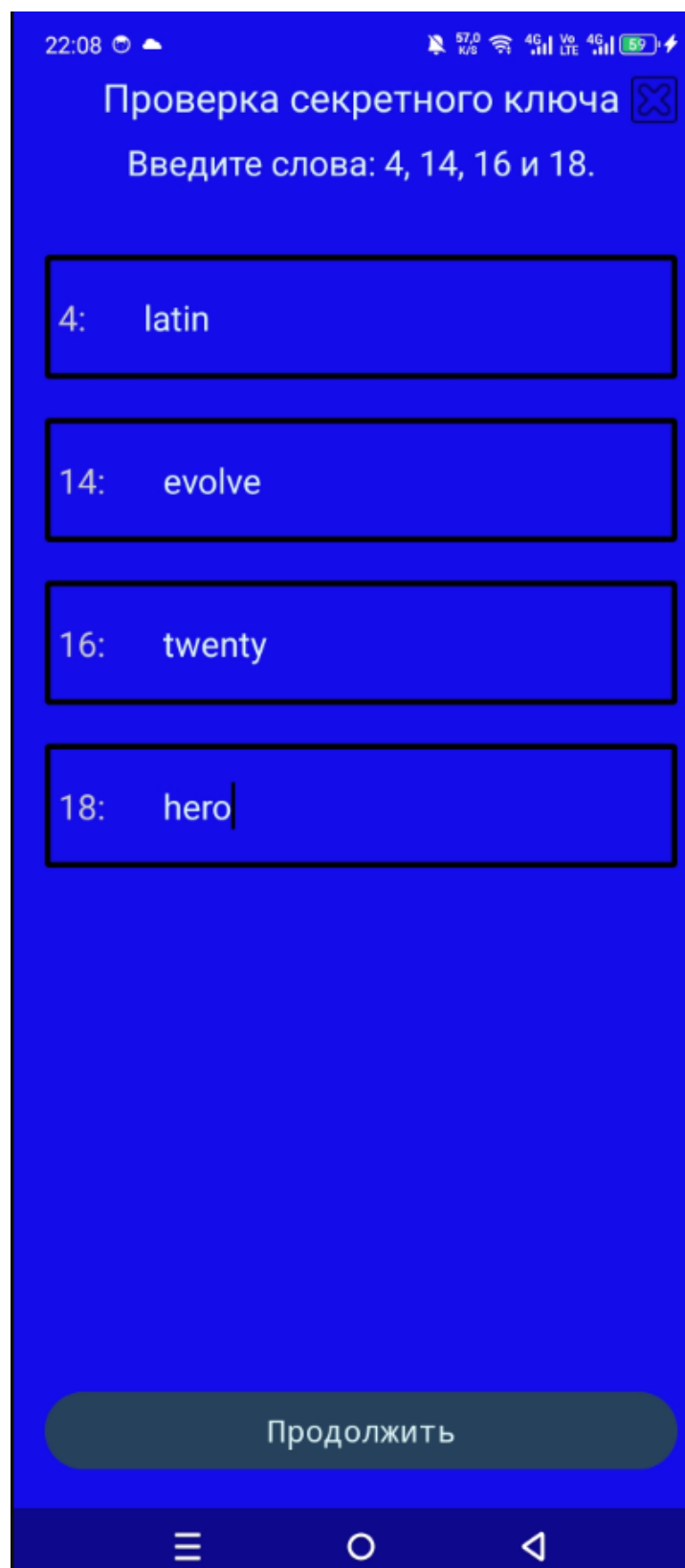


Рисунок 14 – Ввод 4 случайных слов из seed

После успешного ввода 4 слов seed пользователю предлагается придумать PIN-код из 4 цифр (рисунок 15).

					ПГУ 1.090301.29.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		



Рисунок 15 – Создание PIN-кода

Далее нужно подтвердить созданный PIN-код (рисунок 16).

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		43



Рисунок 16 –Подтверждение PIN-кода

После успешного подтверждения PIN-кода открывается главное окно приложения (рисунок 17).

					ПГУ 1.090301.29.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

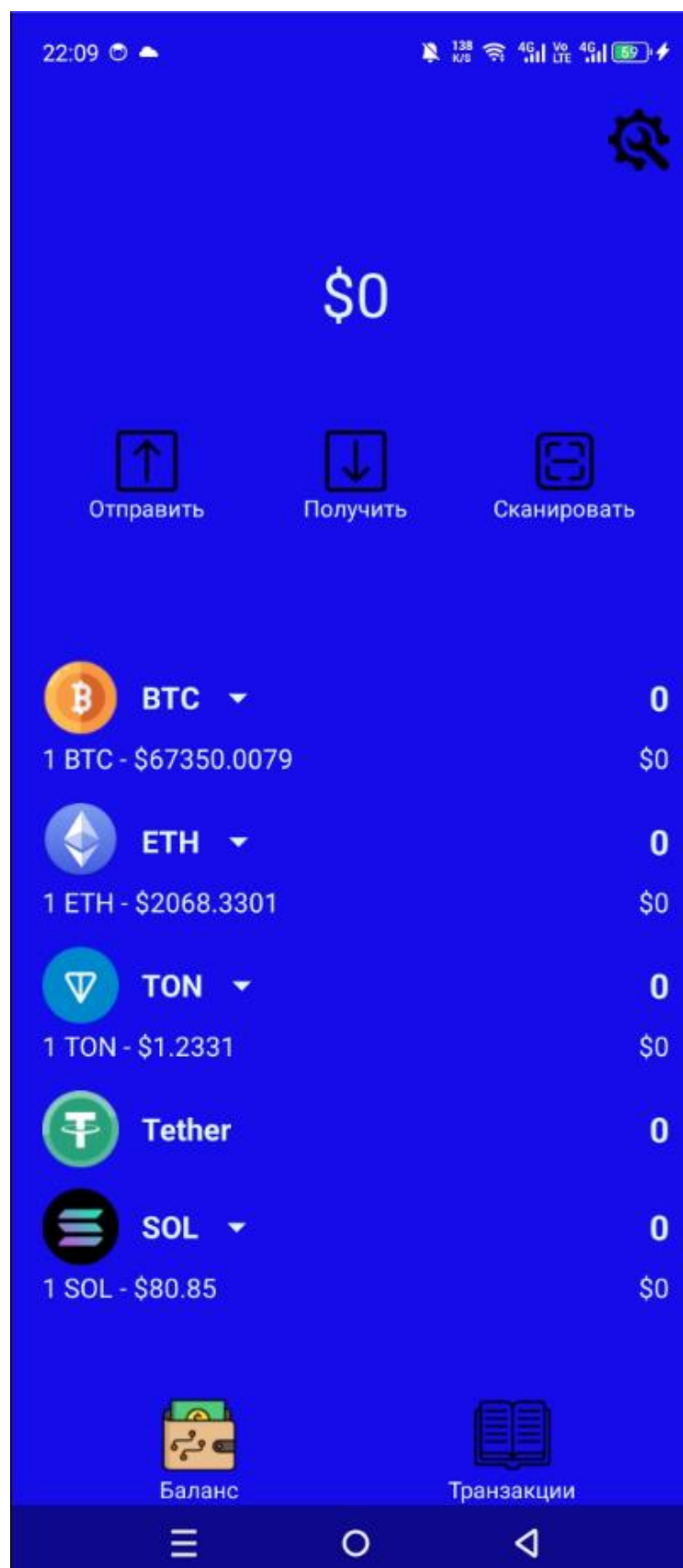


Рисунок 17 – Главное окно приложения

Перейдем в настройки, чтобы кастомизировать интерфейс (рисунок 18).

					ПГУ 1.090301.29.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

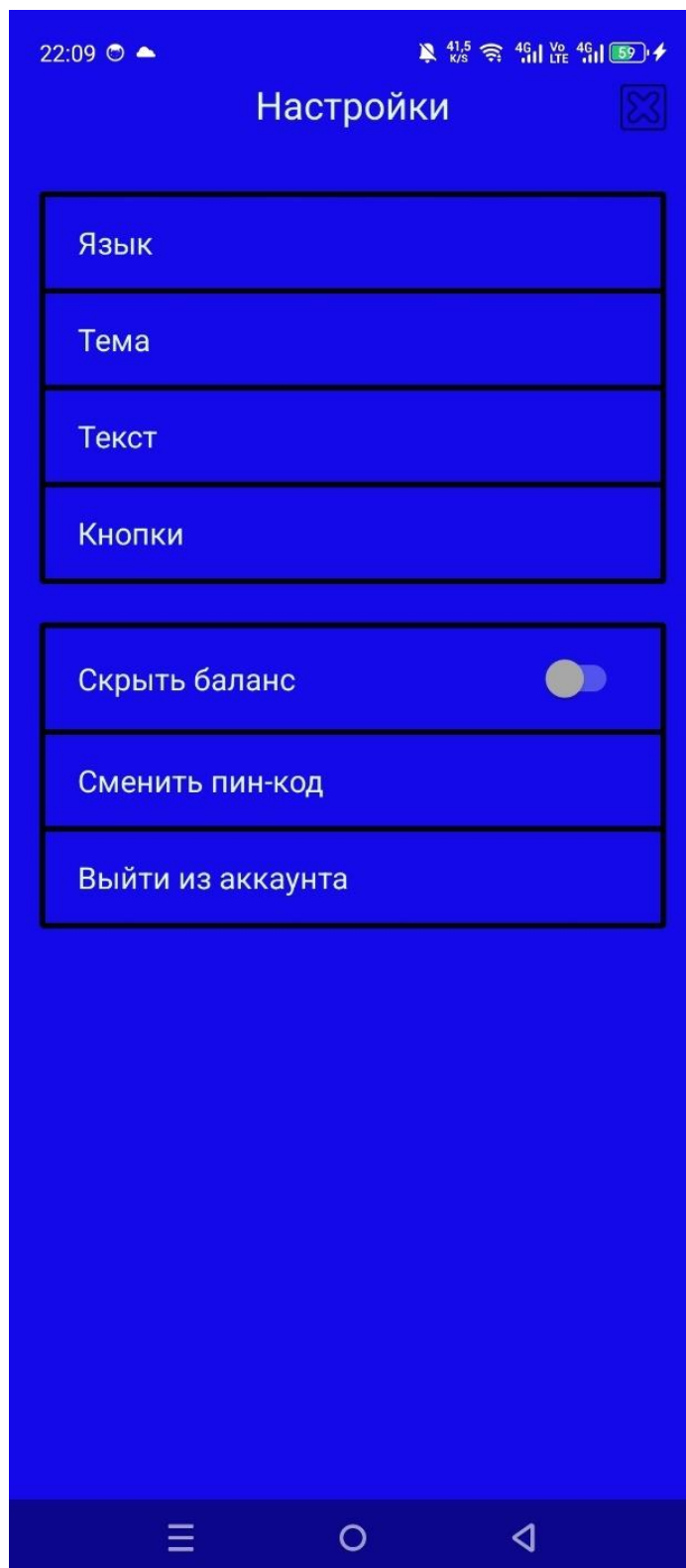


Рисунок 18 – Раздел «Настройки»

По умолчанию используются язык, который установлен на устройстве пользователя, но при желании можно выбрать 1 из 5 языков, которые поддерживаются, применим английский язык (рисунок 19).

					ПГУ 1.090301.29.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

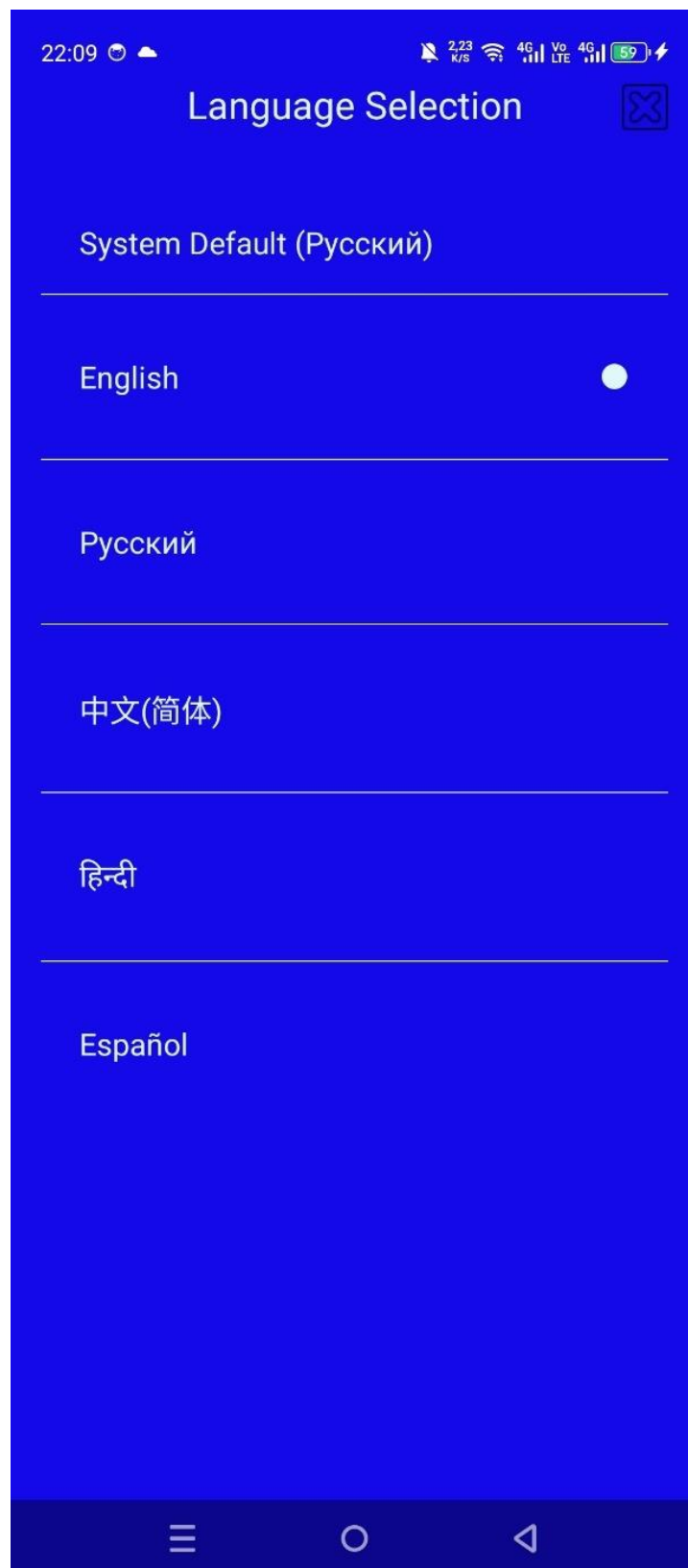


Рисунок 19 – Выбор английского языка

Чтобы увидеть изменения можно посетить основное окно приложения (рисунок 20).

					ПГУ 1.090301.29.001 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

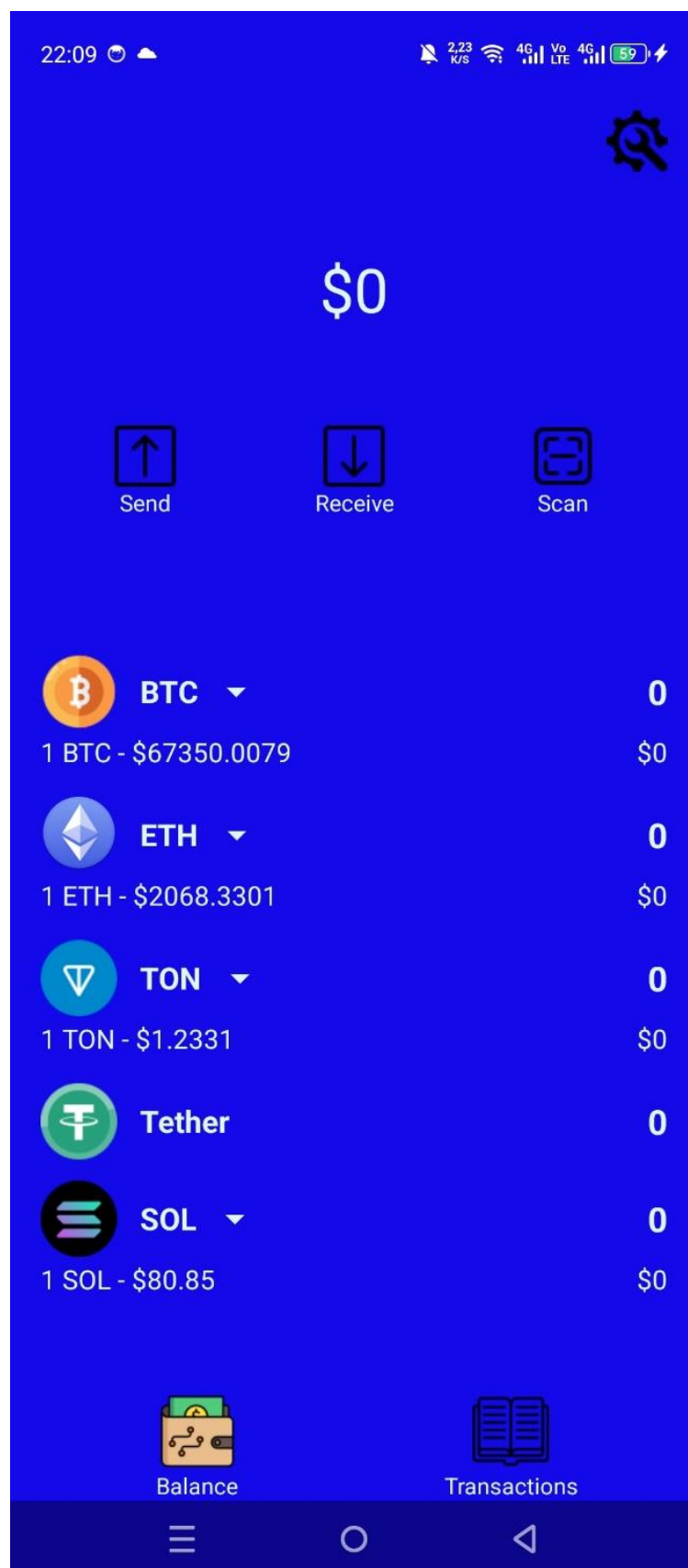


Рисунок 20 – Основное окно приложения с измененным языком

Далее изменим цвет темы приложения (рисунок 21).

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		48

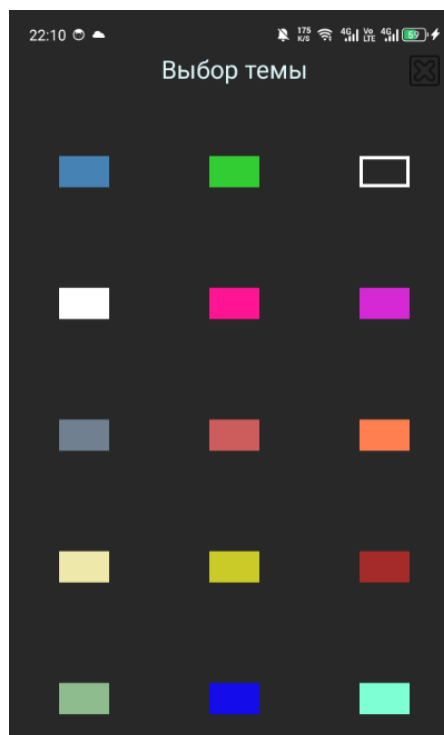


Рисунок 21 – Применение нового цвета темы

Изменим цвет текста приложения (рисунок 22).

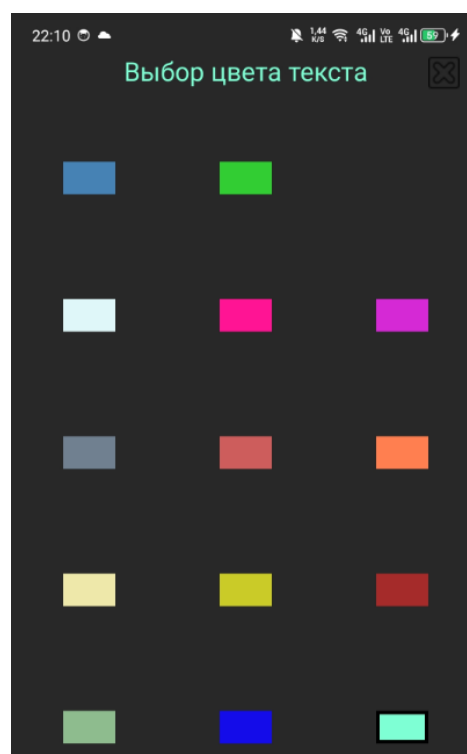


Рисунок 22 – Измененный цвет текста приложения

Изменим цвет кнопок (рисунок 23).

					ПГУ 1.090301.29.001 ПЗ	Лист
						49
Изм.	Лист	№ докум.	Подпись	Дата		

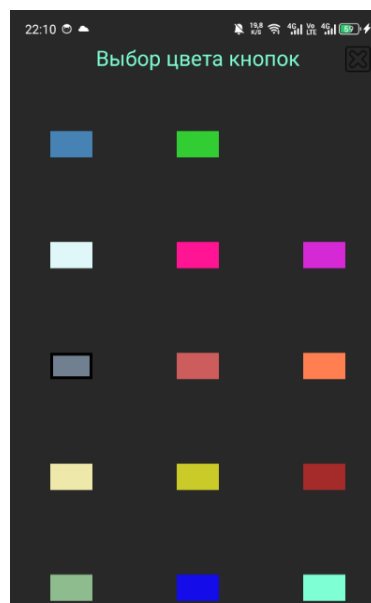


Рисунок 23 – Изменение цвета кнопок

Результат смены цвета кнопок можно увидеть в окне ввода адреса получателя при исходящей транзакции, введем адрес, на который будет осуществлен перевод (рисунок 24).

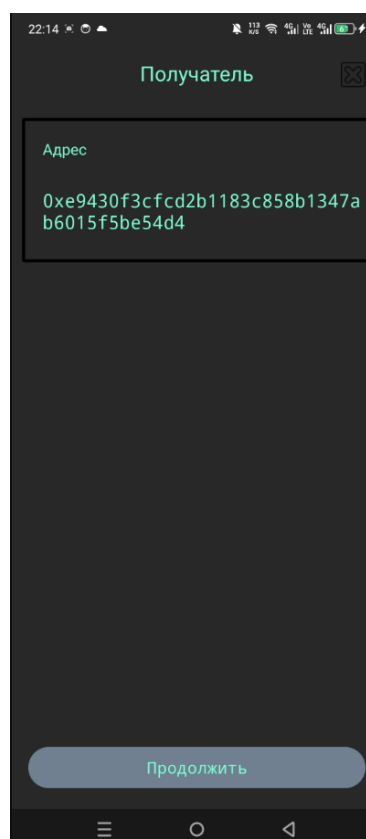


Рисунок 24 – Ввод адреса получателя

					ПГУ 1.090301.29.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение автоматически определило, что адрес получателя относится к ETH и предложило перейти на следующий шаг (рисунок 25).

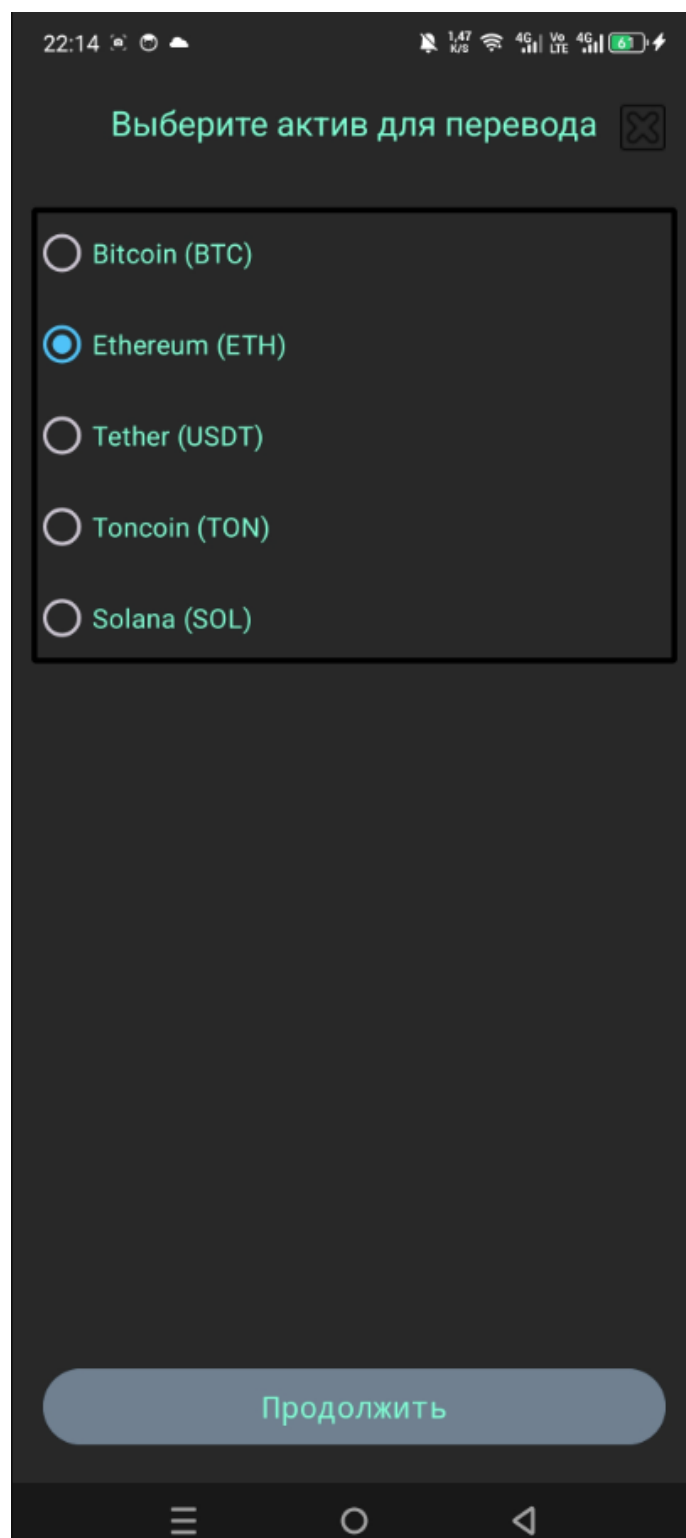


Рисунок 25 – Автоматическое определение сети

Далее необходимо ввести количество токенов для перевода (рисунок 26).

					ПГУ 1.090301.29.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

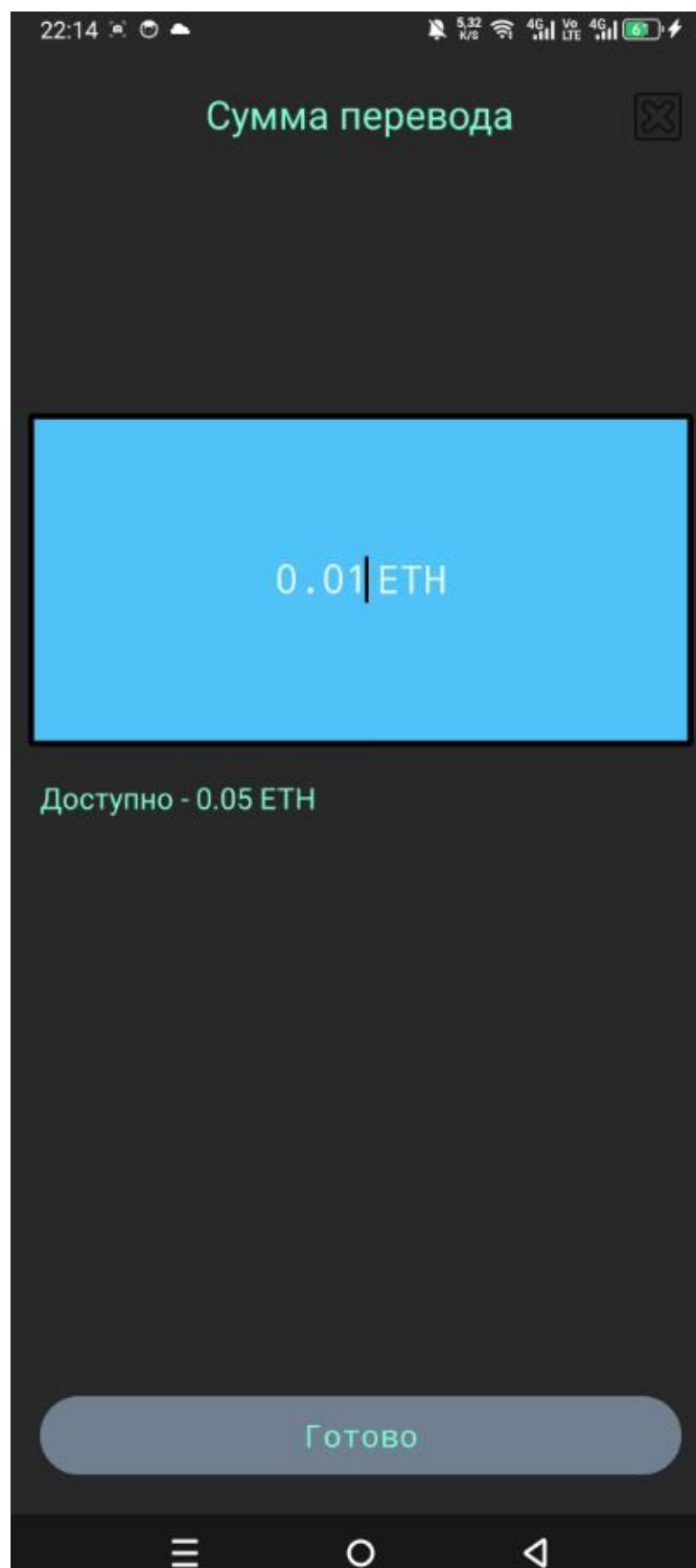


Рисунок 26 – Ввод количества токенов для исходящей транзакции

Приложение предлагает скопировать хэш транзакции, чтобы проверить в обозревателе осуществление транзакции (рисунок 27).

					ПГУ 1.090301.29.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

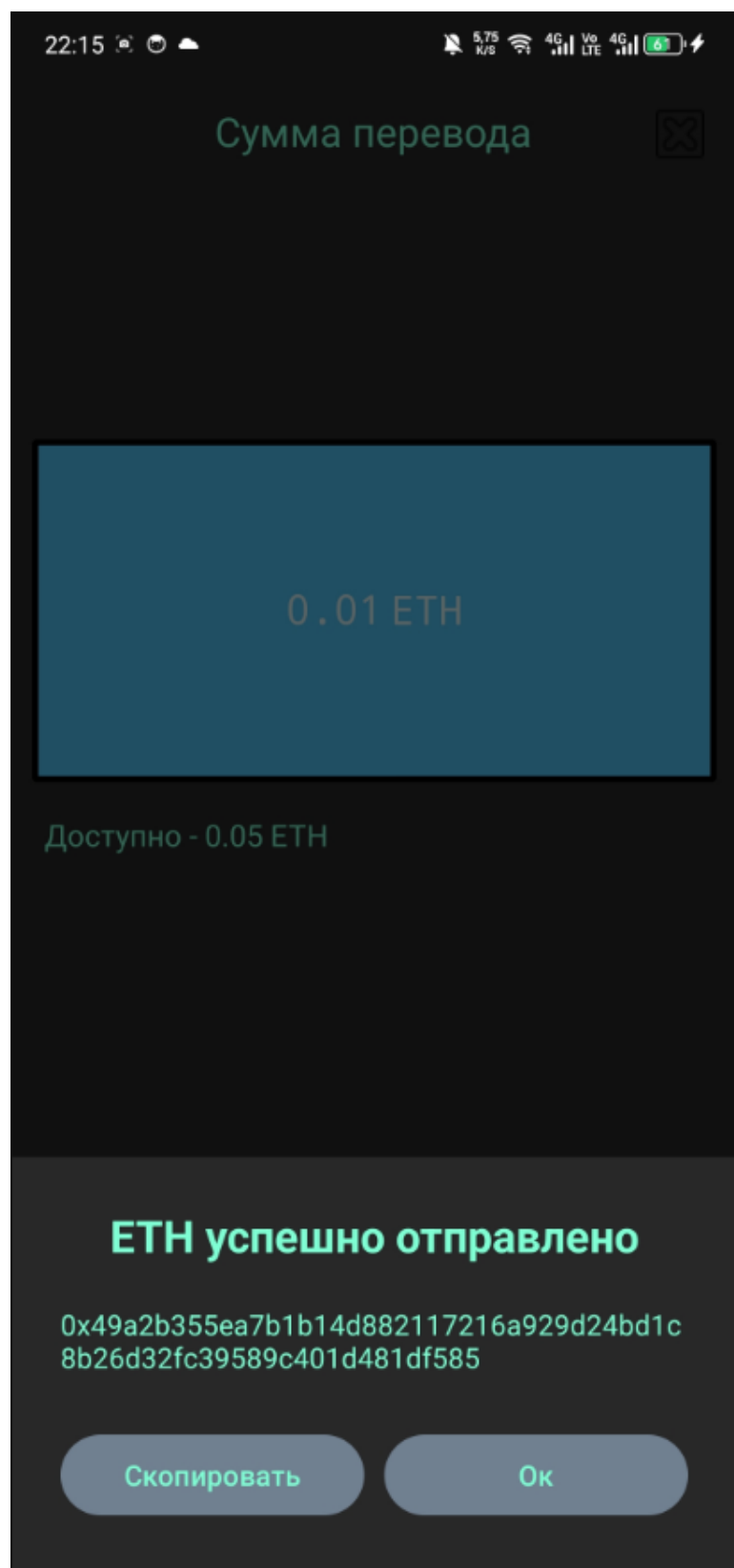


Рисунок 27 – Хэш транзакции

Проверим данную транзакцию в обозревателе по полученному хэшу (рисунок 28).

					ПГУ 1.090301.29.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

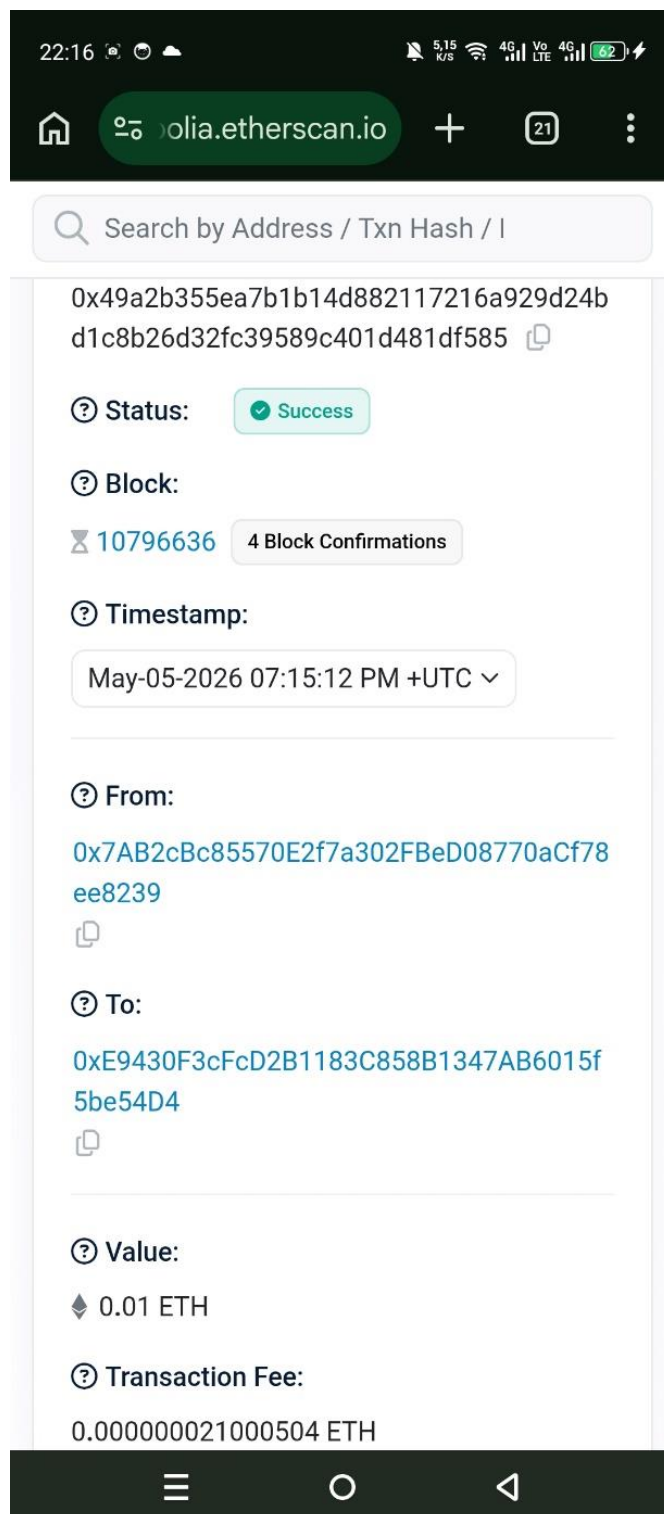


Рисунок 28 – Подтверждение успешной исходящей транзакции

Для того чтобы было удобно пополнять свой кошелек, была добавлена генерация QR-кода и копирование адреса данной сети. В главном меню (рисунок 21) необходимо нажать на «Пополнить», затем выбрать нужный токен (рисунок 25) и откроется окно с QR-кодом и адресом для перевода (рисунок 29).

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

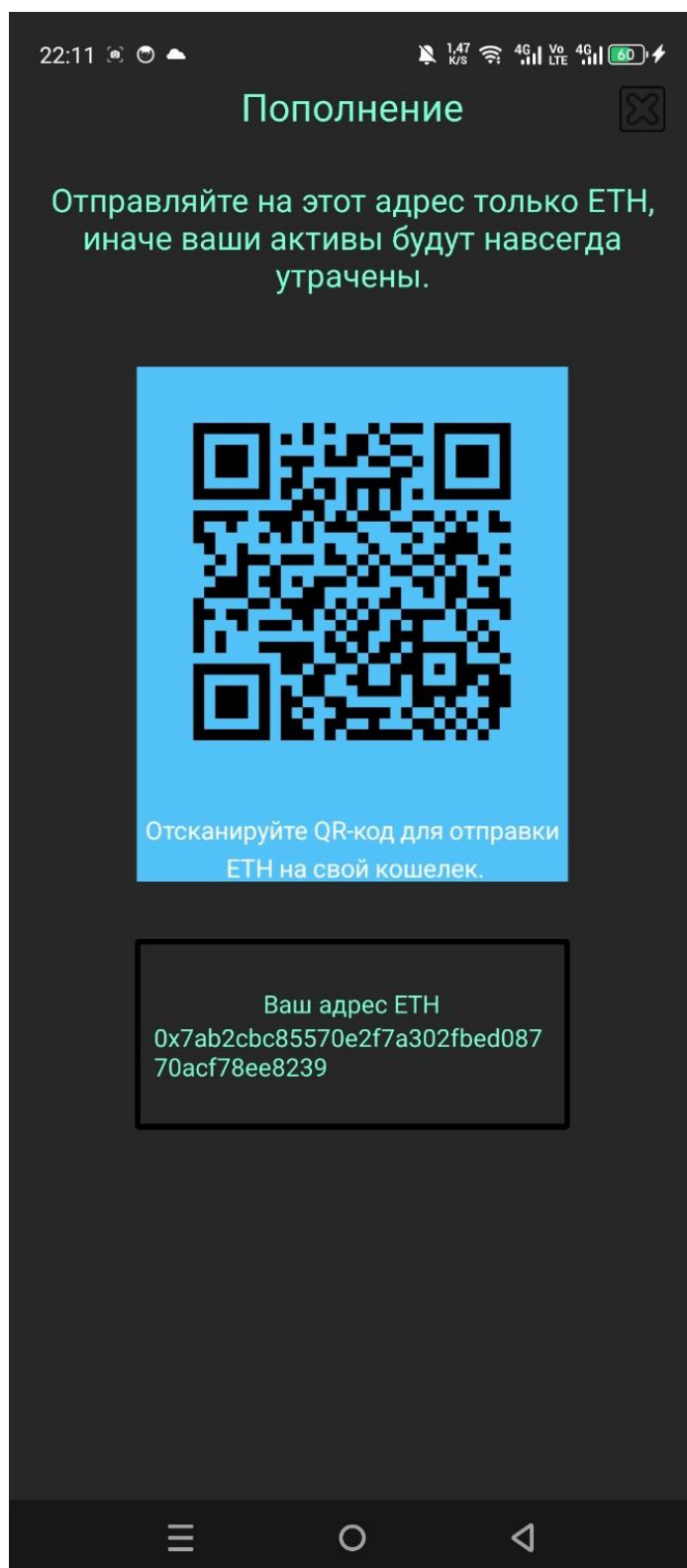


Рисунок 29 – QR-код с адресом для входящей транзакции

Чтобы не заходить в сторонние обозреватели есть отдельный раздел, который называется «Транзакции» в нем хранятся вся история транзакций (рисунок 30).

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		55

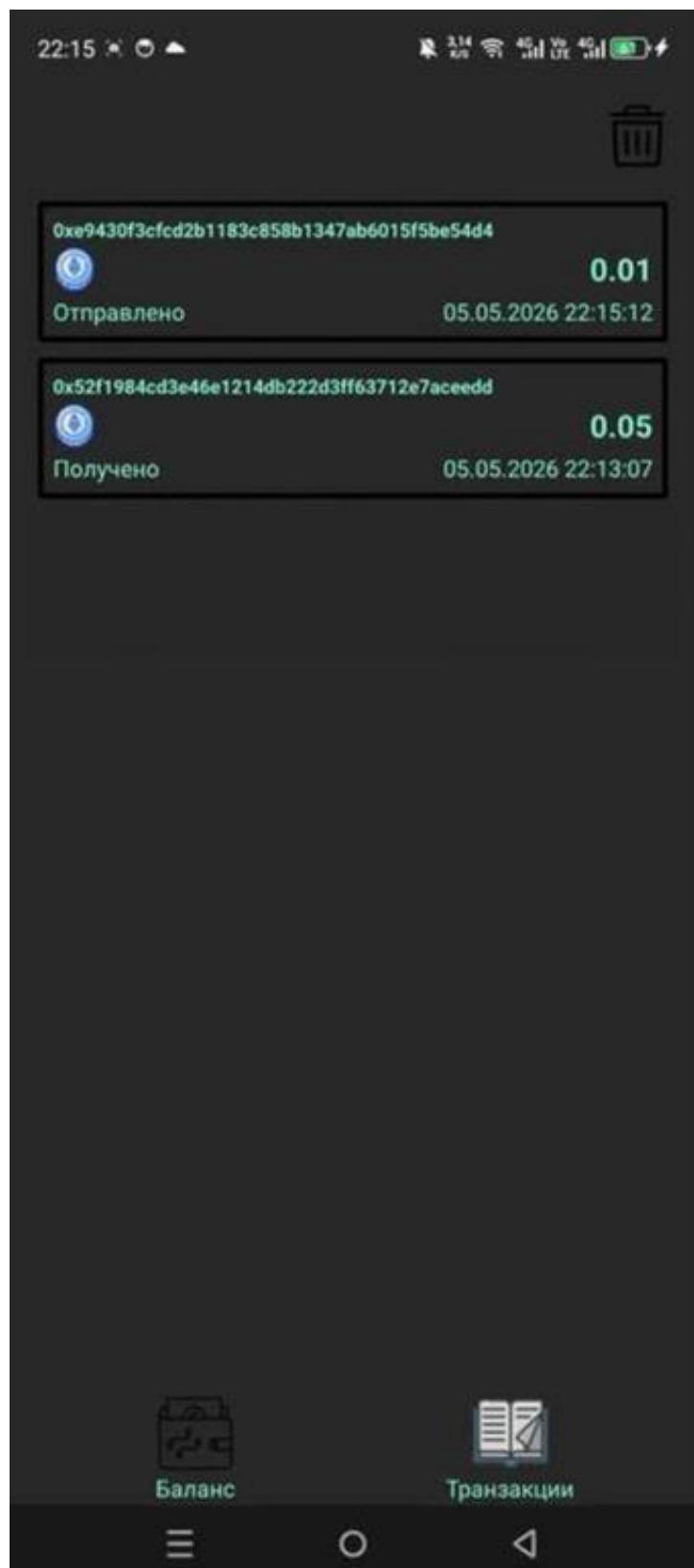


Рисунок 30 – История транзакций

По желанию можно удалить историю транзакций нажав на корзину в правом верхнем углу (рисунок 30) и подтвердить данную операцию (рисунок 31).

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		56

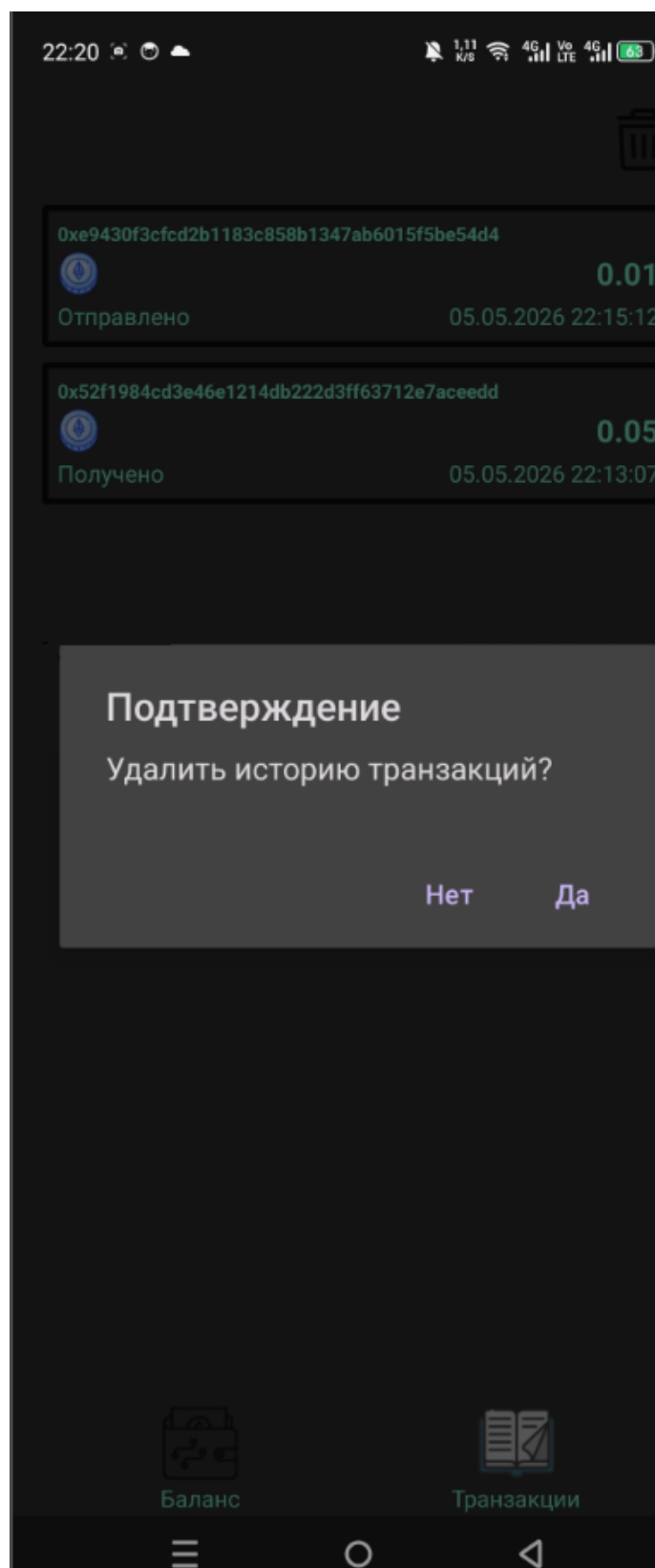


Рисунок 31 – Подтверждение удаления истории транзакций

Результат удаления истории транзакций можно увидеть после нажатия на кнопку «Да» (рисунок 32).

					ПГУ 1.090301.29.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

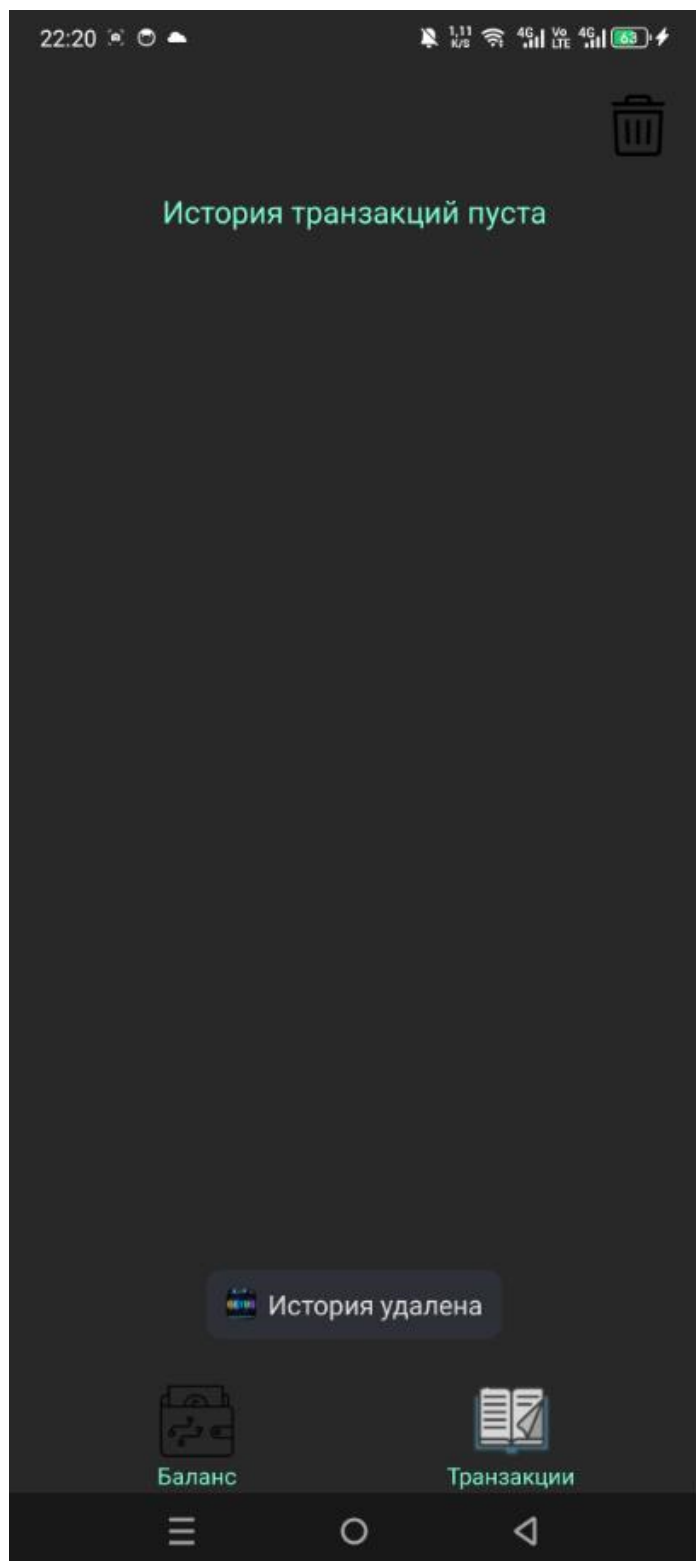


Рисунок 32 – Очищенная история транзакций

После осуществления входящей и исходящей транзакции на главном экране отображается актуальное количество токенов и общий баланс в долларах США (рисунок 33).

					ПГУ 1.090301.29.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

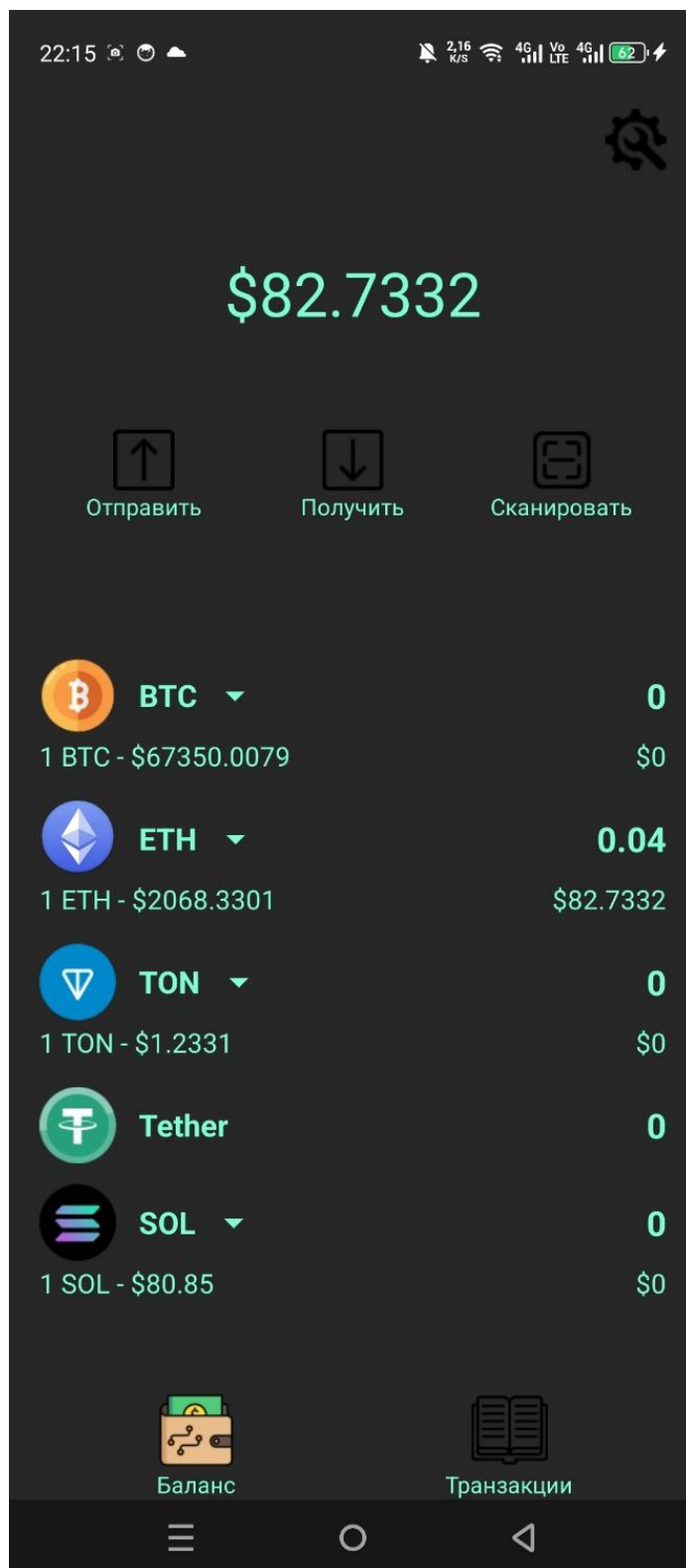


Рисунок 33 – Главный экран приложения

Для удобства осуществления исходящих транзакций был добавлен сканер QR-кодов, чтобы его открыть нужно нажать на «Сканировать» (рисунок 33), после сканирование QR-кода автоматически подставится адрес получателя (рисунок 34).

					ПГУ 1.090301.29.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

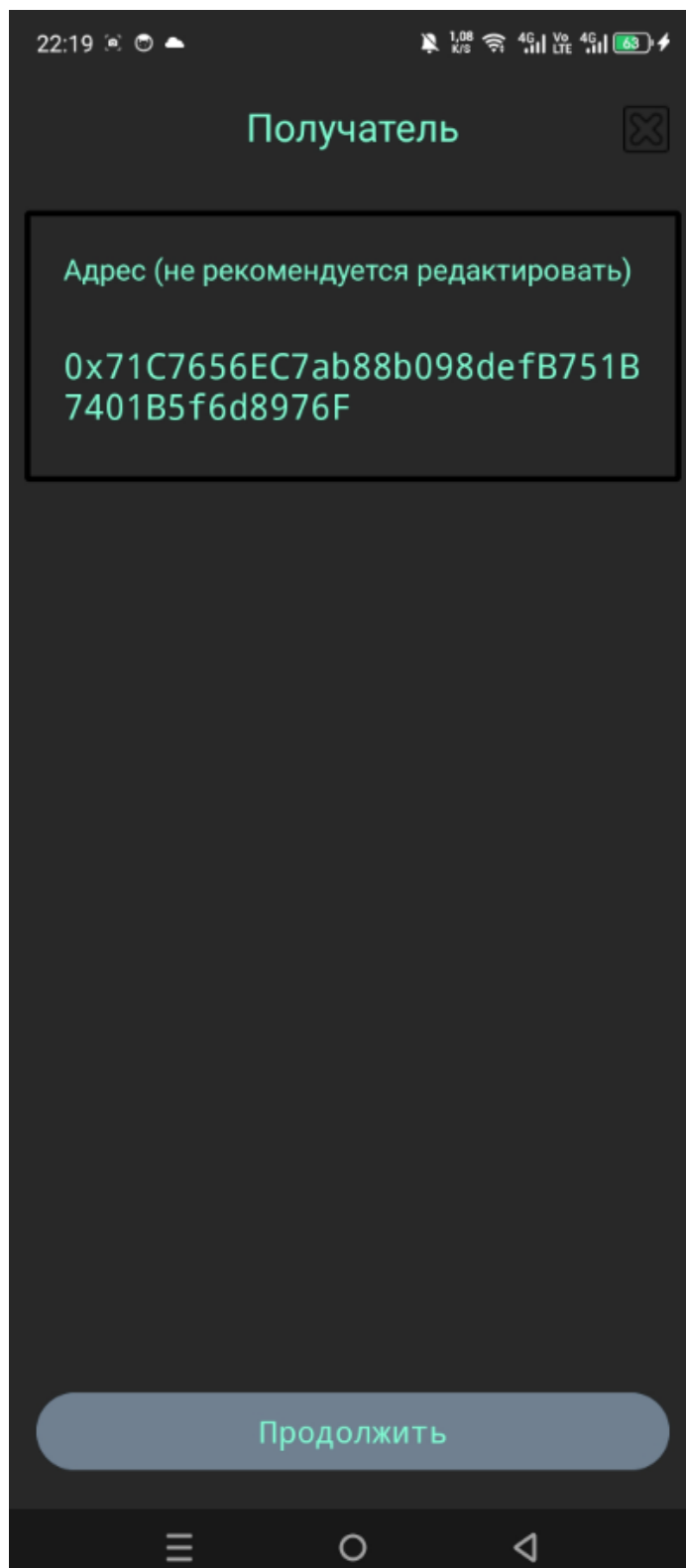


Рисунок 34 – Автоматическая установка адреса получателя

При этом происходят проверки на введенный адрес, например, отсканированный ранее адрес является недействительным, т.е. с ним ранее не происходили операции в блокчейне (рисунок 35).

					ПГУ 1.090301.29.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

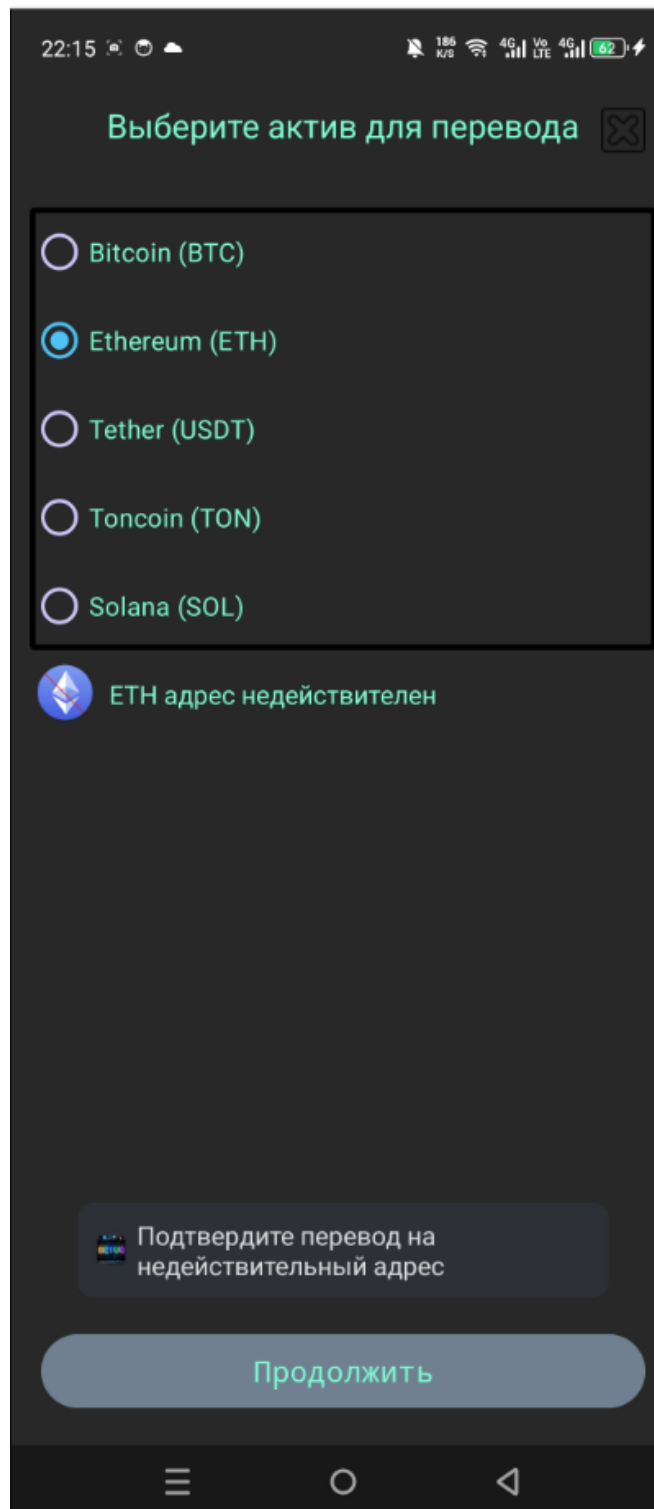


Рисунок 35 – Проверка введенного адреса

Осуществим вход в аккаунт, из которого был выполнен выход в самом начале (рисунок 10). Нажмем на «Привязать существующий кошелек» (рисунок 11). После этого откроется окно для ввода seed из 24 слов для восстановления кошелька (рисунок 36).

22:21 1.19 K/S 4G LTE 4G LTE 68%

✕

Введите секретный ключ

Для получения доступа к своим активам, введите 24 секретных слова, которые вы получили при создании кошелька.

1:

2:

3:

4:

5:

6:

7:

8:

☰ ○ ◀

Рисунок 36 – Ввод seed

После ввода 24 начинается проверка на корректность ввода, если хотя бы 1 слово было введено с ошибкой, то восстановить доступ не получится (рисунок 37).

					ПГУ 1.090301.29.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		



Рисунок 37 – Некорректный ввод seed

После успешного ввода seed пользователю предлагается придумать PIN-кода (рисунок 15) и подтвердить его (рисунок 16). После этого откроется окно с кошельком, который использовался ранее (рисунок 38).

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		63

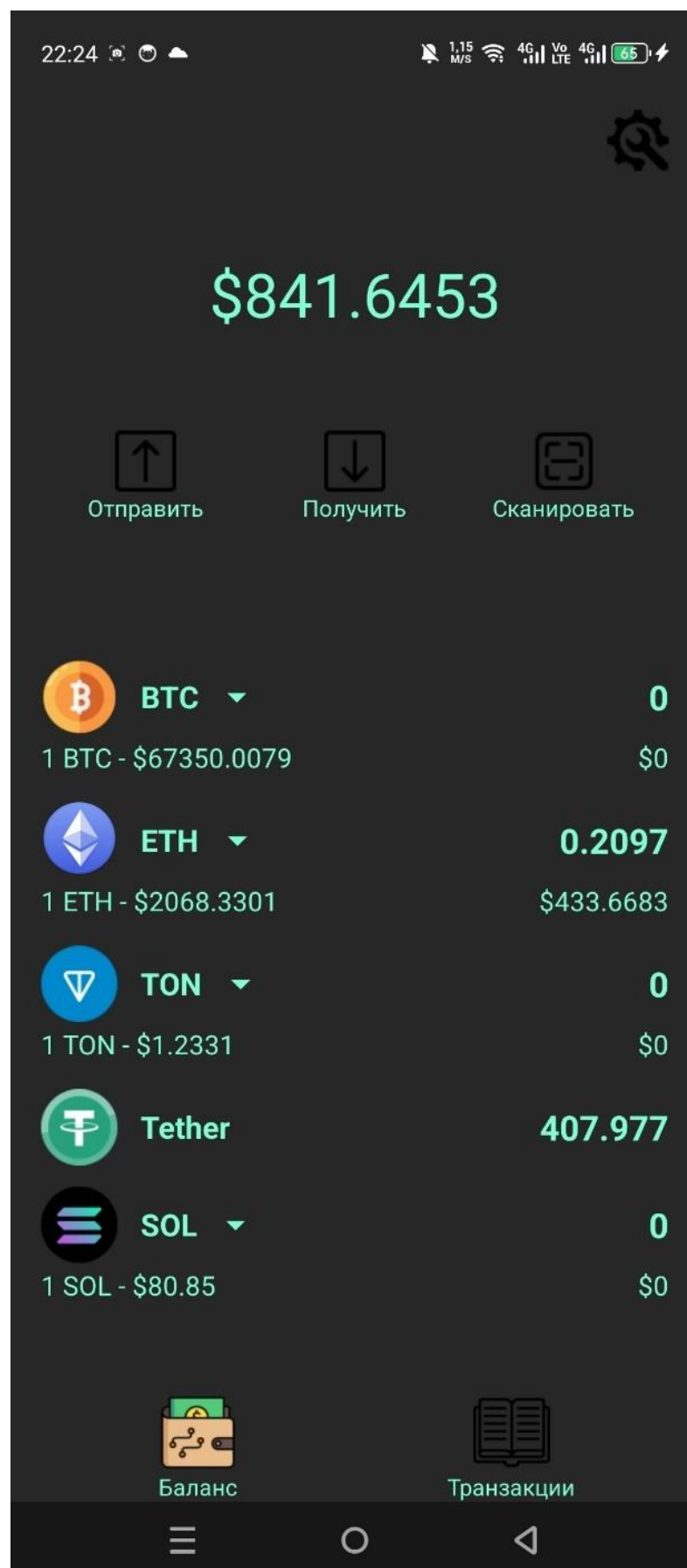


Рисунок 38 – Главное окно приложения

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		64

2 Безопасность жизнедеятельности

В наше время работа за компьютером стала неотъемлемой частью жизни многих людей, что приводит к появлению различных проблем со здоровьем, таких как напряжение глаз, неправильная осанка, туннельный синдром и других заболеваний. В особенности это касается людей чья профессиональная деятельность связана с долгим нахождением перед экраном монитора. Для того, чтобы обезопасить себя от проблем продолжительной работы за компьютером, необходимо соблюдать ряд правил. В этой главе мы рассмотрим необходимые меры, при соблюдении которых, можно предотвратить все возможные проблемы, связанные с долгой работой за компьютером.

2.1 Требования к организации рабочего места

При организации рабочего места необходимо соблюдать ряд рекомендаций, которые помогут минимизировать риски для здоровья. Рассмотрим список предлагаемых рекомендаций:

- эргономичное кресло. Для комфортной работы за компьютером важно приобрести хорошее и качественное кресло, которое обладает возможностью регулировки высоты и угла наклона, и необходимой формой для спины и комплекции человека;

- правильная высота стола. Рабочий стол должен быть на такой высоте, чтобы руки находились под углом 90 градусов при работе с клавиатурой и мышью;

- положение монитора. Чтобы предотвратить проблемы со зрением, монитор должен находиться на комфортном расстоянии от глаз. Рекомендуемым значением является 50-70 см от монитора. Положение монитора должно быть такое чтобы, верхняя часть экрана была на уровне глаз.

Пример соблюдения всех правил можно увидеть на рисунке 39.

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		65



Рисунок 39 – Правильное положение при работе за компьютером

2.2 Требования к освещению

Одна из главных проблем, которая возникает при долгой работе за компьютером – это вред для глаз. Неправильное освещение помещения и настроек яркости монитора, может привести к усталости глаз, сухости, покраснению, размытости зрения и головным болям. Чтобы предотвратить это, необходимо соблюдать некоторые требования к освещению в помещении:

- естественное освещение. Рекомендуется, по возможности, использовать естественный свет, при этом избегать попадания прямых солнечных лучей на монитор. Естественный свет снижает напряжение глаз и улучшает общее самочувствие;

- искусственное освещение. Для работы в вечернее время рекомендуется использовать искусственное освещение, такие как настольная лампа, комнатное освещение и т.д. Предпочтительно комбинировать освещения для более лучшего эффекта. Освещение должно быть ярким, чтобы избежать напряжения глаз, но не слишком интенсивным, чтобы исключить появления бликом на экране монитора. Также нужно учитывать правильное расположение источников света, локальный

источник, например, настольная лампа, должен быть установлен сбоку от монитора, чтобы не создавать прямых бликов на экране;

– настройка монитора. Правильная настройка яркости и контрастности монитора, также может уменьшить напряжение глаз при длительной работе за ПК. Рекомендуется в вечернее время использовать ночной режим, который делает цвета экрана более теплыми. Это поможет уменьшить нагрузку на глаза, а также улучшить качество сна.

2.3 Требования к режиму отдыха

Чтобы избежать проблем со здоровьем при длительной работе за компьютером, необходимо ежечасно делать небольшой перерыв, для выполнения разминки, чтобы избежать мышечного напряжения и улучшить кровообращение в суставах и мышцах.

Не стоит забывать и про глаза, которые так же подвержены воздействию, поэтому будет полезно сделать зарядку, которая включает в себя круговые движения открытыми глазами, смена точки фокусировки взгляда с близкой до дальней.

Важно также соблюдать строгий график работы и отдыха, ведь постоянный труд может привести к негативным последствиям. Необходимо составить четкое расписание и расписать план работы, для оптимизации своего времени. Не следует работать в дни отдыха, переработки могут плохо сказаться на вашем физическом и ментальном здоровье.

При соблюдении данных рекомендаций вы сможете обеспечить себе комфортную работу с минимальным риском для своего здоровья.

Заключение

В ходе выполнения данной выпускной квалификационной работы, было разработано Android-приложение – некастодиальный мультивалютный криптокошелек для платформы Android. В процессе разработки были освоены современные подходы к созданию защищённых мобильных приложений в среде Android Studio с использованием языков Kotlin и Java, а также криптографических библиотек (например, BitcoinJ, Web3j) и системного хранилища ключей Android Keystore для изоляции приватных ключей.

Разработанное приложение соответствует принципам некастодиальности: все криптографические ключи генерируются, хранятся и используются исключительно на устройстве пользователя. Реализована поддержка нескольких криптовалют (BTC, ETH, USDT и др.), а также функции создания и импорта кошельков по сид-фразе, отправки и получения средств, просмотра баланса и истории транзакций. Приложение обеспечивает надёжную защиту ключей от несанкционированного доступа и компрометации.

В перспективе возможны следующие улучшения: добавление поддержки дополнительных криптовалют и токенов (например, BSC, Solana, TON), интеграция с аппаратными кошельками (Ledger, Trezor), внедрение биометрической аутентификации для подтверждения операций, реализация резервного копирования с шифрованием, а также расширение аналитических модулей и улучшение пользовательского интерфейса для более удобного управления активами.

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		68

Список использованных источников

1. Разработка консольных приложений с помощью Microsoft Visual Studio 2017: учебное пособие / Ю.В. Тракимус – Новосибирск: Изд-во НГТУ, 2018. – 3 с.
2. Учебное пособие по языку Java Часть 1: учебное пособие / А. В. Гаврилов, О.А. Дегтерьева, И.А. Лезин, ИВ. Лезина – Самара, 2010. – 7-9 с.
3. Metanit.com: «Руководство по языку Kotlin». – [Электронный ресурс] / Режим доступа: <https://metanit.com/kotlin/tutorial/>, свободный. (дата обращения: 17.04.2026).
4. Введение в архитектурные шаблоны проектирования: MVC, MVP, MVVM. [Электронный ресурс] / Режим доступа: <https://gochadev.ru/2024/02/13/vvedenie-v-arhitekturnie-shablony-proektirovaniya-mvc-mvp-mvvm/>, свободный. (дата обращения 02.05.2024).
5. Корниенко С.А., Гохович В.А. Обзор криптоалгоритмов RSA и AES. Оценка их криптостойкости. // Синергия наук. 2017. № 14. – С. 460-466. [Электронный ресурс] / Режим доступа: <http://synergy-journal.ru/archive/article0858>, свободный. (дата обращения: 10.05.2026).

Приложение А

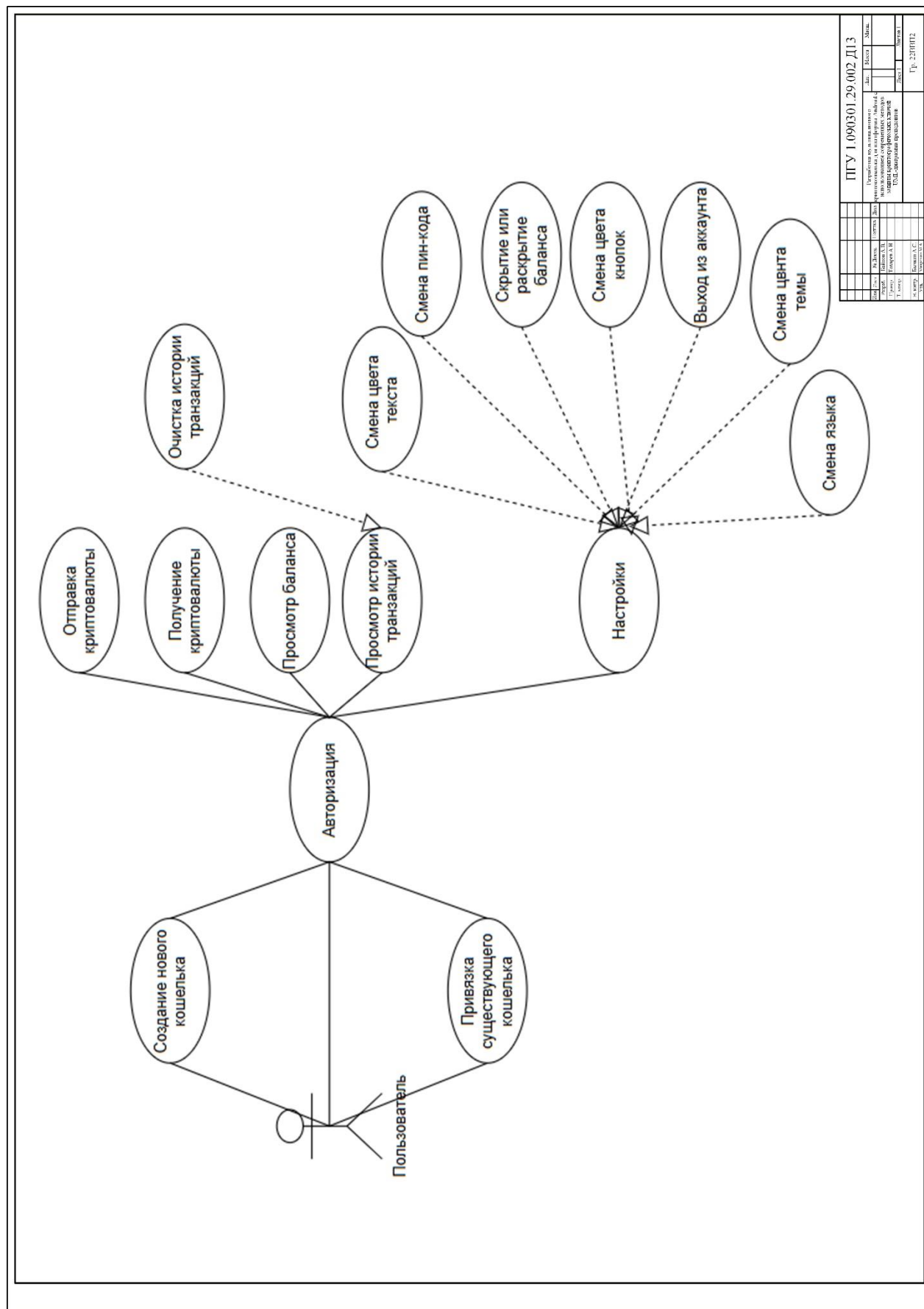
(обязательное)

Разработка мультивалютного криптокошелька для платформы Android с использованием современных методов защиты криптографических ключей

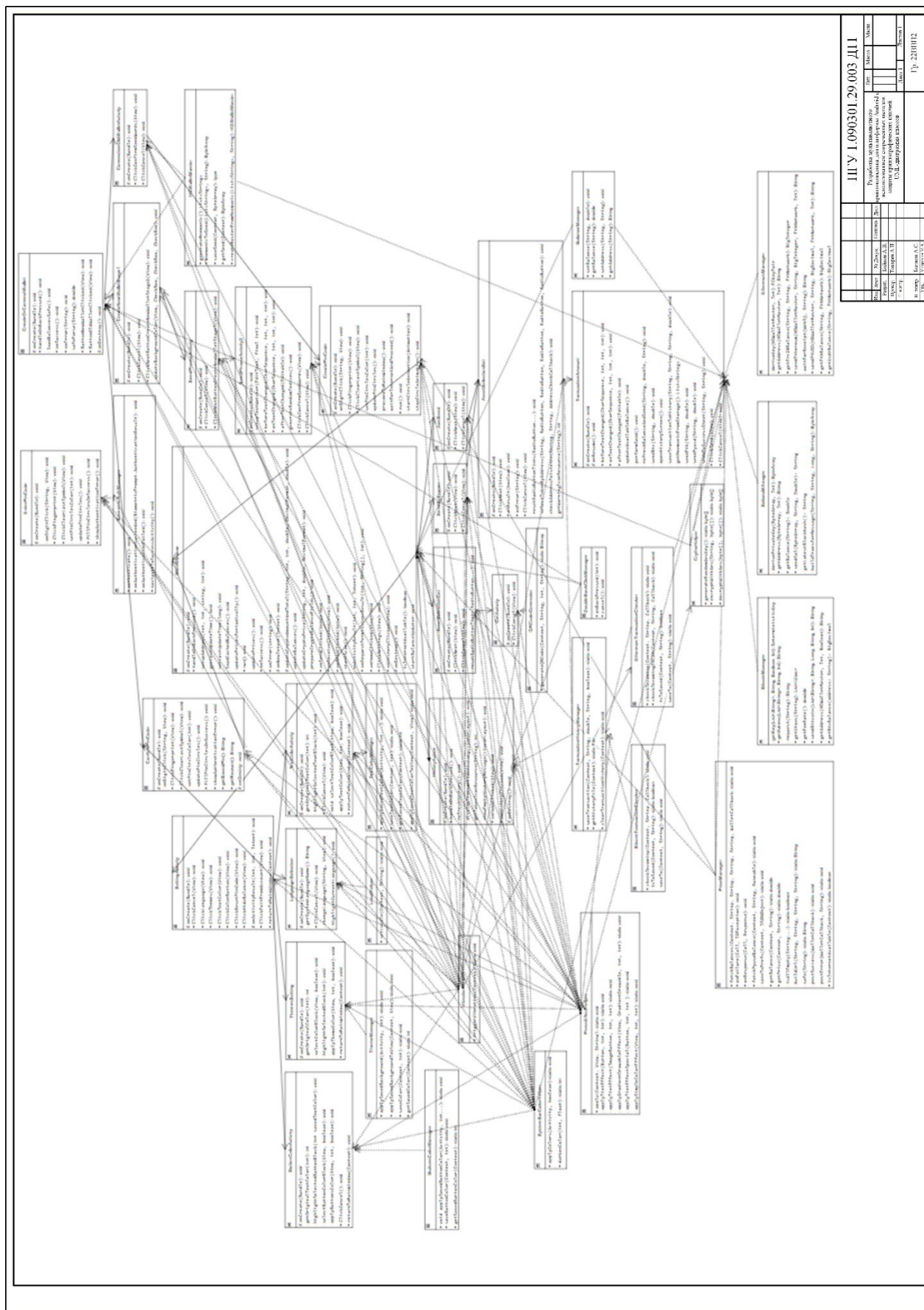
UML-диаграммы

					ПГУ 1.090301.29.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

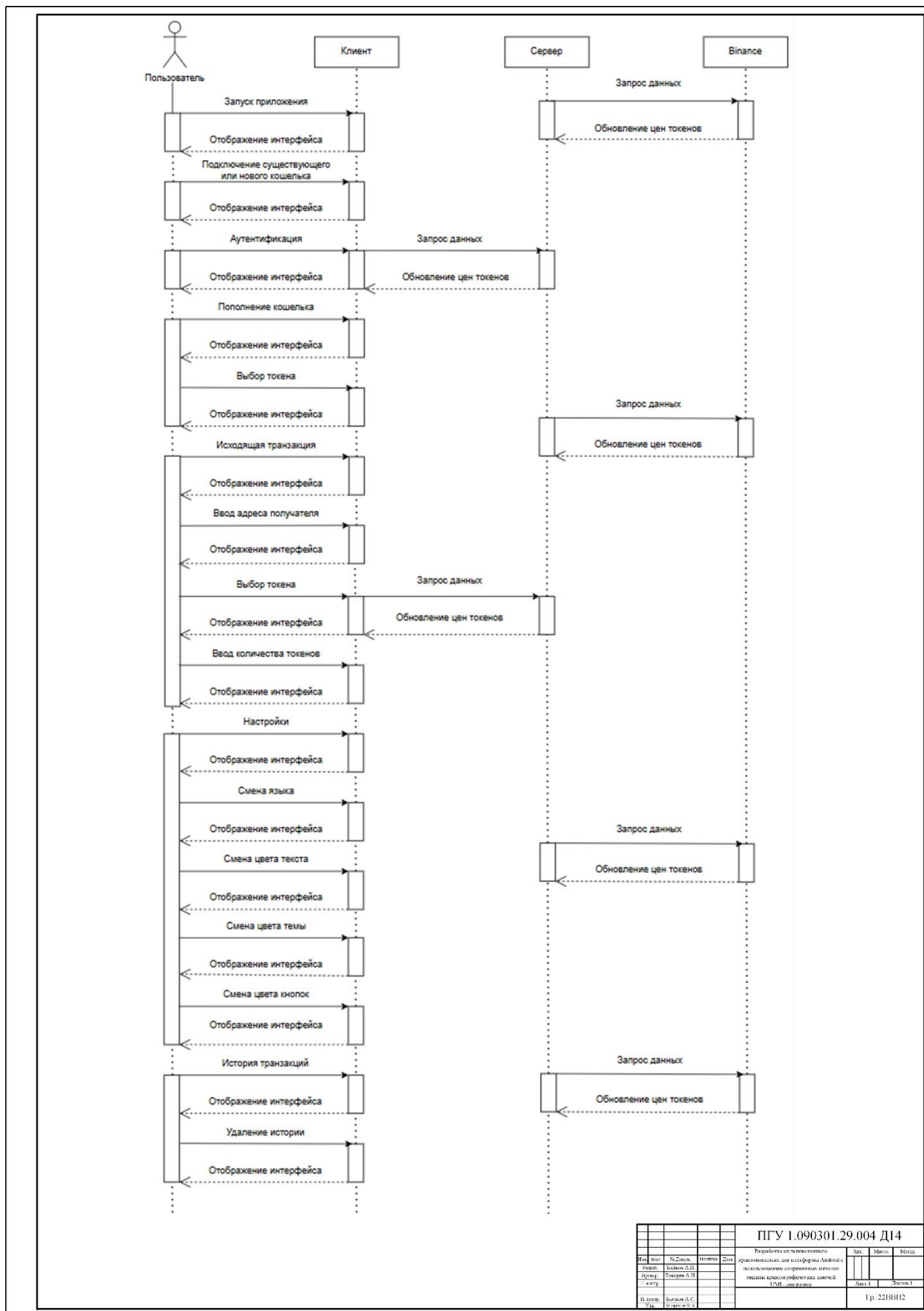
Приложение А.1 – UML-диаграмма прецедентов



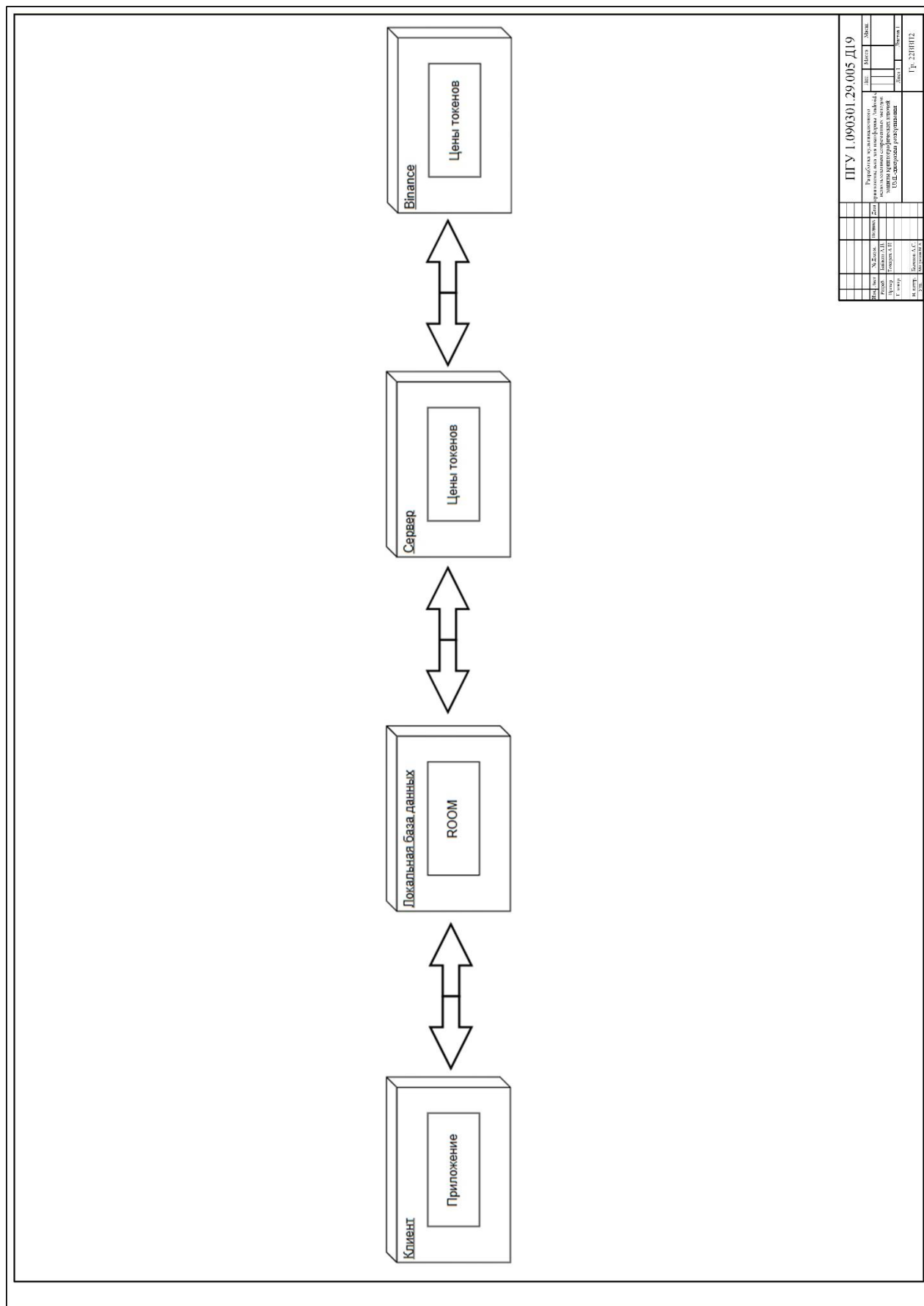
Приложение А.2 – UML-диаграмма классов



Приложение А.3 – UML-диаграмма последовательностей



Приложение А.4 – UML-диаграмма развёртывания



ПГУ 1.090301.29.005 Д19									
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель
Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель	Исполнитель

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.29.001 ПЗ

Лист

74

Приложение Б

(обязательное)

Разработка мультивалютного криптокошелька для платформы Android с использованием современных методов защиты криптографических ключей

Листинг программы

					ПГУ 1.090301.29.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

Файл MainWindow.java

```
package com.example.coldbitwallet;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Locale;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;
import android.Manifest;
import android.app.AlertDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.content.res.ColorStateList;
import android.net.ConnectivityManager;
import android.net.NetworkCapabilities;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.provider.Settings;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ImageButton;
import androidx.activity.EdgeToEdge;
import androidx.activity.OnBackPressedCallback;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.example.coldbitwallet.wallet.bitcoin.BitcoinManager;
import com.example.coldbitwallet.wallet.ethereum.EthereumManager;
import com.journeyapps.barcodescanner.CaptureActivity;
public class MainWindow extends AppCompatActivity {
    private static final int REQUEST_CODE_SCAN = 1; // Код для
запроса сканирования
    private static final int CAMERA_PERMISSION_REQUEST_CODE = 100;
// Код запроса разрешения на камеру
    private boolean isBalanceHidden;
    private DoubleBackClickManager doubleBackPressHandler;
```

```

        private Handler LoopForPriceCryptho = new
Handler(Looper.getMainLooper()); // Для обновления цен, раз в пять
минут
        private static final String PREFS_NAME = "PriceUpdatePrefs"; //
Ключ для SharedPreferences
        private static final String KEY_LAST_UPDATE_TIME =
"lastUpdateTime"; // Время последнего обновления
        private static final long UPDATE_INTERVAL = 300000; // 5 минут
        private boolean isConnected;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable(this);
// SETTINGS
    String lang = getSharedPreferences("settings", MODE_PRIVATE)
        .getString("language", "system");
    LocaleHelper.setLocale(this, lang);
    SharedPreferences settings = getSharedPreferences("settings",
MODE_PRIVATE);
    isBalanceHidden = settings.getBoolean("hide_balance", false);
    setContentView(R.layout.activity_main_window);
    doubleBackPressedHandler = new DoubleBackClickManager(this);
    SystemBarColorHelper.applyColors(this, true);
    ThemeManager.applySavedBackground(this, R.id.mainLayout);
    ThemeManager.applySavedBackground(this, R.id.main);
// UI INIT
    prepareCryptoBalanceFields();
    int savedColor = TextColorManager.getSavedTextColor(this);
    int[] textViewIds = {
        R.id.textHistory, R.id.textHome, R.id.ReceiveHint,
R.id.ScanHint,
        R.id.textView2, R.id.BalanceText,
        R.id.title_btc, R.id.amount_btc, R.id.price_per_btc,
R.id.total_value_btc, R.id.address_btc,
        R.id.title_eth, R.id.amount_eth, R.id.price_per_eth,
R.id.total_value_eth, R.id.address_eth,
        R.id.title_ton, R.id.amount_ton, R.id.price_per_ton,
R.id.total_value_ton, R.id.address_ton,
        R.id.title_usdt, R.id.amount_usdt, R.id.price_per_usdt,
R.id.total_value_usdt, R.id.address_usdt,
        R.id.title_sol, R.id.amount_sol, R.id.price_per_sol,
R.id.total_value_sol, R.id.address_sol
    };
    for (int id : textViewIds) {
        TextColorManager.applySavedTextColor(this, id);
    }
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
        Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

```

        v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
        return insets;
    });
    getOnBackPressedDispatcher().addCallback(this, new
OnBackPressedCallback(true) {
        @Override
        public void handleOnBackPressed() {
            doubleBackPressHandler.onBackPressed(R.string.toast_exit_app);
        }
    });
    // BALANCE MANAGER INIT
    SharedPreferences prefs = getSharedPreferences("MyPrefs",
MODE_PRIVATE);
    BalanceManager bm = BalanceManager.getInstance();
    // адпека
    bm.setAddress("btc", prefs.getString("bitcoin_address", ""));
    bm.setAddress("eth", prefs.getString("ethereum_address", ""));
    bm.setAddress("sol", prefs.getString("solana_address", ""));
    bm.setAddress("ton", prefs.getString("ton_address", ""));
    bm.setAddress("usdt", prefs.getString("ethereum_address", ""));
    // АДПЕКА UI
    setupAddress(R.id.address_btc, R.id.expand_btc, "btc", savedColor);
    setupAddress(R.id.address_eth, R.id.expand_eth, "eth", savedColor);
    setupAddress(R.id.address_ton, R.id.expand_ton, "ton", savedColor);
    setupAddress(R.id.address_sol, R.id.expand_sol, "sol",
savedColor);
    // INTERNET
    TextView internetWarningText =
findViewById(R.id.internetWarningText);
    internetWarningText.setVisibility(isInternetAvailable() ?
View.INVISIBLE : View.VISIBLE);
    registerReceiver(internetReceiver,
        new
IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION));
    loadCachedBalances(); // кеш
    startBalanceUpdater(); // RPC
    updateBalances(); // UIые
    updatePricesPeriodically();
}
private void setupAddress(int textId, int expandId, String token,
int color) {
    TextView addressView = findViewById(textId);
    ImageButton expand = findViewById(expandId);
    String address = BalanceManager.getInstance().getAddress(token);
    addressView.setText(address);
    expand.setImageTintList(ColorStateList.valueOf(color));
    expand.setOnClickListener(v -> {
        if (addressView.getVisibility() == View.GONE) {
            addressView.setVisibility(View.VISIBLE);
            expand.setImageResource(R.drawable.ic_arrow_up);
        }
    });
}

```

```

        } else {
            addressView.setVisibility(View.GONE);
            expand.setImageResource(R.drawable.ic_arrow_down);
        }
    });
}

// Метод для получения времени последнего обновления из
SharedPreferences
private long getLastUpdateTime() {
    SharedPreferences prefs = getSharedPreferences(PREFS_NAME,
MODE_PRIVATE);
    return prefs.getLong(KEY_LAST_UPDATE_TIME, 0); // Возвращаем 0,
если обновлений не было
}

// Метод для обновления времени последнего обновления в
SharedPreferences
private void setLastUpdateTime(long time) {
    SharedPreferences prefs = getSharedPreferences(PREFS_NAME,
MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();
    editor.putLong(KEY_LAST_UPDATE_TIME, time);
    editor.apply();
}

private void loadCachedBalances() {
    SharedPreferences prefs = getSharedPreferences("MyPrefs",
MODE_PRIVATE);
    BalanceManager bm = BalanceManager.getInstance();

    bm.setBalance("btc",
        Double.parseDouble(prefs.getString("btc_balance",
"0.0")));
    bm.setBalance("eth",
        Double.parseDouble(prefs.getString("eth_balance",
"0.0")));

    bm.setBalance("sol",
        Double.parseDouble(prefs.getString("sol_balance",
"0.0")));

    bm.setBalance("ton",
        Double.parseDouble(prefs.getString("ton_balance",
"0.0")));

    bm.setBalance("usdt",
        Double.parseDouble(prefs.getString("usdt_balance",
"0.0")));
}

// Метод для обновления цен раз в 5 минут
private void updatePricesPeriodically() {
    LoopForPriceCryphto.postDelayed(new Runnable() {
        @Override

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		79

```

        public void run() {
            // Проверяем, прошло ли 5 минут с последнего обновления
            long lastUpdateTime = getLastUpdateTime();
            long currentTime = System.currentTimeMillis();
            // Если прошло больше 5 минут, обновляем цены
            if (currentTime - lastUpdateTime > UPDATE_INTERVAL) {
                updatePrices();
            }
            LoopForPriceCryptho.postDelayed(this, UPDATE_INTERVAL);
// Запускаем задачу снова через 5 минут
        }
    }, UPDATE_INTERVAL); // Первый запуск через 5 минут
}
// Метод для обновления цен
private void updatePrices() {
    if (!isInternetAvailable()) {
        Log.d("PriceManager", "Нет интернета. Используем локальные
данные.");
        updateBalances();
        return;
    }
    Log.d("PriceManager", "Запрос к серверу...");
    // Адреса
    String btcAddress =
BalanceManager.getInstance().getAddress("BTC");
    String ethAddress =
BalanceManager.getInstance().getAddress("ETH");
    String tonAddress =
BalanceManager.getInstance().getAddress("TON");
    String solAddress =
BalanceManager.getInstance().getAddress("SOL");
    PriceManager.fetchBalances(
        this,
        btcAddress,
        ethAddress,
        tonAddress,
        solAddress,
        new PriceManager.WalletCallback() {
            @Override
            public void onSuccess() {
                Log.d("PriceManager", "Данные обновлены");
                updateBalances();
                setLastUpdateTime(System.currentTimeMillis());
            }
            @Override
            public void onError(String error) {
                Log.e("PriceManager", "Ошибка: " + error);
                updateBalances(); // старые данные
            }
        }
    );
}

```

```

}
@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    setIntent(intent);
    // Проверяем, прошло ли 5 минут с последнего обновления
    long lastUpdateTime = getLastUpdateTime();
    long currentTime = System.currentTimeMillis();
    // Если прошло больше 5 минут, разрешаем обновление
    if (currentTime - lastUpdateTime > UPDATE_INTERVAL) {
        updatePrices(); // Обновляем цены при запуске приложения
    }
}

private void updateCryptoAmountAndTotal(
    String cryptoKey,
    int amountId,
    int totalValueId,
    double price,
    DecimalFormat df,
    double balance
) {
    TextView amountTextView = findViewById(amountId);
    TextView totalValueTextView = findViewById(totalValueId);
    amountTextView.setText(df.format(balance)); // Отображаем
количество токенов

    if (!cryptoKey.equalsIgnoreCase("usdt")) {
        double totalValue = balance * price;
        totalValueTextView.setText("$" + df.format(totalValue));
    } else {
        totalValueTextView.setText(""); // USDT без общей стоимости
    }
}

private void updateBalances() {

    DecimalFormatSymbols symbols = new
    DecimalFormatSymbols(Locale.US);

    DecimalFormat df = new DecimalFormat("0.00000", symbols);
    df.setRoundingMode(RoundingMode.CEILING);

    DecimalFormat df2 = new DecimalFormat("0.####", symbols);
    df2.setRoundingMode(RoundingMode.CEILING);

    prepareCryptoBalanceFields(); // 0. UI reset

    SharedPreferences prefs =
    getSharedPreferences(PriceManager.PREFS_NAME, MODE_PRIVATE); // 1.
    READ PRICES FROM SharedPrefs

    double btcPrice = prefs.getFloat("btc_price", 0f);

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		81

```

double ethPrice = prefs.getFloat("eth_price", 0f);
double tonPrice = prefs.getFloat("ton_price", 0f);
double solPrice = prefs.getFloat("sol_price", 0f);
double usdtPrice = 1.0;
// 2. UPDATE PRICE UI
updateCryptoPrice("btc", R.id.price_per_btc, btcPrice, df2);
updateCryptoPrice("eth", R.id.price_per_eth, ethPrice, df2);
updateCryptoPrice("ton", R.id.price_per_ton, tonPrice, df2);
updateCryptoPrice("sol", R.id.price_per_sol, solPrice, df2);
updateCryptoPrice("usdt", R.id.price_per_usdt, usdtPrice, df2);
if (isBalanceHidden) return; // 3. HIDE BALANCE CHECK
// 4. READ BALANCES ONLY FROM PREFS
double btc = Double.parseDouble(prefs.getString("btc_balance",
"0.0"));
double eth = Double.parseDouble(prefs.getString("eth_balance",
"0.0"));
double ton = Double.parseDouble(prefs.getString("ton_balance",
"0.0"));
double sol = Double.parseDouble(prefs.getString("sol_balance",
"0.0"));
String usdtRaw = prefs.getString("usdt_balance", "0.0");
double usdt = Double.parseDouble(usdtRaw);
// 5. UPDATE TOKEN AMOUNTS + TOTALS
updateCryptoAmountAndTotal("btc", R.id.amount_btc,
R.id.total_value_btc, btcPrice, df2, btc);
updateCryptoAmountAndTotal("eth", R.id.amount_eth,
R.id.total_value_eth, ethPrice, df2, eth);
updateCryptoAmountAndTotal("ton", R.id.amount_ton,
R.id.total_value_ton, tonPrice, df2, ton);
updateCryptoAmountAndTotal("sol", R.id.amount_sol,
R.id.total_value_sol, solPrice, df2, sol);
updateCryptoAmountAndTotal("usdt", R.id.amount_usdt,
R.id.total_value_usdt, usdtPrice, df2, usdt);
}
// 6. TOTAL BALANCE
double total =
    btc * btcPrice +
        eth * ethPrice +
        ton * tonPrice +
        sol * solPrice +
        usdt;
TextView balanceText = findViewById(R.id.BalanceText);
balanceText.setText("$" + df2.format(total));
// Обновление только цены токена
private void updateCryptoPrice(String cryptoKey, int pricePerId,
double price, DecimalFormat df) {
    TextView pricePerTextView = findViewById(pricePerId);

    if (!cryptoKey.equalsIgnoreCase("usdt")) {
        String formattedPrice = df.format(price);

```

```

        pricePerTextView.setText(String.format(Locale.US, "1 %s -
%s", cryptoKey.toUpperCase(), formattedPrice));
        pricePerTextView.setVisibility(View.VISIBLE);
    } else {
        pricePerTextView.setVisibility(View.GONE);
    }
}
private void prepareCryptoBalanceFields() {
    int[] amountIds = {
        R.id.amount_btc, R.id.amount_eth, R.id.amount_ton,
        R.id.amount_usdt, R.id.amount_sol
    };
    int[] totalValueIds = {
        R.id.total_value_btc, R.id.total_value_eth,
        R.id.total_value_ton, R.id.total_value_usdt, R.id.total_value_sol
    };
    for (int id : amountIds) {
        TextView amountTextView = findViewById(id);
        amountTextView.setText(isBalanceHidden ? "?" : "");
    }
    for (int id : totalValueIds) {
        TextView totalValueTextView = findViewById(id);
        totalValueTextView.setText("");
    }
    // текст общего баланса сразу скрыть/обновить
    TextView balanceText = findViewById(R.id.BalanceText);
    balanceText.setText(isBalanceHidden ? "?" : "");
}
// Обработчик для нажатия на кнопку "Отправить"
public void onSendClicked(View view) {
    PressEffectHelper.apply(this, view, "sss");
    if (!isConnected) {
        Toast.makeText(this, R.string.check_internet_connection,
        Toast.LENGTH_SHORT).show();
        return;
    }
    // Если баланс скрыт, то ничего не делаем
    if (isBalanceHidden) { Toast.makeText(this,
    R.string.toast_transaction_with_hidden_balance,
        Toast.LENGTH_SHORT).show(); return;}
    Intent intent = new Intent(MainWindow.this, JustSend.class);
    startActivity(intent);
}
// Обработчик для нажатия на кнопку "Получить"
public void onReceiveClicked(View view) {
    PressEffectHelper.apply(this, view, "sss");
    Intent intent = new Intent(MainWindow.this,
    AssetSelectionGet.class);
    startActivity(intent);
}
// Обработчик для нажатия на кнопку "Сканировать"

```

```

public void onScanClicked(View view) {
    PressEffectHelper.apply(this, view, "sss");
    if (!isConnected) {
        Toast.makeText(this, R.string.check_internet_connection,
            Toast.LENGTH_SHORT).show();
        return;
    }
    // Если баланс скрыт, то ничего не делаем
    if (isBalanceHidden) { Toast.makeText(this,
        R.string.toast_transaction_with_hidden_balance,
            Toast.LENGTH_SHORT).show(); return;}
    // Проверка разрешения на камеру
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
        // Если разрешение не предоставлено, запрашиваем его
        if
        (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.CAMERA)) {
            // Появляется объяснение пользователю, почему необходимо
            разрешение
            Toast.makeText(this,
                getString(R.string.toast_qr_permission), Toast.LENGTH_SHORT).show();
        }
        // Запрашиваем разрешение
        ActivityCompat.requestPermissions(this, new
            String[]{Manifest.permission.CAMERA},
            CAMERA_PERMISSION_REQUEST_CODE);
    } else {
        startScan(); // Разрешение на камеру уже есть, запускаем
        сканирование
    }
}
// Метод для начала сканирования
private void startScan() {
    Intent intent = new Intent(MainWindow.this,
        CaptureActivity.class); // Запуск активности CaptureActivity для
        сканирования QR-кодов
    startActivityForResult(intent, REQUEST_CODE_SCAN);
}
// Метод для обработки результата сканирования QR-кода
@Override
protected void onActivityResult(int requestCode, int resultCode,
    Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE_SCAN && resultCode == RESULT_OK
        && data != null) {
        String scanResult = data.getStringExtra("SCAN_RESULT"); //
        Получаем результат сканирования QR-кода
        // Обработка результата сканирования
        if (scanResult != null) {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		84

```

        Intent resultIntent = new Intent(MainWindow.this,
SendByScanner.class);
        resultIntent.putExtra("SCAN_RESULT", scanResult); //
передаем результат в другую активность
        startActivity(resultIntent); // запускаем новую
активность с результатом
    }
}
// Обработчик для результата запроса разрешений
@Override
public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
    if (requestCode == CAMERA_PERMISSION_REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            // Разрешение на камеру получено, можно запускать
сканирование
            startScan();
        } else {
            // Разрешение на камеру отклонено, показываем сообщение
Toast.makeText(this,
getString(R.string.toast_qr_permission), Toast.LENGTH_SHORT).show();
            // Проверяем, отклонено ли разрешение навсегда
            if
(!ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.CAMERA)) {
                // Если "Не показывать снова", предлагаем перейти в
настройки
                new AlertDialog.Builder(this)
                .setMessage(getString(R.string.go_settings_activation_scan_text))
                .setPositiveButton(getString(R.string.go_to_settings_text), (dialog,
which) -> {
                    // Переход в настройки приложения
                    Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                    Uri uri = Uri.fromParts("package",
getPackageName(), null);
                    intent.setData(uri);
                    startActivity(intent);
                })
                .setNegativeButton(getString(R.string.cancel_text), (dialog, which)
-> dialog.dismiss())
                .create()
                .show();
            }
        }
    }
}

```

```

public void onHomeClicked(View view) {
    PressEffectHelper.apply(this, view, "sss");
}
public void ClickSettings(View view) {
    PressEffectHelper.apply(this, view, "sss");
    Intent intent = new Intent(MainWindow.this,
SettingsActivity.class);
    startActivity(intent);
}
public void onHistoryClicked(View view) {
    PressEffectHelper.apply(this, view, "sss");
    Intent intent = new Intent(MainWindow.this,
HistoryAction.class);
    startActivity(intent);
    finish();
}
@Override
protected void onDestroy() {
    super.onDestroy();
    LoopForPriceCryptho.removeCallbacksAndMessages(null);
    unregisterReceiver(internetReceiver);
    if (doubleBackPressHandler != null) {
        doubleBackPressHandler.cancel();
    }
    if (scheduler != null) {
        scheduler.shutdownNow();
    }
}

    private boolean isInternetAvailable() {
        ConnectivityManager cm =
            (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        if (cm != null) {
            NetworkCapabilities capabilities =
cm.getNetworkCapabilities(cm.getActiveNetwork());
            return capabilities != null &&
capabilities.hasCapability(NetworkCapabilities.NET_CAPABILITY_INTERN
ET);
        }
        return false;
    }
    private final BroadcastReceiver internetReceiver = new
BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            isConnected = isInternetAvailable();

            TextView internetWarningText =
findViewById(R.id.internetWarningText);

            if (isConnected) {

```

```

        internetWarningText.setVisibility(View.INVISIBLE);
    } else {
        internetWarningText.setVisibility(View.VISIBLE);
    }
}

};
private ScheduledExecutorService scheduler;
private void startBalanceUpdater() {
    scheduler = Executors.newSingleThreadScheduledExecutor();
    scheduler.scheduleWithFixedDelay(() -> {
        try {
            String btcAddress =
BalanceManager.getInstance().getAddress("btc");
            String ethAddress =
BalanceManager.getInstance().getAddress("eth");

            SharedPreferences prefs =
                getSharedPreferences("wallet_data",
MODE_PRIVATE);

            BalanceManager bm = BalanceManager.getInstance();
            // BTC
            BigDecimal btc =
BitcoinManager.INSTANCE.getBtcBalance(btcAddress);
            Log.d("BTC_BALANCE", "BTC = " +
btc.toString());
            bm.setBalance("btc", btc.doubleValue());
            // ETH
            BigDecimal ethBd =
EthereumManager.INSTANCE.getEthBalance(
                ethAddress,
                EthereumManager.EthNetwork.SEPOLIA
            );
            double eth = ethBd.doubleValue();
            Log.d("ETH_BALANCE", "ETH = " + eth);
            bm.setBalance("eth", eth);
            // USDT (ERC20)
            BigDecimal usdtBd =
EthereumManager.INSTANCE.getUsdtBalance(
                ethAddress,
                EthereumManager.EthNetwork.SEPOLIA
            );

            double usdt = usdtBd.doubleValue();

            Log.d("USDT_BALANCE", "USDT RAW(BigDecimal) = " +
usdtBd.toString());
            Log.d("USDT_BALANCE", "USDT DOUBLE = " + usdt);

            bm.setBalance("usdt", usdt);

            // SINGLE CACHE WRITE

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						87
Изм.	Лист	№ докум.	Подпись	Дата		

```

        prefs.edit()
            .putString("btc_balance",
btc.toPlainString())
            .putString("eth_balance",
String.valueOf(eth))
            .putString("usdt_balance",
String.valueOf(usdt))
            .apply();
        // UI
        runOnUiThread(this::updateBalances);

    } catch (Exception e) {
        Log.e("BALANCE", "Updater error", e);
    }

}, 0, 15, TimeUnit.SECONDS);
}
}

```

Файл AssetSelectionGet.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;
import android.widget.RadioGroup;
import android.widget.RadioButton;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class AssetSelectionGet extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_asset_selection_get);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        // Список всех id элементов
    }
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		88

```

        int[] textViewIds = {
            // Системные штуки
            R.id.Recipient, R.id.NextButtonSendByScanner,
R.id.token_btc, R.id.token_eth, R.id.token_ton,
            R.id.token_usdt, R.id.token_sol
        }
        // Применяем цвет ко всем id
        for (int id : textViewIds) {
            TextColorManager.applySavedTextColor(this, id);
        }
        // Применение сохраненного цвета для кнопок
        Button nextButton =
findViewById(R.id.NextButtonSendByScanner);
        PressEffectHelper.apply(this, nextButton, "spec");
        // Применяем обводку
        LinearLayout tokenAddressContainer =
findViewById(R.id.tokenList);
        tokenAddressContainer.setBackgroundResource(R.drawable.rounded_bg);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });
        RadioGroup tokenGroup = findViewById(R.id.tokenList);
        RadioButton token_btc = findViewById(R.id.token_btc);
        RadioButton token_usdt = findViewById(R.id.token_usdt);
        RadioButton token_eth = findViewById(R.id.token_eth);
        RadioButton token_ton = findViewById(R.id.token_ton);
        RadioButton token_sol = findViewById(R.id.token_sol);
        tokenGroup.setOnCheckedChangeListener((group, checkedId) ->
{
            resetRadioButtonTints(token_btc, token_eth, token_usdt,
token_ton, token_sol);
            if (checkedId == R.id.token_btc) {
token_btc.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.BTCColor));
            } else if (checkedId == R.id.token_usdt) {
token_usdt.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.USDTColor));
            } else if (checkedId == R.id.token_eth) {
token_eth.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.ETHColor));
            } else if (checkedId == R.id.token_ton) {
token_ton.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.TONColor));
            } else if (checkedId == R.id.token_sol) {

```

```

token_sol.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.SOLColor));
        } else {
            RadioButton selected = findViewById(checkedList);
selected.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.SystemColor));
        }
    });
}

public void ClickNext(View view) {
    PressEffectHelper.apply(this, view, "spec");
    RadioGroup radioGroup = findViewById(R.id.tokenList);
    int selectedId = radioGroup.getCheckedRadioButtonId();
    if (selectedId == -1) {
        Toast.makeText(this,
getString(R.string.toast_select_token), Toast.LENGTH_SHORT).show();
        VibrationHelper.triggerVibration(this);
    } else {
        Intent intent = new Intent(AssetSelectionGet.this,
GetActivity.class); // Ведет к экрану получения
        RadioButton selectedRadio = findViewById(selectedId);
        String fullText = selectedRadio.getText().toString();
        String selectedToken =
fullText.split("\\\\(")[1].replace(")", "").trim();
        intent.putExtra("selected_token", selectedToken);
        startActivity(intent);
    }
}

public void ClickCancel(View view) {
    PressEffectHelper.apply(this, view, "sss");
    finish();
}

private void resetRadioButtonTints(RadioButton... buttons) {
    for (RadioButton btn : buttons) {

btn.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.SystemColor));
    }
}
}

```

Файл AssetSelectionSet.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		90

```

import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.RadioGroup;
import android.widget.RadioButton;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import org.json.JSONArray;
import org.json.JSONObject;
import org.json.JSONException;
import java.io.IOException;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class AssetSelectionSet extends AppCompatActivity {
    private String scanResult;
    public static boolean ethAddressPreviouslyWarned = false;
    public static boolean usdtAddressPreviouslyWarned = false;
    public static boolean solAddressPreviouslyWarned = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_asset_selection_set);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        // Список всех id элементов
        int[] textViewIds = {
            // Системные штуки
            R.id.Recipient, R.id.NextButtonSendByScanner,
R.id.token_btc, R.id.token_eth, R.id.token_ton,
            R.id.token_usdt, R.id.token_sol, R.id.warningText
        };
        // Применяем цвет ко всем id
        for (int id : textViewIds) {
            TextColorManager.applySavedTextColor(this, id);
        }
        ButtonsColorManager.applySavedButtonColor(this,
R.id.NextButtonSendByScanner);
        Button nextButton =

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		91

```

findViewById(R.id.NextButtonSendByScanner);
    PressEffectHelper.apply(this, nextButton, "spec");
    // Применяем обводку
    LinearLayout tokenAddressContainer =
findViewById(R.id.tokenList);
tokenAddressContainer.setBackgroundResource(R.drawable.rounded_bg);
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
    return insets;
});
scanResult = getIntent().getStringExtra("SCAN_RESULT");
RadioGroup tokenGroup = findViewById(R.id.tokenList);
// Получаем ссылки на кнопки
RadioButton token_btc = findViewById(R.id.token_btc);
RadioButton token_usdt = findViewById(R.id.token_usdt);
RadioButton token_eth = findViewById(R.id.token_eth);
RadioButton token_ton = findViewById(R.id.token_ton);
RadioButton token_sol = findViewById(R.id.token_sol);
// Логика для выбора токена по адресу
selectTokenByAddress(scanResult, token_btc, token_eth,
token_sol, token_usdt, token_ton);
// Устанавливаем слушатель
tokenGroup.setOnCheckedChangeListener((group, checkedId) ->
{
    // Сбрасываем цвет у всех
    resetRadioButtonTints(token_btc, token_eth, token_usdt,
token_ton, token_sol);
    // Устанавливаем нужный цвет только выбранной кнопке
    if (checkedId == R.id.token_btc) {
token_btc.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.BTCColor));
    } else if (checkedId == R.id.token_usdt) {
token_usdt.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.USDTColor));
    } else if (checkedId == R.id.token_sol) {
token_sol.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.SOLColor));
    } else if (checkedId == R.id.token_eth) {
token_eth.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.ETHColor));
    } else if (checkedId == R.id.token_ton) {
token_ton.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.TONColor));
    } else {
        RadioButton selected = findViewById(checkedId);
selected.setButtonTintList(ContextCompat.getColorStateList(this,
R.color.SystemColor));
    }
}
}

```

```

    }
    });
public void ClickNext(View view) {
    PressEffectHelper.apply(this, view, "spec");
    RadioGroup radioGroup = findViewById(R.id.tokenList);
    int selectedId = radioGroup.getCheckedRadioButtonId();
    if (selectedId == -1) {
        Toast.makeText(this, getString(R.string.toast_select_token),
Toast.LENGTH_SHORT).show();
        VibrationHelper.triggerVibration(this);
    } else {
        RadioButton selectedRadio = findViewById(selectedId);
        String fullText = selectedRadio.getText().toString();
        String selectedToken = fullText.split("\\(")[1].replace(")",
"".trim());
        String recipientAddress = scanResult;
        switch (selectedToken) {
            case "BTC":
                checkAddressExistence(selectedToken,
recipientAddress, new AddressCheckCallback() {
                    @Override
                    public void onResult(boolean exists) {
                        if (exists) {
                            Intent intent = new
Intent(AssetSelectionSet.this, TransactionAmount.class);
                            intent.putExtra("selected_token",
selectedToken);
                            intent.putExtra("SCAN_RESULT",
recipientAddress);
                            startActivity(intent);
                            LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
                            warningLayout.setVisibility(View.GONE);
                        } else {
                            VibrationHelper.triggerVibration(AssetSelectionSet.this);
                            LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
                            warningLayout.setVisibility(View.VISIBLE);
                            TextView warningText =
findViewById(R.id.warningText);
                            String warningMessage =
getString(R.string.warning_unvalid_address, selectedToken);
                            warningText.setText(warningMessage);
                            ImageView warningIcon =
findViewById(R.id.warningIcon);
                            int iconResId =
getWarningIconResource(selectedToken); // метод ниже
                            warningIcon.setImageResource(iconResId);
                        }
                    }
                })
            @Override

```

```

        public void onError(String errorMessage) {

Toast.makeText(AssetSelectionSet.this, "Ошибка сети",
Toast.LENGTH_SHORT).show();
VibrationHelper.triggerVibration(AssetSelectionSet.this);
        }
    });
    break;
    case "ETH":
        checkAddressExistence(selectedToken,
recipientAddress, new AddressCheckCallback() {
            @Override
            public void onResult(boolean exists) {
                if (exists || ethAddressPreviouslyWarned) {
                    // Переход дальше
                    Intent intent = new
Intent(AssetSelectionSet.this, TransactionAmount.class);
                    intent.putExtra("selected_token",
selectedToken);
                    intent.putExtra("SCAN_RESULT",
recipientAddress);
                    startActivity(intent);
                    LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
                    warningLayout.setVisibility(View.GONE);
                } else {
                    // Первый раз – предупреждение
                    ethAddressPreviouslyWarned = true;
                    Toast.makeText(AssetSelectionSet.this,
R.string.toast_warning_unvalid_address_true,
Toast.LENGTH_SHORT).show();
                    VibrationHelper.triggerVibration(AssetSelectionSet.this);
                    LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
                    warningLayout.setVisibility(View.VISIBLE);
                    TextView warningText =
findViewById(R.id.warningText);
                    String warningMessage =
getString(R.string.warning_unvalid_address_true, selectedToken);
                    warningText.setText(warningMessage);
                    ImageView warningIcon =
findViewById(R.id.warningIcon);
                    int iconResId =
getWarningIconResource(selectedToken);
                    warningIcon.setImageResource(iconResId);
                }
            }
        @Override
        public void onError(String errorMessage) {
            Toast.makeText(AssetSelectionSet.this,
"Ошибка сети", Toast.LENGTH_SHORT).show();

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		94

```

VibrationHelper.triggerVibration(AssetSelectionSet.this);
        }
    });
    break;
    case "USDT":
checkAddressExistence(selectedToken, recipientAddress, new
AddressCheckCallback() {
    @Override
    public void onResult(boolean exists) {
        if (exists || usdtAddressPreviouslyWarned) {
            Intent intent = new
Intent(AssetSelectionSet.this, TransactionAmount.class);
            intent.putExtra("selected_token",
selectedToken);
            intent.putExtra("SCAN_RESULT",
recipientAddress);
            startActivity(intent);
            LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
warningLayout.setVisibility(View.GONE);
        } else {
            usdtAddressPreviouslyWarned = true;
            Toast.makeText(
                AssetSelectionSet.this,
R.string.toast_warning_unvalid_address_true,
                Toast.LENGTH_SHORT
            ).show();
            VibrationHelper.triggerVibration(AssetSelectionSet.this);
            LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
warningLayout.setVisibility(View.VISIBLE);
            TextView warningText =
findViewById(R.id.warningText);
            String warningMessage = getString(
R.string.warning_unvalid_address_true,
                selectedToken
            );
            warningText.setText(warningMessage);
            ImageView warningIcon =
findViewById(R.id.warningIcon);
            int iconResId =
getWarningIconResource(selectedToken);
            warningIcon.setImageResource(iconResId);
        }
    }
    @Override
    public void onError(String errorMessage) {
        Toast.makeText(
            AssetSelectionSet.this,
            "Ошибка сети",

```

```

                                Toast.LENGTH_SHORT
                                ).show();
VibrationHelper.triggerVibration(AssetSelectionSet.this);
    }
    });
    break;
    case "SOL":
        checkAddressExistence(selectedToken,
recipientAddress, new AddressCheckCallback() {
            @Override
            public void onResult(boolean exists) {
                if (exists || solAddressPreviouslyWarned) {
                    Intent intent = new
Intent(AssetSelectionSet.this, TransactionAmount.class);
                    intent.putExtra("selected_token",
selectedToken);
                    intent.putExtra("SCAN_RESULT",
recipientAddress);
                    startActivity(intent);
                    LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
                    warningLayout.setVisibility(View.GONE);
                } else {
                    solAddressPreviouslyWarned = true;
                    Toast.makeText(AssetSelectionSet.this,
R.string.toast_warning_unvalid_address_true,
Toast.LENGTH_SHORT).show();
                    VibrationHelper.triggerVibration(AssetSelectionSet.this);
                    LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
                    warningLayout.setVisibility(View.VISIBLE);
                    TextView warningText =
findViewById(R.id.warningText);
                    String warningMessage =
getString(R.string.warning_unvalid_address_true, selectedToken);
                    warningText.setText(warningMessage);
                    ImageView warningIcon =
findViewById(R.id.warningIcon);
                    int iconResId =
getWarningIconResource(selectedToken);
                    warningIcon.setImageResource(iconResId);
                }
            }
            @Override
            public void onError(String errorMessage) {
                Toast.makeText(AssetSelectionSet.this,
"Ошибка сети", Toast.LENGTH_SHORT).show();
                VibrationHelper.triggerVibration(AssetSelectionSet.this);
            }
        });

```

```

        break;
    case "TON": {
        checkAddressExistence(selectedToken,
recipientAddress, new AddressCheckCallback() {
            @Override
            public void onResult(boolean exists) {
                if (exists) {
                    Intent intent = new
Intent(AssetSelectionSet.this, TransactionAmount.class);
                    intent.putExtra("selected_token",
selectedToken);
                    intent.putExtra("SCAN_RESULT",
recipientAddress);
                    startActivity(intent);
                    LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
                    warningLayout.setVisibility(View.GONE);
                } else {
                    VibrationHelper.triggerVibration(AssetSelectionSet.this);
                    LinearLayout warningLayout =
findViewById(R.id.addressWarningLayout);
                    warningLayout.setVisibility(View.VISIBLE);
                    TextView warningText =
findViewById(R.id.warningText);
                    String warningMessage =
getString(R.string.warning_unvalid_address, selectedToken);
                    warningText.setText(warningMessage);
                    ImageView warningIcon =
findViewById(R.id.warningIcon);
                    int iconResId =
getWarningIconResource(selectedToken);
                    warningIcon.setImageResource(iconResId);
                }
            }
        })

@Override
public void onError(String errorMessage) {
    Toast.makeText(AssetSelectionSet.this, "Ошибка сети",
Toast.LENGTH_SHORT).show();

    VibrationHelper.triggerVibration(AssetSelectionSet.this);
}
    });
    break;
}
default:
    // Если другой токен, пока пропускаем
    Intent intent = new Intent(AssetSelectionSet.this,
TransactionAmount.class);
    intent.putExtra("selected_token", selectedToken);
    intent.putExtra("SCAN_RESULT", recipientAddress);
    startActivity(intent);
}

```

```

        break;
    }
}

public void ClickCancel(View view) {
    PressEffectHelper.apply(this, view, "sss");
    finish();
}

private void resetRadioButtonTints(RadioButton... buttons) {
    for (RadioButton btn : buttons) {
        btn.setButtonTintList(ContextCompat.getColorStateList(this,
            R.color.SystemColor)); // цвет по умолчанию
    }
    solAddressPreviouslyWarned = false;
    ethAddressPreviouslyWarned = false;
}

private void selectTokenByAddress(String address, RadioButton
    token_btc, RadioButton token_eth, RadioButton token_sol, RadioButton
    token_usdt, RadioButton token_ton) {
    if (address.matches("^(1|3)[a-km-zA-HJ-NP-Z1-9]{25,34}$")) {
        // BTC Legacy или P2SH
        token_btc.setChecked(true);

        token_btc.setButtonTintList(ContextCompat.getColorStateList(this,
            R.color.BTCColor));
    } else if (address.matches("^(bc1|BC1)[a-z0-9]{39,59}$" ||
        address.matches("^(tb1|TB1)[a-z0-9]{39,59}$")) {
        // BTC Bech32 (mainnet bc1 или testnet tb1)
        token_btc.setChecked(true);
        token_btc.setButtonTintList(ContextCompat.getColorStateList(this,
            R.color.BTCColor));
    } else if (address.matches("[1-9A-HJ-NP-Za-km-z]{32,44}")) {
        // SOL адрес имеет длину от 32 до 44 символов и состоит
        только из букв и цифр
        token_sol.setChecked(true);
        token_sol.setButtonTintList(ContextCompat.getColorStateList(this,
            R.color.SOLColor)); // Устанавливаем кастомный цвет для SOL
    } else if (address.matches("^0x[a-fA-F0-9]{40}$")) {
        // ETH адреса начинаются с "0x" и содержат 40
        шестнадцатеричных символов
        token_eth.setChecked(true);
        token_eth.setButtonTintList(ContextCompat.getColorStateList(this,
            R.color.ETHColor)); // Устанавливаем кастомный цвет для ETH
    } else if (address.matches("^(EQ|Ef|kQ|Uf|UQA|0:)[0-9A-Za-z_-
        ]{30,}$")) {
        // TON адрес — начинаются с EQ, Ef, kQ, Uf, UQA или 0:
        token_ton.setChecked(true);
        token_ton.setButtonTintList(ContextCompat.getColorStateList(this,
            R.color.TONColor));
    } else {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		98

```

        // Если адрес не распознан, не активируем ни один
        радиобаттон и сбрасываем цвета
        token_btc.setChecked(false);
        token_eth.setChecked(false);
        token_sol.setChecked(false);
        token_usdt.setChecked(false);
        token_ton.setChecked(false);
        resetRadioButtonTints(token_btc, token_eth, token_usdt,
        token_ton, token_sol); // Сбрасываем цвета
    }
}
private void checkAddressExistence(String token, String address,
AddressCheckCallback callback) {
    switch (token) {
        case "BTC": {
            String url;
            // Проверяем testnet-префиксы: tb1 (bech32), m или n
            (legacy/P2SH)
            if (address.startsWith("tb1") ||
            address.startsWith("m") || address.startsWith("n")) {
                url =
                "https://blockstream.info/testnet/api/address/" + address;
            } else {
                url = "https://blockchain.info/rawaddr/" +
                address;
            }
            Log.d("BTC_CHECK", "Проверка адреса: " + address);
            Log.d("BTC_CHECK", "URL запроса: " + url);
            OkHttpClient client = new OkHttpClient();
            Request request = new
            Request.Builder().url(url).build();
            new Thread(() -> {
                try (Response response =
                client.newCall(request).execute()) {
                    Log.d("BTC_CHECK", "Код ответа: " +
                    response.code());
                    if (response.isSuccessful()) {
                        // Для testnet API возвращает 200 даже
                        если адрес без транзакций
                        runOnUiThread(() ->
                        callback.onResult(true));
                    } else {
                        runOnUiThread(() ->
                        callback.onResult(false));
                    }
                } catch (IOException e) {
                    Log.e("BTC_CHECK", "Сетевая ошибка: " +
                    e.getMessage(), e);
                    runOnUiThread(() ->
                    callback.onError(e.getMessage()));
                }
            })
        }
    }
}

```

```

        }).start();
        break;
    }
    case "ETH": {
        String apiKey =
"57K3YZ7ZFT5K6RQH24D7YA8TW7RIVNYVZ3";
        String url =
"https://api.etherscan.io/api?module=account&action=txlist&address="
+ address + "&startblock=0&endblock=99999999&sort=asc&apikey=" +
apiKey;

        Log.d("ETH_CHECK", "Проверка адреса: " + address);
        Log.d("ETH_CHECK", "URL запроса: " + url);
        OkHttpClient client = new OkHttpClient();
        Request request = new
Request.Builder().url(url).build();
        new Thread(() -> {
            try (Response response =
client.newCall(request).execute()) {
                Log.d("ETH_CHECK", "Код ответа: " +
response.code());

                if (response.isSuccessful() &&
response.body() != null) {
                    String body = response.body().string();
                    Log.d("ETH_CHECK", "Ответ: " + body);

                    JSONObject json = new JSONObject(body);
                    String status =
json.getString("status");

                    boolean hasTransactions =
status.equals("1");

                    runOnUiThread(() ->
callback.onResult(hasTransactions));
                } else {
                    runOnUiThread(() ->
callback.onResult(false));
                }
            } catch (IOException | JSONException e) {
                Log.e("ETH_CHECK", "Ошибка: " +
e.getMessage(), e);
                runOnUiThread(() ->
callback.onError(e.getMessage()));
            }
        }).start();
        break;
    }
    case "USDT": {
        String apiKey =
"57K3YZ7ZFT5K6RQH24D7YA8TW7RIVNYVZ3";
        String url =
"https://api.etherscan.io/api?module=account&action=tokenx"

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		100

```

        + "&address=" + address
        +
"&contractaddress=0xdAC17F958D2ee523a2206206994597C13D831ec7"
        +
"&startblock=0&endblock=99999999&sort=asc&apikey=" + apiKey;
        Log.d("USDT_CHECK", "Проверка адреса: " + address);
        Log.d("USDT_CHECK", "URL запроса: " + url);
        OkHttpClient client = new OkHttpClient();
        Request request = new
Request.Builder().url(url).build();
        new Thread(() -> {
            try (Response response =
client.newCall(request).execute()) {
                Log.d("USDT_CHECK", "Код ответа: " +
response.code());
                if (response.isSuccessful() &&
response.body() != null) {
                    String body = response.body().string();
                    Log.d("USDT_CHECK", "Ответ: " + body);
                    JSONObject json = new JSONObject(body);
                    String status =
json.getString("status");
                    boolean hasTransactions =
status.equals("1");
                    runOnUiThread(() ->
callback.onResult(hasTransactions));
                } else {
                    runOnUiThread(() ->
callback.onResult(false));
                }
            } catch (IOException | JSONException e) {
                Log.e("USDT_CHECK", "Ошибка: " +
e.getMessage(), e);
                runOnUiThread(() ->
callback.onError(e.getMessage()));
            }
        }).start();
        break;
    }
    case "SOL": {
        Log.d("SOL_CHECK", "Проверка адреса: " + address);

        OkHttpClient client = new OkHttpClient();

        String json = "{"
            + "\"jsonrpc\":\"2.0\", \"
            + "\"id\":1, \"
            + "\"method\":\"getSignaturesForAddress\", \"
            + "\"params\":[" + address + "\",
{\\\"limit\\\":1}]]"
            + "}";

```

```

        RequestBody body = RequestBody.create(
            json, MediaType.get("application/json"));
        Request request = new Request.Builder()
            .url("https://api.mainnet-beta.solana.com")
            .post(body)
            .build();
        new Thread(() -> {
            try (Response response =
client.newCall(request).execute()) {
                Log.d("SOL_CHECK", "Код ответа: " +
response.code());

                if (response.isSuccessful() &&
response.body() != null) {
                    String responseBody =
response.body().string();
                    Log.d("SOL_CHECK", "Ответ: " +
responseBody);
                    JSONObject jsonObject = new
JSONObject(responseBody);
                    JSONArray resultArray =
jsonObject.optJSONArray("result");
                    boolean hasTransactions = resultArray !=
null && resultArray.length() > 0;
                    runOnUiThread(() ->
callback.onResult(hasTransactions));
                } else {
                    runOnUiThread(() ->
callback.onResult(false));
                }
            } catch (IOException | JSONException e) {
                Log.e("SOL_CHECK", "Ошибка: " +
e.getMessage(), e);
                runOnUiThread(() ->
callback.onError(e.getMessage()));
            }
        }).start();
        break;
    }
    case "TON": {
        Log.d("TON_CHECK", "Проверка адреса: " + address);
        OkHttpClient client = new OkHttpClient();
        // Используем API TON от tonapi.io
        String url = "https://tonapi.io/v2/accounts/" +
address;

        Request request = new Request.Builder()
            .url(url)
            .get()
            .build();
        new Thread(() -> {

```

```

        try (Response response =
client.newCall(request).execute()) {
            Log.d("TON_CHECK", "Код ответа: " +
response.code());

            if (response.isSuccessful()) {
                // Адрес существует
                runOnUiThread() ->
callback.onResult(true));
            } else if (response.code() == 400) {
                // Адрес не существует
                runOnUiThread() ->
callback.onResult(false));
            } else {
                // Прочая ошибка
                runOnUiThread() ->
callback.onError("Ошибка: код " + response.code());
            }
        } catch (IOException e) {
            Log.e("TON_CHECK", "Ошибка сети: " +
e.getMessage(), e);
            runOnUiThread() ->
callback.onError(e.getMessage());
        }
    }).start();
    break;
}

}

}

}

interface AddressCheckCallback {
    void onResult(boolean exists);
    void onError(String errorMessage);
}

private int getWarningIconResource(String token) {
    switch (token) {
        case "BTC": return R.drawable.nobtc;
        case "ETH": return R.drawable.noeth_new;
        case "USDT": return R.drawable.noeth_new;
        case "TON": return R.drawable.nonew_ton;
        case "SOL": return R.drawable.nosol_new;
        default: return R.drawable.notfoundpng;
    }
}
}
}

```

Файл BalanceManager.java

```

package com.example.coldbitwallet;

import java.util.HashMap;
import java.util.Map;

public class BalanceManager {

```

```

        private static final BalanceManager instance = new
BalanceManager();
        private final Map<String, Double> balances = new HashMap<>();
        private final Map<String, String> addresses = new HashMap<>();
        private BalanceManager() {}
        public static BalanceManager getInstance() {
            return instance;
        }
        public void setBalance(String key, double value) {
            if (key.equalsIgnoreCase("usdt")) {
                value = value / 1_000_000.0;
            }
            balances.put(key.toLowerCase(), value);
        }
        public double getBalance(String token) {
            if (token != null &&
balances.containsKey(token.toLowerCase())) {
                return balances.get(token.toLowerCase());
            }
            return 0.0;
        }
        public void setAddress(String token, String address) {
            if (token != null && address != null) {
                addresses.put(token.toLowerCase(), address);
            }
        }
        public String getAddress(String token) {
            if (token != null &&
addresses.containsKey(token.toLowerCase())) {
                return addresses.get(token.toLowerCase());
            }
            return "";
        }
    }
}

```

Файл BiometricAuthManager.java

```

package com.example.coldbitwallet;
import android.app.Activity;
import android.content.Intent;
import android.os.Build;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.RequiresApi;
import androidx.biometric.BiometricManager;
import androidx.biometric.BiometricPrompt;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.FragmentActivity;
import java.util.concurrent.Executor;

public class BiometricAuthManager {
    private final FragmentActivity context;

```

```

        private final Class<?> successActivity;
        private final boolean returnResultOnly;
        public BiometricAuthManager(FragmentActivity context, Class<?>
successActivity) {
            this.context = context;
            this.successActivity = successActivity;
            this.returnResultOnly = false;
        }
        public BiometricAuthManager(FragmentActivity context) {
            this.context = context;
            this.successActivity = null;
            this.returnResultOnly = true;
        }
        @RequiresApi(api = Build.VERSION_CODES.M)
        public void authenticate() {
            BiometricManager biometricManager =
BiometricManager.from(context);
            int canAuthenticate =
biometricManager.canAuthenticate(BiometricManager.Authenticators.BIO
METRIC_WEAK);
            if (canAuthenticate != BiometricManager.BIOMETRIC_SUCCESS) {
                Toast.makeText(context,
R.string.scan_fingerprint_unavailable, Toast.LENGTH_SHORT).show();
                return;
            }
            Executor executor = ContextCompat.getMainExecutor(context);
            BiometricPrompt biometricPrompt = new BiometricPrompt(context,
executor, new BiometricPrompt.AuthenticationCallback() {
                @Override
                public void onAuthenticationSucceeded(@NonNull
BiometricPrompt.AuthenticationResult result) {
                    super.onAuthenticationSucceeded(result);

                    if (context instanceof EnterPinCode) {
                        ((EnterPinCode) context).fillPinCirclesOnSuccess();
                    } else if (context instanceof ConfirmPinCode) {
                        ((ConfirmPinCode)
context).fillPinCirclesOnSuccess();
                    }
                    context.runOnUiThread(() -> {
                        if (returnResultOnly) {
                            Intent resultIntent = new Intent();
                            if (context instanceof ConfirmPinCode) {
                                ConfirmPinCode confirmPinCode =
(ConfirmPinCode) context;
                                resultIntent.putExtra("entered_pin", confirmPinCode.getSavedPin());
                                // Метод для получения сохранённого пина
                                resultIntent.putExtra("reason",
confirmPinCode.getReason()); // Метод для получения причины
                            }
                        }
                    });
                }
            });
        }
    }
}

```

```

        context.setResult(Activity.RESULT_OK,
resultIntent);
        context.finish();
    } else {
        navigateToSuccessActivity();
    }
});
}
@Override
    public void onAuthenticationFailed() {
        super.onAuthenticationFailed();

        if (context instanceof EnterPinCode) {
            ((EnterPinCode)
context).showAuthenticationError();
        } else if (context instanceof ConfirmPinCode) {
            ((ConfirmPinCode)
context).showAuthenticationError();
        }
    }
});
    BiometricPrompt.PromptInfo promptInfo = new
BiometricPrompt.PromptInfo.Builder()
        .setTitle(context.getString(R.string.enter_with_finger_message))
        .setNegativeButton(context.getString(R.string.cancel_text))

        .setAllowedAuthenticators(BiometricManager.Authenticators.BIOMETRIC_
WEAK)

        .build();
    biometricPrompt.authenticate(promptInfo);
}
    private void navigateToSuccessActivity() {
        context.runOnUiThread() -> {
            Intent intent = new Intent(context, successActivity);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(intent);
            context.finish();
        });
    }
}

```

Файл BitcoinManager.kt

```

package com.example.coldbitwallet.wallet.bitcoin
import android.util.Log
import
com.example.coldbitwallet.wallet.HDWalletManager.HDWalletMaster
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody.Companion.toRequestBody
import org.bitcoinj.core.*

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		106

```

import org.bitcoinj.crypto.*
import org.bitcoinj.params.TestNet3Params
import org.bitcoinj.script.ScriptBuilder
import org.bitcoinj.wallet.DeterministicKeyChain
import org.bitcoinj.wallet.DeterministicSeed
import org.json.JSONArray
import org.json.JSONObject
import java.math.BigDecimal
import java.util.concurrent.TimeUnit
object BitcoinManager {

    private val params = TestNet3Params.get()
    private val client = OkHttpClient.Builder()
        .connectTimeout(15, TimeUnit.SECONDS)
        .readTimeout(15, TimeUnit.SECONDS)
        .build()
    private const val DUST = 546L
    data class Utxo(
        val txid: String,
        val vout: Int,
        val value: Long,
        val script: String
    )
    // KEY
    private fun getKey(
        mnemonic: List<String>,
        passphrase: String,
        change: Boolean,
        index: Int
    ): DeterministicKey {
        val seed = DeterministicSeed(mnemonic, null, passphrase, 0L)
        val keyChain =
            DeterministicKeyChain.builder().seed(seed).build()

        val path = listOf(
            ChildNumber(84, true),    // 84'
            ChildNumber(1, true),    // 1'
            ChildNumber(0, true),    // 0'
            ChildNumber(if (change) 1 else 0, false),
            ChildNumber(index, false)
        )
        return keyChain.getKeyByPath(HDPath(path), true)
    }
    fun getAddress(
        mnemonic: List<String>,
        passphrase: String = "",
        index: Int = 0
    ): String {
        val key = getKey(mnemonic, passphrase, false, index)
        return SegwitAddress.fromKey(params, key).toString()
    }
}

```

```

        private fun request(url: String): String {
            val res =
client.newCall(Request.Builder().url(url).build()).execute()
            if (!res.isSuccessful) throw Exception("HTTP ${res.code}")
            return res.body?.string() ?: ""
        }
        fun getUtxos(address: String): List<Utxo> {
            val url =
"https://blockstream.info/testnet/api/address/$address/utxo"
            val response = request(url)
            val json = JSONArray(response)
            return (0 until json.length()).map { i ->
                val o = json.getJSONObject(i)
                Utxo(
                    o.getString("txid"),
                    o.getInt("vout"),
                    o.getLong("value"),
                    "" // скрипт не нужен, он вычисляется из ключа
                )
            }
        }
        fun getFeeRate(): Double {
            return try {
                val json =
JSONObject(request("https://blockstream.info/testnet/api/fee-
estimates"))
                json.optDouble("1", 5.0)
            } catch (e: Exception) {
                5.0
            }
        }
        @OptIn(ExperimentalStdlibApi::class)
        fun sendBitcoin(
            mnemonic: List<String>,
            toAddress: String,
            amount: Long,
            passphrase: String = "",
            index: Int = 0
        ): String {
            val key = getKey(mnemonic, passphrase, false, index)
            val fromAddress = getAddress(mnemonic, passphrase, index)
            Log.d("BitcoinManager", "From address: $fromAddress")
            Log.d("BitcoinManager", "To address: $toAddress")
            Log.d("BitcoinManager", "Amount satoshis: $amount")
            val utxos = getUtxos(fromAddress)
            if (utxos.isEmpty()) throw Exception("Нет средств")
            val selected = mutableListOf<Utxo>()
            var total = 0L
            for (u in utxos.sortedBy { it.value }) {
                selected.add(u)
                total += u.value
            }
        }
    }
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		108

```

        if (total >= amount) break
    }
    if (selected.isEmpty()) throw Exception("Недостаточно
средств")

    Log.d("BitcoinManager", "Selected ${selected.size} UTXOs,
total $total satoshis")

    val tx = Transaction(params)

    for (u in selected) {
        val outPoint = TransactionOutPoint(
            params,
            u.vout.toLong(),
            Sha256Hash.wrap(u.txid)
        )
        tx.addInput(TransactionInput(params, tx, ByteArray(0),
outPoint))
    }
    val feeRate = getFeeRate()
    val vBytes = tx.inputs.size * 68 + 2 * 31 + 10
    val fee = (feeRate * vBytes).toLong()
    Log.d("BitcoinManager", "Fee: $fee satoshis, rate: $feeRate
sat/vB, vBytes: $vBytes")

    val to = Address.fromString(params, toAddress)
    tx.addOutput(Coin.valueOf(amount), to)

    var change = total - amount - fee
    if (change >= DUST) {
        val changeKey = getKey(mnemonic, passphrase, true, 0)
        val changeAddr = SegwitAddress.fromKey(params,
changeKey)
        tx.addOutput(Coin.valueOf(change), changeAddr)
        Log.d("BitcoinManager", "Change: $change satoshis to $changeAddr")
    }
    val address = SegwitAddress.fromKey(params, key)
    val scriptCode = ScriptBuilder.createP2PKHOutputScript(key)
    Log.d("BitcoinManager", "ScriptPubKey: ${scriptCode}")
    tx.inputs.forEachIndexed { i, input ->
        val u = selected[i]
        val hash = tx.hashForWitnessSignature(
            i,
            scriptCode,
            Coin.valueOf(u.value),
            Transaction.SigHash.ALL,
            false
        )
        val sig = key.sign(hash)
        val txSig = TransactionSignature(sig,
Transaction.SigHash.ALL, false)

```

```

        Log.d("BitcoinManager", "Input $i: hash=$hash,
sig=${txSig.encodeToBitcoin().joinToString("") { "%02x".format(it)
}}")

        input.scriptSig = ScriptBuilder.createEmpty()
        input.setWitness(TransactionWitness(2).apply {
            setPush(0, txSig.encodeToBitcoin())
            setPush(1, key.pubKey)
        })
    }
    tx.verify()
    val hex = tx.bitcoinSerialize().joinToString("") {
"%02x".format(it) }
    Log.d("BitcoinManager", "Raw tx hex: $hex")
    val res = client.newCall(
        Request.Builder()
            .url("https://mempool.space/testnet/api/tx")
            .post(hex.toByteArray().toRequestBody())
            .build()
    ).execute()
    val body = res.body?.string() ?: throw Exception("Empty
response")
    if (!res.isSuccessful) {
        throw Exception("Broadcast error: $body")
    }
    return body.trim('')
}
@JvmStatic
fun getAddress(
    master: HDWalletMaster,
    index: Int = 0,
    change: Boolean = false
): String {
    val path = listOf(
        ChildNumber(84, true),
        ChildNumber(1, true),
        ChildNumber(0, true),
        ChildNumber(if (change) 1 else 0, false),
        ChildNumber(index, false)
    )
    // Пошаговая деривация
    var key = master.masterKey
    for (child in path) {
        key = HDKeyDerivation.deriveChildKey(key, child)
    }

    return SegwitAddress.fromKey(params, key).toString()
}
fun getBtcBalance(address: String): BigDecimal {
    return try {
        val url =
"https://blockstream.info/testnet/api/address/$address"

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		110

```

val response = request(url)
    val json = JSONObject(response)
    val chainStats = json.getJSONObject("chain_stats")
    val funded = chainStats.getLong("funded_txo_sum")
    val spent = chainStats.getLong("spent_txo_sum")
    val satoshis = funded - spent
    Log.d("BTC_BALANCE", "funded=$funded spent=$spent
sat=$satoshis")
    BigDecimal(satoshis)
        .divide(BigDecimal(100_000_000))
    } catch (e: Exception) {
        Log.e("BTC_BALANCE", "error", e)
        BigDecimal.ZERO
    }
}
}

```

Файл BitcoinTransactionChecker.java

```

package com.example.coldbitwallet;

import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;
import org.json.JSONArray;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class BitcoinTransactionChecker {
    private static final String TAG = "BTC_CHECKER";
    public interface Callback {
        void onFinish();
    }

    public static void checkIncoming(Context context, String
address, Callback callback) {
        new Thread(() -> {
            try {
                String urlStr =
"https://blockstream.info/testnet/api/address/" + address + "/txs";
                URL url = new URL(urlStr);
                HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
                conn.setRequestMethod("GET");
                BufferedReader reader = new BufferedReader(
                    new InputStreamReader(conn.getInputStream())
                );

                StringBuilder result = new StringBuilder();
                String line;
            } catch (Exception e) {
                Log.e(TAG, "Error: " + e.getMessage());
            }
        }).start();
    }
}

```

```

        while ((line = reader.readLine()) != null) {
            result.append(line);
        }
reader.close();
JSONArray txs = new JSONArray(result.toString());
for (int i = 0; i < txs.length(); i++) {
    JSONObject tx = txs.getJSONObject(i);
    String txid = tx.getString("txid");
    JSONArray vouts = tx.getJSONArray("vout");
    for (int j = 0; j < vouts.length(); j++) {
        JSONObject out = vouts.getJSONObject(j);
        String outAddress =
out.optString("scriptpubkey_address", "");
        if (address.equals(outAddress)) {
            long value = out.getLong("value");
            double amount = value / 1e8;

            if (!isTxSaved(context, txid)) {
                Log.d(TAG, "NEW INCOMING TX: " +
txid);
TransactionHistoryManager.saveTransaction(
                                context,
                                "BTC",
                                amount,
                                address,
                                false
                                );
                saveTx(context, txid);
            }
        }
    }
}
} catch (Exception e) {
    Log.e(TAG, "Error checking BTC txs", e);
} finally {
    // Вызываем колбэк в главном потоке
    if (callback != null) {
        new
android.os.Handler(android.os.Looper.getMainLooper())
        .post(callback::onFinished);
    }
}
}).start();
}
private static boolean isTxSaved(Context context, String txid) {
    SharedPreferences prefs =
context.getSharedPreferences("btc_txs", Context.MODE_PRIVATE);
    return prefs.contains(txid);
}
private static void saveTx(Context context, String txid) {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		112

```

        SharedPreferences prefs =
context.getSharedPreferences("btc_txs", Context.MODE_PRIVATE);
        prefs.edit().putBoolean(txid, true).apply();
    }
}

```

Файл ButtonColorActivity.java

```

package com.example.coldbitwallet;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class ButtonColorActivity extends AppCompatActivity {
    private View selectedTextView = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_button_color);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);

        ThemeManager.applySavedBackground(this, R.id.main);

        TextColorManager.applySavedTextColor(this,
R.id.text_button_text);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
    return insets;
});
    int savedButtonsColor =
ButtonsColorManager.getSavedButtonColor(this);

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		113

```

        highlightSelectedButtonBlock(savedButtonsColor);
    }
    private int getOriginalTextColor(int viewId) {
        if (viewId == R.id.blueButtonBlock) {
            return ContextCompat.getColor(this, R.color.Blue);
        } else if (viewId == R.id.greenButtonBlock) {
            return ContextCompat.getColor(this, R.color.Green);
        } else if (viewId == R.id.blackButtonBlock) {
            return ContextCompat.getColor(this, R.color.Black);
        } else if (viewId == R.id.whiteButtonBlock) {
            return ContextCompat.getColor(this, R.color.TextColor);
        } else if (viewId == R.id.pinkButtonBlock) {
            return ContextCompat.getColor(this, R.color.Pink);
        } else if (viewId == R.id.purpleButtonBlock) {
            return ContextCompat.getColor(this, R.color.Purple);
        } else if (viewId == R.id.grayButtonBlock) {
            return ContextCompat.getColor(this, R.color.Gray);
        } else if (viewId == R.id.redButtonBlock) {
            return ContextCompat.getColor(this, R.color.Red);
        } else if (viewId == R.id.OrangeButtonBlock) {
            return ContextCompat.getColor(this, R.color.Orange);
        } else if (viewId == R.id.PaleButtonBlock) {
            return ContextCompat.getColor(this, R.color.PaleGold);
        } else if (viewId == R.id.yellowButtonBlock) {
            return ContextCompat.getColor(this, R.color.Yellow);
        } else if (viewId == R.id.brownButtonBlock) {
            return ContextCompat.getColor(this, R.color.Brown);
        } else if (viewId == R.id.darkGrayButtonBlock) {
            return ContextCompat.getColor(this, R.color.darkGray);
        } else if (viewId == R.id.trueMainButtonBlock) {
            return ContextCompat.getColor(this,
R.color.MainBackground);
        } else if (viewId == R.id.AquamarineButtonBlock) {
            return ContextCompat.getColor(this, R.color.Aquamarine);
        } else {
            return ContextCompat.getColor(this, R.color.Blue);
        }
    }
    private void highlightSelectedButtonBlock(int savedTextColor) {
        int[] textColorBlockIds = {
            R.id.brownButtonBlock, R.id.blueButtonBlock,
R.id.redButtonBlock, R.id.greenButtonBlock,
            R.id.purpleButtonBlock, R.id.whiteButtonBlock,
R.id.grayButtonBlock, R.id.PaleButtonBlock,
            R.id.OrangeButtonBlock, R.id.blackButtonBlock,
R.id.pinkButtonBlock, R.id.trueMainButtonBlock,
            R.id.yellowButtonBlock, R.id.darkGrayButtonBlock,
R.id.AquamarineButtonBlock
        };
        for (int id : textColorBlockIds) {
            View block = findViewById(id);

```

```

        if (block != null && getOriginalTextColor(id) ==
savedTextColor) {
            boolean isBlack = (id == R.id.blackButtonBlock);
            selectButtonColorBlock(block, isBlack);
            break;
        }
    }

    private void selectButtonColorBlock(View block, boolean
isBlackBlock) {
        if (selectedTextView != null) {

selectedTextView.setBackgroundColor(getOriginalTextColor(selectedTex
tView.getId())); // Сброс цвета
        }
        selectedTextView = block;
        GradientDrawable background = new GradientDrawable();
        background.setColor(getOriginalTextColor(block.getId()));
// Цвет блока
        background.setStroke(8, isBlackBlock ? Color.WHITE :
Color.BLACK); // Обводка
        block.setBackground(background);
    }

    private void applyButtonsColor(View view, int colorResId,
boolean isBlackTheme) {
        int normal = ContextCompat.getColor(this, colorResId);
        int savedThemeColor = ThemeManager.getSavedColor(this); //
Получаем текущий цвет темы
        int savedTextColor =
TextColorManager.getSavedTextColor(this); // Получаем текущий цвет
текста
        if (normal == savedThemeColor) {
            Toast.makeText(this,
getString(R.string.same_background_and_button_warning),
Toast.LENGTH_SHORT).show();
            VibrationHelper.triggerVibration(this);
            return; // Не сохраняем цвет текста
        }
        else if (normal == savedTextColor) {
            Toast.makeText(this,
getString(R.string.same_text_and_button_warning),
Toast.LENGTH_SHORT).show();
            VibrationHelper.triggerVibration(this);
            return; // Не сохраняем цвет текста
        }
    }

    ButtonsColorManager.saveButtonColor(this, normal);
    selectButtonColorBlock(view, isBlackTheme);
    returnToMainWindow(this);
}

    public void ClickCancel(View view) {
        PressEffectHelper.apply(this, view, "sss");
    }

```

```

        finish();
    }
    public void ClickBlue(View view) {
        applyButtonsColor(view, R.color.Blue, false);
    }
    public void ClickGreen(View view) {
        applyButtonsColor(view, R.color.Green, false);
    }
    public void ClickBlack(View view) {
        applyButtonsColor(view, R.color.Black, true);
    }
    public void ClickWhite(View view) {
        applyButtonsColor(view, R.color.TextColor, false);
    }
    public void ClickPink(View view) {
        applyButtonsColor(view, R.color.Pink, false);
    }
    public void ClickPurple(View view) {
        applyButtonsColor(view, R.color.Purple, false);
    }
}

    public void ClickGray(View view) {
        applyButtonsColor(view, R.color.Gray, false);
    }
    public void ClickRed(View view) {
        applyButtonsColor(view, R.color.Red, false);
    }
    public void ClickOrange(View view) {
        applyButtonsColor(view, R.color.Orange, false);
    }
    public void ClickPale(View view) {
        applyButtonsColor(view, R.color.PaleGold, false);
    }
    public void ClickYellow(View view) {
        applyButtonsColor(view, R.color.Yellow, false);
    }
    public void ClickBrown(View view) {
        applyButtonsColor(view, R.color.Brown, false);
    }
    public void ClickDarkGray(View view) {
        applyButtonsColor(view, R.color.darkGray, false);
    }
    public void ClickMain(View view) {
        applyButtonsColor(view, R.color.MainBackground, false);
    }
    public void ClickAquamarine(View view) {
        applyButtonsColor(view, R.color.Aquamarine, false);
    }
    public void returnToMainWindow(Context context) {
        Intent intent = new Intent(context, MainWindow.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);

```

```

        context.startActivity(intent);

        if (context instanceof Activity) {
            ((Activity) context).finish();
        }
    }
}

```

Файл ButtonsColorManager.java

```

package com.example.coldbitwallet;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.widget.Button;

public class ButtonsColorManager {

    private static final String PREFS_NAME = "button_color_prefs";
    private static final String KEY_BUTTON_COLOR =
"selected_button_color";
    // Применение сохраненного цвета ко всем переданным кнопкам
    public static void applySavedButtonColor(Activity activity,
int... buttonIds) {
        int color = getSavedButtonColor(activity);
        for (int id : buttonIds) {
            Button button = activity.findViewById(id);
            if (button != null) {
                button.setBackgroundColor(color);
            }
        }
    }
    // Сохранение выбранного цвета кнопки
    public static void saveButtonColor(Context context, int
colorResId) {
        SharedPreferences prefs =
context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
        prefs.edit().putInt(KEY_BUTTON_COLOR, colorResId).apply();
    }
    // Получение сохраненного цвета кнопок
    public static int getSavedButtonColor(Context context) {
        SharedPreferences prefs =
context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
        if (prefs.contains(KEY_BUTTON_COLOR)) {
            return prefs.getInt(KEY_BUTTON_COLOR, R.color.Blue); //
По умолчанию, если не найден
        } else {
            return R.color.Blue;
        }
    }
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		117

Файл ConfirmPinCode.java

```
package com.example.coldbitwallet;

import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.ColorStateList;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class ConfirmPinCode extends AppCompatActivity {

    private View[] pinCircles;
    private TextView hintEnterPin;
    private String pinCode = "";
    private String savedPin;
    private String resetPin;
    private String action;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_enter_pin_code);
        resetPin = getIntent().getStringExtra("reason");
        action = getIntent().getStringExtra("ACTION");
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        int savedColor = ThemeManager.getSavedColor(this); //
Получаем сохранённый цвет темы
        int savedColorText =
TextColorManager.getSavedTextColor(this); // Получаем сохранённый
цвет текста
        // Применение цвета текста
        int[] textElements = {
            R.id.HintEnterPin, R.id.ButtonFor0, R.id.ButtonFor1,
R.id.ButtonFor2, R.id.ButtonFor3,
            R.id.ButtonFor4, R.id.ButtonFor5, R.id.ButtonFor6,
R.id.ButtonFor7, R.id.ButtonFor8, R.id.ButtonFor9
        };
    }
```

					ПГУ 1.090301.29.001 ПЗ	Лист
						118
Изм.	Лист	№ докум.	Подпись	Дата		

```

        for (int id : textElements) {
            TextColorManager.applySavedTextColor(this, id);
        }
        // Применение заднего фона для кнопок
        int[] buttonIds = {
            R.id.ButtonFor0, R.id.ButtonFor1, R.id.ButtonFor2,
            R.id.ButtonFor3, R.id.ButtonFor4,
            R.id.ButtonFor5, R.id.ButtonFor6, R.id.ButtonFor7,
            R.id.ButtonFor8, R.id.ButtonFor9,
        };
        for (int id : buttonIds) {
            Button btn = findViewById(id);
            if (btn != null) {
                btn.setBackgroundTintList(ColorStateList.valueOf(savedColor));
            }
        }
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
            (v, insets) -> {
                Insets systemBars =
                    insets.getInsets(WindowInsetsCompat.Type.systemBars());
                v.setPadding(systemBars.left, systemBars.top,
                    systemBars.right, systemBars.bottom);
                return insets;
            });
        pinCircles = new View[]{
            findViewById(R.id.circle1),
            findViewById(R.id.circle2),
            findViewById(R.id.circle3),
            findViewById(R.id.circle4)
        };
        setPinCirclesColor(savedColorText);
        hintEnterPin = findViewById(R.id.HintEnterPin);
        SharedPreferences preferences =
            getSharedPreferences("MyPrefs", MODE_PRIVATE);
        savedPin = preferences.getString("user_pin", "");
        BiometricAuthManager authManager = new
            BiometricAuthManager(this);
        authManager.authenticate();
    }
    pinCircles = new View[]{
        findViewById(R.id.circle1),
        findViewById(R.id.circle2),
        findViewById(R.id.circle3),
        findViewById(R.id.circle4)
    };
    setPinCirclesColor(savedColorText);
    hintEnterPin = findViewById(R.id.HintEnterPin);
    SharedPreferences preferences =
        getSharedPreferences("MyPrefs", MODE_PRIVATE);
    savedPin = preferences.getString("user_pin", "");

```

```

        BiometricAuthManager authManager = new
BiometricAuthManager(this);
        authManager.authenticate();
    }
    public void ClickOn0(View view) { onDigitClick("0", view); }
    public void ClickOn1(View view) { onDigitClick("1", view); }
    public void ClickOn2(View view) { onDigitClick("2", view); }
    public void ClickOn3(View view) { onDigitClick("3", view); }
    public void ClickOn4(View view) { onDigitClick("4", view); }
    public void ClickOn5(View view) { onDigitClick("5", view); }
    public void ClickOn6(View view) { onDigitClick("6", view); }
    public void ClickOn7(View view) { onDigitClick("7", view); }
    public void ClickOn8(View view) { onDigitClick("8", view); }
    public void ClickOn9(View view) { onDigitClick("9", view); }
    private void onDigitClick(String digit, View view) {
        if (pinCode.length() < 4) {
            pinCode += digit;
            PressEffectHelper.apply(this, view, "but");
            updatePinCircles();
        }
    }
    public void ClickFingerprint(View view) {
        PressEffectHelper.apply(this, view, "imgb");

        BiometricAuthManager authManager = new
BiometricAuthManager(this);

        authManager.authenticate();
    }
    public void ClickClearLastSymbol(View view) {

        if (!pinCode.isEmpty()) {
            pinCode = pinCode.substring(0, pinCode.length() - 1);
            updatePinCircles();
        }

        PressEffectHelper.apply(this, view, "imgb");
    }
    private void setPinCirclesColor(int color) {
        for (View circle : pinCircles) {
            // Получаем текущий Drawable
            GradientDrawable drawable = (GradientDrawable)
circle.getBackground();
            drawable.setColor(color); // Устанавливаем новый цвет
        }
    }
    private void updatePinCircles() {
        // Обновляем кружки с пин-кодом
        for (int i = 0; i < pinCircles.length; i++) {
            if (i < pinCode.length()) {

pinCircles[i].setBackgroundResource(R.drawable.pin_circle_filled);

```

```

        } else {

pinCircles[i].setBackgroundResource(R.drawable.pin_circle_empty);
        }
    }
    if (pinCode.length() == 4) {
        if (pinCode.equals(savedPin)) {
            Intent resultIntent = new Intent();
            if ("TOGGLE_HIDE_BALANCE".equals(action)) {
                resultIntent.putExtra("ACTION",
"TOGGLE_HIDE_BALANCE");
                setResult(RESULT_OK, resultIntent); // Передаем
Intent с данными
                finish();
            } else if ("change_pin".equals(resetPin)) {
                Intent intent = new Intent(ConfirmPinCode.this,
CreatePinCode.class);
                startActivity(intent);
                finish(); // Закрываем активность
            } else if ("EXIT".equals(action)) {
                resultIntent.putExtra("ACTION", "EXIT");
                setResult(RESULT_OK, resultIntent);
                finish();
            } else {
                Intent intent = new Intent(ConfirmPinCode.this,
MainWindow.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK
| Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
                finish();
            }
        } else {
            for (View circle : pinCircles) {
circle.setBackgroundResource(R.drawable.pin_circle_error);
            }
            VibrationHelper.triggerVibration(this);
            hintEnterPin.setText(getString(R.string.error_pin_code));
            pinCircles[0].postDelayed(() -> {
                pinCode = "";
                for (View circle : pinCircles) {

circle.setBackgroundResource(R.drawable.pin_circle_empty);
                }

            hintEnterPin.setText(getString(R.string.enter_pin_code));
            }, 200);
        }
    }
}

public void fillPinCirclesOnSuccess() {
    for (View circle : pinCircles) {

```

```

circle.setBackgroundResource(R.drawable.pin_circle_filled);
    }
    Intent resultIntent = new Intent(); // Добавьте это
    if ("TOGGLE_HIDE_BALANCE".equals(action)) {
        resultIntent.putExtra("ACTION", "TOGGLE_HIDE_BALANCE");
        setResult(RESULT_OK, resultIntent); // Передаем Intent с
данными
        finish();
    } else if ("change_pin".equals(resetPin)) {
        Intent intent = new Intent(ConfirmPinCode.this,
CreatePinCode.class);
        startActivity(intent);
        finish();
    } else if ("EXIT".equals(action)) {
        resultIntent.putExtra("ACTION", "EXIT");
        setResult(RESULT_OK, resultIntent);
        finish();
    } else {
        Intent intent = new Intent(ConfirmPinCode.this,
MainWindow.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
        finish();
    }
}

public void showAuthenticationError() {
    for (View circle : pinCircles) {

circle.setBackgroundResource(R.drawable.pin_circle_error);
    }
    VibrationHelper.triggerVibration(this);
    pinCircles[0].postDelayed(() -> {
        pinCode = "";
        for (View circle : pinCircles) {
circle.setBackgroundResource(R.drawable.pin_circle_empty);
        }
    }, 200);
}

    public String getSavedPin() {
        return savedPin;
    }

    public String getReason() {
        return action;
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }
}

```

Файл ConnectOldWalletActivity.java

```
package com.example.coldbitwallet;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.example.coldbitwallet.wallet.HDWalletManager;
import com.example.coldbitwallet.wallet.bitcoin.BitcoinManager;
import com.example.coldbitwallet.wallet.ethereum.EthereumManager;
import com.example.coldbitwallet.wallet.solana.SolanaManager;
import com.example.coldbitwallet.wallet.ton.TonManager;
import org.bitcoinj.crypto.MnemonicCode;
import java.util.ArrayList;
import java.util.List;

public class ConnectionOldWalletActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_connection_old_wallet);
        TextView hint =
findViewById(R.id.connection_old_wallet_hint);
hint.setHintTextColor(Color.LTGRAY);
// Применение сохраненного цвета для кнопок
Button nextButton = findViewById(R.id.CreatePinCode);
PressEffectHelper.apply(this, nextButton, "spec");
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        // Установка хинтов для TextView от wordNumber1 до
wordNumber24
        for (int i = 1; i <= 24; i++) {
            // Формируем имя ресурса для TextView, например
"wordNumber1"
            String textViewId = "wordNumber" + i;
```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		123

```

        int resID = getResources().getIdentifier(textViewId,
        "id", getPackageName());
        // Находим TextView по ID
        TextView wordNumberTextView = findViewById(resID);
        if (wordNumberTextView != null) {
            // Устанавливаем хинт для TextView
            wordNumberTextView.setHint((i) + ":");
            wordNumberTextView.setHintTextColor(Color.LTGRAY);
        }
    }
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
    (v, insets) -> {
        Insets systemBars =
        insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top,
        systemBars.right, systemBars.bottom);
        return insets;
    });
}
public void ClickConfirmSeedWords(View view) {
    List<String> mnemonicWords = new ArrayList<>();
    boolean allFilled = true;
    for (int i = 1; i <= 24; i++) {
        EditText wordInput =
        findViewById(getResources().getIdentifier("wordInput" + i, "id",
        getPackageName()));
        if (wordInput != null) {
            String word =
            wordInput.getText().toString().trim().toLowerCase();
            if (word.isEmpty()) {
                allFilled = false;
                break;
            }
            mnemonicWords.add(word);
        }
    }
    if (!allFilled) {
        Toast.makeText(this,
        getString(R.string.toast_error_enter_seed_phrase),
        Toast.LENGTH_SHORT).show();
        VibrationHelper.triggerVibration(this);
        return;
    }
    // Проверка checksum
    try {
        MnemonicCode.INSTANCE.check(mnemonicWords);
    } catch (Exception e) {
        Toast.makeText(this,
        getString(R.string.toast_error_enter_seed_phrase),
        Toast.LENGTH_SHORT).show();
        VibrationHelper.triggerVibration(this);
    }
}

```

```

        return;
    }
    // Перевод сид-фразы в seedBytes
    byte[] seedBytes =
HDWalletManager.mnemonicToSeed(mnemonicWords, "");
    // Создание master key
    HDWalletManager.HDWalletMaster master = new
HDWalletManager.HDWalletMaster(seedBytes);
    // Получение адресов
    String btcAddress = BitcoinManager.getAddress(master, 0,
false);
    String ethAddress = EthereumManager.getAddress(master, 0);
    String solAddress = SolanaManager.getAddress(seedBytes, 0);
    String tonAddress = TonManager.getAddress(seedBytes, true,
false);
    // Сохранение seed и адресов через HDWalletManager или
SharedPreferences
    HDWalletManager.saveSeed(this, seedBytes);
    SharedPreferences prefs = getSharedPreferences("MyPrefs",
Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = prefs.edit();
    editor.putString("bitcoin_address", btcAddress);
    editor.putString("ethereum_address", ethAddress);
    editor.putString("solana_address", solAddress);
    editor.putString("ton_address", tonAddress);
    editor.apply();

    // Переход на следующий экран
    startActivity(new Intent(this, CreatePinCode.class));
}
public void ClickCancel(View view) {
    PressEffectHelper.apply(this, view, "sss");
    finish();
}
}

```

Файл CreateNewWalletStage1.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		125

```

import androidx.core.view.WindowInsetsCompat;
import java.util.ArrayList;

public class CreateNewWalletStage1 extends AppCompatActivity {
    private ArrayList<String> seedWords;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_create_new_wallet_stage1);

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
            (v, insets) -> {
                Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
                v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
                return insets;
            });
        seedWords =
getIntent().getStringArrayListExtra("seedWords");
        CheckBox checkBox1 = findViewById(R.id.checkBox1);
        CheckBox checkBox2 = findViewById(R.id.checkBox2);
        CheckBox checkBox3 = findViewById(R.id.checkBox3);
        View rootLayout = findViewById(R.id.main);
        CompoundButton.OnCheckedChangeListener listener = (buttonView,
isChecked) -> {
            updateBackgroundColor(rootLayout, checkBox1, checkBox2,
checkBox3);
        };

        // Применение сохраненного цвета для кнопок
        Button nextButton =
findViewById(R.id.NextButtonCreateNewWalletStage2);
        PressEffectHelper.apply(this, nextButton, "spec");

        checkBox1.setOnCheckedChangeListener(listener);
        checkBox2.setOnCheckedChangeListener(listener);
        checkBox3.setOnCheckedChangeListener(listener);

        // Первоначальная установка цвета
        updateBackgroundColor(rootLayout, checkBox1, checkBox2,
checkBox3);
    }
    public void ClickCancel(View view) {
        PressEffectHelper.apply(this, view, "sss");
        finish();
    }
    public void ClickNextButtonCreateNewWalletStage2(View view) {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		126

```

        CheckBox checkBox1 = findViewById(R.id.checkBox1);
        CheckBox checkBox2 = findViewById(R.id.checkBox2);
        CheckBox checkBox3 = findViewById(R.id.checkBox3);
        if (checkBox1.isChecked() && checkBox2.isChecked() &&
checkBox3.isChecked()) {
            Intent intent = new Intent(this,
SeedPhraseActivity.class);
            intent.putStringArrayListExtra("seedWords", seedWords);
            startActivity(intent);
        } else {
            VibrationHelper.triggerVibration(this);
            // Показываем предупреждение пользователю
            Toast.makeText(this, R.string.toast_check_box,
Toast.LENGTH_SHORT).show();
        }
    }

    private void updateBackgroundColor(View rootLayout, CheckBox
cb1, CheckBox cb2, CheckBox cb3) {
        int count = 0;
        if (cb1.isChecked()) count++;
        if (cb2.isChecked()) count++;
        if (cb3.isChecked()) count++;
        int colorResId;
        switch (count) {
            case 1:
                colorResId = R.color.PinError;
                break;
            case 2:
                colorResId = R.color.Yellow;
                break;
            case 3:
                colorResId = R.color.Green;
                break;
            default:
                rootLayout.setBackgroundColor(getResources().getColor(R.color.MainBa
ckground));
                return;
        }
        int color = ContextCompat.getColor(this, colorResId);
        rootLayout.setBackgroundColor(color);
    }
}

```

Файл CreateOrConnectWallet.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.security.keystore.KeyGenParameterSpec;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		127

```

import android.security.keystore.KeyProperties;
import android.util.Base64;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.activity.OnBackPressedCallback;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.example.coldbitwallet.wallet.HDWalletManager;
import com.example.coldbitwallet.wallet.bitcoin.BitcoinManager;
import com.example.coldbitwallet.wallet.ethereum.EthereumManager;
import com.example.coldbitwallet.wallet.solana.SolanaManager;
import com.example.coldbitwallet.wallet.ton.TonManager;
import java.security.KeyPairGenerator;
import java.security.KeyStore;
import java.security.PublicKey;
import java.util.ArrayList;
import javax.crypto.Cipher;

public class CreateOrConnectWallet extends AppCompatActivity {
    private DoubleBackClickManager doubleBackPressHandler;
    private ArrayList<String> generatedSeedWords;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings", MODE_PRIVATE)
            .getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_create_or_connect_wallet);
        doubleBackPressHandler = new DoubleBackClickManager(this);
        getOnBackPressedDispatcher().addCallback(this,
            new OnBackPressedCallback(true) {
                @Override
                public void handleOnBackPressed() {

doubleBackPressHandler.onBackPressed(R.string.toast_exit_app);
                }
            });
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        int[] textViewIds = {
            R.id.ButtonNewWallet,
            R.id.ButtonOldWallet,
            R.id.IceBitText
        };
    }
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		128

```

        for (int id : textViewIds) {
            TextColorManager.applySavedTextColor(this, id);
        }
        Button newWalletButton =
findViewById(R.id.ButtonNewWallet);
        Button oldWalletButton = findViewById(R.id.ButtonOldWallet);
        PressEffectHelper.apply(this, newWalletButton, "spec");
        PressEffectHelper.apply(this, oldWalletButton, "spec");

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
        (v, insets) -> {
Insets systemBars = insets.getInsets(
            WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
                systemBars.right, systemBars.bottom);
            return insets;
        });

        SharedPreferences preferences =
getSharedPreferences("MyPrefs", MODE_PRIVATE);
        String savedPin = preferences.getString("user_pin", null);

        if (savedPin != null) {
            startActivity(new Intent(this, ConfirmPinCode.class));
            finish();
            return;
        }

        // Не даём упасть при балансе
loadBalancesSafe();
    }
    private void loadBalancesSafe() {
        try {
            WalletApiClient.fetchBalances(this, new
WalletApiClient.BalanceCallback() {

                @Override
                public void onSuccess() {
                    if (isFinishing() || isDestroyed()) return;
                    SharedPreferences prefs =
getSharedPreferences("MyPrefs",
MODE_PRIVATE);

                    double btc =
safeParse(prefs.getString("bitcoin_balance", "0"));
                    double eth =
safeParse(prefs.getString("eth_balance", "0"));
                    double sol =
safeParse(prefs.getString("sol_balance", "0"));
                    double ton =
safeParse(prefs.getString("ton_balance", "0"));

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		129

```

        double usdt =
safeParse(prefs.getString("usdt_balance", "0"));
        BalanceManager manager =
BalanceManager.getInstance();
        manager.setBalance("btc", btc);
        manager.setBalance("eth", eth);
        manager.setBalance("sol", sol);
        manager.setBalance("ton", ton);
        manager.setBalance("usdt", usdt);
        Log.d("Wallet", "Балансы обновлены");
    }
    @Override
    public void onError(String error) {
        Log.e("Wallet", "Ошибка балансов: " + error);
    }
});
} catch (Exception e) {
    Log.e("Wallet", "Crash prevented", e);
}
}
private double safeParse(String value) {
    try {
        if (value == null || value.trim().isEmpty()) return 0.0;
        return Double.parseDouble(value);
    } catch (Exception e) {
        return 0.0;
    }
}
public void ButtonNewWalletClicked(View view) {
    Toast.makeText(this,
        getString(R.string.toast_create_wallet),
        Toast.LENGTH_SHORT).show();try {
        generatedSeedWords =
            (ArrayList<String>)
HDWalletManager.generateMnemonic();
        if (generatedSeedWords == null ||
generatedSeedWords.isEmpty()) {
            Toast.makeText(this,
                "Seed generation failed",
                Toast.LENGTH_SHORT).show();
            return;
        }
        String seed = String.join(" ", generatedSeedWords);

        byte[] aesKey = CryptoHelper.generateRandomAesKey();
        byte[] encryptedSeed =
            CryptoHelper.encryptWithAes(seed, aesKey);
        KeyStore keyStore =
            KeyStore.getInstance("AndroidKeyStore");
        keyStore.load(null);
        if (!keyStore.containsAlias("seed_rsa_key")) {

```

```

        KeyPairGenerator keyPairGenerator =
            KeyPairGenerator.getInstance(
                "RSA", "AndroidKeyStore");
        keyPairGenerator.initialize(
            new KeyGenParameterSpec.Builder(
                "seed_rsa_key",
                KeyProperties.PURPOSE_ENCRYPT |
KeyProperties.PURPOSE_DECRYPT).setDigests(KeyProperties.DIGEST_SHA25
6,
KeyProperties.DIGEST_SHA512)
                .setEncryptionPaddings(
KeyProperties.ENCRYPTION_PADDING_RSA_PKCS1)
                .build());
        keyPairGenerator.generateKeyPair();
    }
    PublicKey publicKey =
        keyStore.getCertificate("seed_rsa_key")
            .getPublicKey();
    Cipher rsaCipher =
        Cipher.getInstance("RSA/ECB/PKCS1Padding");
    rsaCipher.init(Cipher.ENCRYPT_MODE, publicKey);
    byte[] encryptedAesKey =
        rsaCipher.doFinal(aesKey);
    SharedPreferences prefs =
        getSharedPreferences("MyPrefs", MODE_PRIVATE);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.FROYO)
    {
        prefs.edit()
            .putString("encrypted_seed",
Base64.encodeToString(encryptedSeed,
                        Base64.DEFAULT))
            .putString("encrypted_aes_key",
Base64.encodeToString(encryptedAesKey,
                        Base64.DEFAULT))
            .apply();
    }
    byte[] seedBytes =
        HDWalletManager.mnemonicToSeed(
            generatedSeedWords, "");
    HDWalletManager.HDWalletMaster master =
        new HDWalletManager.HDWalletMaster(seedBytes);
    String btcAddress =
        BitcoinManager.getAddress(master, 0, false);
    String ethAddress =
        EthereumManager.getAddress(master, 0);
    String solAddress =
        SolanaManager.getAddress(seedBytes, 0);
    String tonAddress =
        TonManager.getAddress(seedBytes, true, false); //
    USDT = ETH адрес
        String usdtAddress = ethAddress;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						131
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.GINGERBREAD) {
            prefs.edit()
                .putString("bitcoin_address", btcAddress)
                .putString("ethereum_address", ethAddress)
                .putString("solana_address", solAddress)
                .putString("ton_address", tonAddress)
                .putString("usdt_address", usdtAddress)
                .apply();
        }
        Intent intent =
            new Intent(this,
                CreateNewWalletStage1.class);
        intent.putStringArrayListExtra(
            "seedWords",
            generatedSeedWords);
        startActivity(intent);
    } catch (Exception e) {
        Log.e("WALLET", "Create wallet crash", e);

        Toast.makeText(this,
            "Wallet creation failed",
            Toast.LENGTH_SHORT).show();
    }
}

public void ButtonOldWalletClicked(View view) {
    startActivity(new Intent(this,
        ConnectionOldWalletActivity.class));
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (doubleBackPressedHandler != null) {
        doubleBackPressedHandler.cancel();
    }
}
}

```

Файл CreatePinCode.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.content.res.ColorStateList;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.os.Handler;
import android.os.Looper;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						132
Изм.	Лист	№ докум.	Подпись	Дата		

```

import android.widget.TextView;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import android.content.SharedPreferences;
import android.widget.Toast;

public class CreatePinCode extends AppCompatActivity {
    private View[] pinCircles;
    private TextView hintNewPin;
    private String pinCode = "";
    private String firstPinCode = "";private boolean isConfirming =
false;
    private boolean isWaitingForInternet = false;
    private Handler internetCheckHandler = new
Handler(Looper.getMainLooper());
private Runnable internetCheckRunnable;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_create_pin_code);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        // Применение цвета текста
        int[] textElements = {
            R.id.HintEnterPin, R.id.ButtonFor0, R.id.ButtonFor1,
R.id.ButtonFor2, R.id.ButtonFor3,
            R.id.ButtonFor4, R.id.ButtonFor5, R.id.ButtonFor6,
R.id.ButtonFor7, R.id.ButtonFor8, R.id.ButtonFor9,
            R.id.internetWarningText
        };
        for (int id : textElements) {
            TextColorManager.applySavedTextColor(this, id);
        }
        hintNewPin = findViewById(R.id.HintEnterPin);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}

```

```

    });
    int savedColor = ThemeManager.getSavedColor(this); //
Получаем сохранённый цвет темы
    int savedColorText =
TextColorManager.getSavedTextColor(this); // Получаем сохранённый
цвет текста
    int[] buttonIds = {
        R.id.ButtonFor0, R.id.ButtonFor1, R.id.ButtonFor2,
R.id.ButtonFor3, R.id.ButtonFor4,
        R.id.ButtonFor5, R.id.ButtonFor6, R.id.ButtonFor7,
R.id.ButtonFor8, R.id.ButtonFor9,
    };
    for (int id : buttonIds) {
        Button btn = findViewById(id);
        if (btn != null) {

btn.setBackgroundTintList (ColorStateList.valueOf (savedColor));
        }
    }
    pinCircles = new View[]{
        findViewById(R.id.circle1),
        findViewById(R.id.circle2),
        findViewById(R.id.circle3),
        findViewById(R.id.circle4)
    };
    setPinCirclesColor(savedColorText);
}
public void ClickOn0(View view) { onDigitClick("0", view); }
public void ClickOn1(View view) { onDigitClick("1", view); }
public void ClickOn2(View view) { onDigitClick("2", view); }
public void ClickOn3(View view) { onDigitClick("3", view); }
public void ClickOn4(View view) { onDigitClick("4", view); }
public void ClickOn5(View view) { onDigitClick("5", view); }
public void ClickOn6(View view) { onDigitClick("6", view); }
public void ClickOn7(View view) { onDigitClick("7", view); }
public void ClickOn8(View view) { onDigitClick("8", view); }
public void ClickOn9(View view) { onDigitClick("9", view); }
private void onDigitClick(String digit, View view) {
    if (pinCode.length() < 4) {
        pinCode += digit;
        PressEffectHelper.apply(this, view, "but");
        updatePinCircles();
    }
}
public void ClickClearLastSymbol(View view) {
    if (pinCode.length() > 0) {
        pinCode = pinCode.substring(0, pinCode.length() - 1);
        updatePinCircles();
    }
    PressEffectHelper.apply(this, view, "imgb");
}
}

```

```

private void setPinCirclesColor(int color) {
    for (View circle : pinCircles) {
        // Получаем текущий Drawable
        GradientDrawable drawable = (GradientDrawable)
circle.getBackground();
        drawable.setColor(color); // Устанавливаем новый цвет
    }
}
private void updatePinCircles() {
    for (int i = 0; i < pinCircles.length; i++) {
        if (i < pinCode.length()) {

pinCircles[i].setBackgroundResource(R.drawable.pin_circle_filled);
        } else {

pinCircles[i].setBackgroundResource(R.drawable.pin_circle_empty);
        }
    }
    if (pinCode.length() == 4) {
        if (!isConfirming) {
            // Сохраняем первый введенный пин
            firstPinCode = pinCode;
            // Подготовка к этапу подтверждения
            pinCode = "";
            isConfirming = true;
            // Сброс кружков
            for (View circle : pinCircles) {
circle.setBackgroundResource(R.drawable.pin_circle_empty);
            }
            // Обновление подсказки

hintNewPin.setText(getString(R.string.create_pin_code2));
        } else {
            if (pinCode.equals(firstPinCode)) {
                waitForInternetAndProceed(); // вынесено в метод
            } else {
                // Ошибка подтверждения
                for (View circle : pinCircles) {

circle.setBackgroundResource(R.drawable.pin_circle_error);
                }
                VibrationHelper.triggerVibration(this);
                // Через 200 мс сбросить
                pinCircles[0].postDelayed(() -> {
                    pinCode = "";
                    firstPinCode = "";
                    isConfirming = false;

hintNewPin.setText(getString(R.string.create_pin_code1));
                    for (View circle : pinCircles) {
circle.setBackgroundResource(R.drawable.pin_circle_empty);

```

```

    }
}, 200);
    }
}

private void proceedToMainWindow() {
    SharedPreferences preferences =
getSharedPreferences("MyPrefs", MODE_PRIVATE);
    SharedPreferences.Editor editor = preferences.edit();
    editor.putString("user_pin", pinCode);
    editor.apply();
    Intent intent = new Intent(CreatePinCode.this,
MainWindow.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(intent);
    finish();
}

private void waitForInternetAndProceed() {
    if (isWaitingForInternet) return;
    isWaitingForInternet = true;

    TextView internetWarningText =
findViewById(R.id.internetWarningText);
    Animation pulseAnimation =
AnimationUtils.loadAnimation(this, R.anim.pulse); // подключаем
анимацию
    if (!PriceManager.isInternetAvailable(this)) {
        internetWarningText.setVisibility(View.VISIBLE);
        Toast.makeText(this,
getString(R.string.check_internet_connection),
Toast.LENGTH_SHORT).show();
        // Запустить анимацию пульсации для кружочков
startCircleAnimation(pulseAnimation);
        internetCheckRunnable = new Runnable() {
            @Override
            public void run() {
                if
(PriceManager.isInternetAvailable(CreatePinCode.this)) {
internetWarningText.setVisibility(View.GONE);
stopCircleAnimation(); // Остановить
анимацию

proceedToMainWindow();
} else {
internetCheckHandler.postDelayed(this,
2000);
}
}
};
internetCheckHandler.postDelayed(internetCheckRunnable, 2000);

```

```

        } else {
            internetWarningText.setVisibility(View.GONE);
            proceedToMainWindow();
        }
    }
    private void startCircleAnimation(Animation animation) {
for (View circle : pinCircles) {
        circle.startAnimation(animation);
    }
}
    private void stopCircleAnimation() {
        for (View circle : pinCircles) {
            circle.clearAnimation();
        }
    }
}
}

```

Файл CryphoHelper.java

```

package com.example.coldbitwallet;

import java.nio.charset.StandardCharsets;
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class CryptoHelper {
    public static byte[] generateRandomAesKey() throws Exception {
        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
        keyGen.init(256); // 256-bit AES
        SecretKey secretKey = keyGen.generateKey();
        return secretKey.getEncoded();
    }

    public static byte[] encryptWithAes(String plainText, byte[]
key) throws Exception {
        // Генерация случайного инициализационного вектора (IV)
        byte[] iv = new byte[16]; // Для AES блок 128 бит (16 байт)
        new SecureRandom().nextBytes(iv);
        SecretKeySpec secretKeySpec = new SecretKeySpec(key, "AES");
        IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec,
ivParameterSpec);
        byte[] encrypted =
cipher.doFinal(plainText.getBytes(StandardCharsets.UTF_8));
        // Сохраняем IV вместе с зашифрованными данными (IV в
начале)
        byte[] encryptedWithIv = new byte[iv.length +
encrypted.length];

```

```

        System.arraycopy(iv, 0, encryptedWithIv, 0, iv.length);
        System.arraycopy(encrypted, 0, encryptedWithIv, iv.length,
encrypted.length);
        return encryptedWithIv;
    }
    // Метод для дешифрования (для примера)
    public static String decryptWithAes(byte[] encryptedData, byte[]
key) throws Exception {
        // Извлекаем IV из зашифрованных данных
        byte[] iv = new byte[16];
        System.arraycopy(encryptedData, 0, iv, 0, iv.length);
        // Извлекаем зашифрованные данные
        byte[] encrypted = new byte[encryptedData.length -
iv.length];
        System.arraycopy(encryptedData, iv.length, encrypted, 0,
encrypted.length);
        SecretKeySpec secretKeySpec = new SecretKeySpec(key, "AES");
        IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec,
ivParameterSpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return new String(decrypted, StandardCharsets.UTF_8);
    }
}

```

Файл DoubleBackClickManager.java

```

package com.example.coldbitwallet;
import android.app.Activity;
import android.os.Handler;
import android.widget.Toast;

public class DoubleBackClickManager {
    private final Activity activity;
    private final Handler handler = new Handler();
    private boolean doubleBackToExitPressedOnce = false;
    private final Runnable resetFlagRunnable = () ->
doubleBackToExitPressedOnce = false;
    public DoubleBackClickManager (Activity activity) {
        this.activity = activity;
    }
    public void onBackPressed(int toastMessageResId) {
        if (doubleBackToExitPressedOnce) {
            activity.finish();
            return;
        }
        doubleBackToExitPressedOnce = true;
        Toast.makeText(activity, toastMessageResId,
Toast.LENGTH_SHORT).show();
        handler.postDelayed(resetFlagRunnable, 2000);
    }
}

```

```

        public void cancel() {
            handler.removeCallbacks(resetFlagRunnable);
        }
    }
}

```

Файл EnterPinCode.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.content.res.ColorStateList;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.os.Build;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import android.content.SharedPreferences;
import android.widget.Toast;

public class EnterPinCode extends AppCompatActivity {

    private View[] pinCircles;
    private TextView hintEnterPin;
    private String pinCode = "";
    private String savedPin;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_enter_pin_code);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        int savedColor = ThemeManager.getSavedColor(this); //
Получаем сохранённый цвет темы
        int savedColorText =
TextColorManager.getSavedTextColor(this); // Получаем сохранённый
цвет текста
        // Применение цвета текста
        int[] textElements = {
            R.id.hintEnterPin, R.id.ButtonFor0, R.id.ButtonFor1,
R.id.ButtonFor2, R.id.ButtonFor3,

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						139
Изм.	Лист	№ докум.	Подпись	Дата		

```

        R.id.ButtonFor4, R.id.ButtonFor5, R.id.ButtonFor6,
        R.id.ButtonFor7, R.id.ButtonFor8, R.id.ButtonFor9
    };
    for (int id : textElements) {
        TextColorManager.applySavedTextColor(this, id);
    }
    // Применение заднего фона для кнопок
    int[] buttonIds = {
        R.id.ButtonFor0, R.id.ButtonFor1, R.id.ButtonFor2,
        R.id.ButtonFor3, R.id.ButtonFor4,
        R.id.ButtonFor5, R.id.ButtonFor6, R.id.ButtonFor7,
        R.id.ButtonFor8, R.id.ButtonFor9,
    };
    for (int id : buttonIds) {
        Button btn = findViewById(id);
        if (btn != null) {

            btn.setBackgroundTintList(ColorStateList.valueOf(savedColor));
        }
    }

    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
        (v, insets) -> {
            Insets systemBars =
            insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
            systemBars.right, systemBars.bottom);
            return insets;
        });
    pinCircles = new View[]{
        findViewById(R.id.circle1),
        findViewById(R.id.circle2),
        findViewById(R.id.circle3),
        findViewById(R.id.circle4)
    };
    setPinCirclesColor(savedColorText);
    hintEnterPin = findViewById(R.id.HintEnterPin);
    SharedPreferences preferences =
    getSharedPreferences("MyPrefs", MODE_PRIVATE);
    savedPin = preferences.getString("user_pin", "");
    BiometricAuthManager authManager = new
    BiometricAuthManager(this, MainWindow.class);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        authManager.authenticate();
    }
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        authManager.authenticate();
    } else {
        Toast.makeText(this,
        getText(R.string.scan_fingerprint_unavailable),
        Toast.LENGTH_SHORT).show();
    }

```

```

    }
}
public void ClickOn0(View view) { onDigitClick("0", view); }
public void ClickOn1(View view) { onDigitClick("1", view); }
public void ClickOn2(View view) { onDigitClick("2", view); }
public void ClickOn3(View view) { onDigitClick("3", view); }
public void ClickOn4(View view) { onDigitClick("4", view); }
public void ClickOn5(View view) { onDigitClick("5", view); }
public void ClickOn6(View view) { onDigitClick("6", view); }
public void ClickOn7(View view) { onDigitClick("7", view); }
public void ClickOn8(View view) { onDigitClick("8", view); }
public void ClickOn9(View view) { onDigitClick("9", view); }
private void onDigitClick(String digit, View view) {
    if (pinCode.length() < 4) {
        pinCode += digit;
        PressEffectHelper.apply(this, view, "but");
        updatePinCircles();
    }
}
public void ClickFingerprint(View view) {
    PressEffectHelper.apply(this, view, "imgb");
    BiometricAuthManager authManager = new BiometricAuthManager(this,
    MainWindow.class);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        authManager.authenticate();
    } else {
        Toast.makeText(this,
    getText(R.string.scan_fingerprint_unavailable),
    Toast.LENGTH_SHORT).show();
    }
}
public void ClickClearLastSymbol(View view) {
    if (pinCode.length() > 0) {
        pinCode = pinCode.substring(0, pinCode.length() - 1);
        updatePinCircles();
    }
    PressEffectHelper.apply(this, view, "imgb");
}
private void setPinCirclesColor(int color) {
    for (View circle : pinCircles) {
        // Получаем текущий Drawable
        GradientDrawable drawable = (GradientDrawable)
    circle.getBackground();
        drawable.setColor(color); // Устанавливаем новый цвет
    }
}
private void updatePinCircles() {
    for (int i = 0; i < pinCircles.length; i++) {
        if (i < pinCode.length()) {
            pinCircles[i].setBackgroundResource(R.drawable.pin_circle_filled);
        } else {

```

```

pinCircles[i].setBackgroundResource(R.drawable.pin_circle_empty);
    }
}
if (pinCode.length() == 4) {
    if (pinCode.equals(savedPin)) {
        // Успешный вход
        Intent intent = new Intent(EnterPinCode.this,
MainWindow.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
        finish();
    } else {
        // Ошибка: показываем красные кружки и вибрацию
        for (View circle : pinCircles) {
circle.setBackgroundResource(R.drawable.pin_circle_error);
        }
        VibrationHelper.triggerVibration(this);
        hintEnterPin.setText(getString(R.string.error_pin_code));
        pinCircles[0].postDelayed(() -> {
            pinCode = "";
            for (View circle : pinCircles) {
circle.setBackgroundResource(R.drawable.pin_circle_empty);
            }
            hintEnterPin.setText(getString(R.string.enter_pin_code));
        }, 200);

    }
}

}

public void fillPinCirclesOnSuccess() {
    for (View circle : pinCircles) {
circle.setBackgroundResource(R.drawable.pin_circle_filled);
    }
}

public void showAuthenticationError() {
for (View circle : pinCircles) {
circle.setBackgroundResource(R.drawable.pin_circle_error);
    }
    VibrationHelper.triggerVibration(this);
    pinCircles[0].postDelayed(() -> {
pinCode = "";
        for (View circle : pinCircles) {
            circle.setBackgroundResource(R.drawable.pin_circle_empty);
        }
        }, 200);
}

}
}

```

Файл EthereumManager.kt

```
package com.example.coldbitwallet.wallet.ethereum
```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		142

```

import android.os.Build
import android.util.Log
import androidx.annotation.RequiresApi
import
com.example.coldbitwallet.wallet.HDWalletManager.HDWalletMaster
import org.bitcoinj.crypto.ChildNumber
import org.bitcoinj.crypto.HDKeyDerivation
import org.web3j.crypto.*
import org.web3j.protocol.Web3j
import org.web3j.protocol.http.HttpService
import org.web3j.protocol.core.DefaultBlockParameterName
import org.web3j.utils.Numeric
import java.math.BigInteger
import org.web3j.abi.FunctionEncoder
import org.web3j.abi.FunctionReturnDecoder
import org.web3j.abi.datatypes.*
import org.web3j.abi.datatypes.generated.Uint256
import org.web3j.abi.TypeReference
import org.web3j.protocol.core.methods.request.Transaction
import org.web3j.utils.Convert
import java.math.BigDecimal

object EthereumManager {

    private const val PYUSD_CONTRACT =
"0xCaC524BcA292aaade2DF8A05cC58F0a65B1B3bB9"
    // NETWORK
    enum class EthNetwork(
        val rpcUrl: String,
        val chainId: Long
    ) {
        MAINNET(
"https://mainnet.infura.io/v3/38610b80dd754126b6abe4633858ed65",
            1
        ),
        SEPOLIA(
"https://sepolia.infura.io/v3/38610b80dd754126b6abe4633858ed65",
            11155111
        )
    }
    // KEY DERIVATION (BIP44 m/44'/60'/0'/0/index)
    private fun deriveKey(
        master: HDWalletMaster,
        index: Int
    ): ECKeyPair {
        val path = listOf(
            ChildNumber(44, true),
            ChildNumber(60, true),
            ChildNumber(0, true),
            ChildNumber(0, false), // change НЕ используется в ETH

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		143

```

        ChildNumber(index, false)
    )

    var key = master.masterKey
    for (child in path) {
        key = HDKeyDerivation.deriveChildKey(key, child)
    }
    return ECKeyPair.create(key.privKeyBytes)
}
// ADDRESS
@JvmStatic
fun getAddress(
    master: HDWalletMaster,
    index: Int = 0
): String {
    val pair = deriveKey(master, index)
    return "0x" + Keys.getAddress(pair.publicKey)
}
// BALANCE
fun getErc20Balance(
    userAddress: String,
    tokenAddress: String,
    network: EthNetwork
): BigInteger {
    val web3 = Web3j.build(HttpService(network.rpcUrl))
    val function = Function(
        "balanceOf",
        listOf(Address(userAddress)),
        listOf(object : TypeReference<Uint256>() {})
    )
    val encoded = FunctionEncoder.encode(function)
    val response = web3.ethCall(
        Transaction.createEthCallTransaction(
            userAddress,
            tokenAddress,
            encoded
        ),
        DefaultBlockParameterName.LATEST
    ).send()
    val decoded = FunctionReturnDecoder.decode(
        response.value,
        function.outputParameters
    )
    return decoded[0].value as BigInteger
}
// SEND ETH
@RequiresApi(Build.VERSION_CODES.N)
fun sendEthereum(
    master: HDWalletMaster,
    toAddress: String,
    amountWei: BigInteger,
    network: EthNetwork,

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		144

```

        index: Int = 0
    ): String {
        val keyPair = deriveKey(master, index)
        val credentials = Credentials.create(keyPair)
        val web3 = Web3j.build(HttpService(network.rpcUrl))
        val fromAddress = credentials.address
        val nonce = web3.ethGetTransactionCount(
            fromAddress,
            DefaultBlockParameterName.PENDING

        ).send().transactionCount
        // ДИНАМИЧЕСКИЙ GAS
        val gasPrice = web3.ethGasPrice().send().gasPrice
        val gasLimit = BigInteger.valueOf(21000)
        // ПРОВЕРКА БАЛАНСА
        val balanceWei = web3.ethGetBalance(
            fromAddress,
            DefaultBlockParameterName.LATEST
        ).send().balance
        val totalCost = amountWei.add(gasPrice.multiply(gasLimit))
        if (balanceWei < totalCost) {
            throw Exception("Недостаточно средств (с комиссией)")
        }
        val tx = RawTransaction.createEtherTransaction(
            nonce,
            gasPrice,
            gasLimit,
            toAddress,
            amountWei
        )
        val signed = TransactionEncoder.signMessage(tx, credentials)
        val hex = Numeric.toHexString(signed)
        val response = web3.ethSendRawTransaction(hex).send()
        if (response.hasError()) {
            throw Exception(response.error.message)
        }
        val txHash = response.transactionHash
        waitForReceipt(web3, txHash)
        return txHash
    }
    @RequiresApi(Build.VERSION_CODES.N)
    private fun waitForReceipt(web3: Web3j, txHash: String) {
        repeat(30) { // до 30 сек
            val receipt =
web3.ethGetTransactionReceipt(txHash).send()
            if (receipt.transactionReceipt.isPresent) {
                val r = receipt.transactionReceipt.get()
                if (r.isStatusOK) {
                    Log.d("ETH", "TX SUCCESS")
                    return
                } else {

```

```

        throw Exception("TX failed")
    }
    }
    Thread.sleep(1000)
}
Log.d("ETH", "TX pending (not confirmed yet)")
}
@RequiresApi(Build.VERSION_CODES.N)
fun sendPYUSD(
    master: HDWalletMaster,
    toAddress: String,
    amount: BigDecimal,
    network: EthNetwork,
    index: Int = 0
): String {
    val keyPair = deriveKey(master, index)
    val credentials = Credentials.create(keyPair)
    val web3 = Web3j.build(HttpService(network.rpcUrl))
    val fromAddress = credentials.address
    val nonce = web3.ethGetTransactionCount(
        fromAddress,
        DefaultBlockParameterName.PENDING
    ).send().transactionCount
    // PYUSD = 6 decimals
    val amountRaw =
amount.multiply(BigDecimal.valueOf(1_000_000)).toBigInteger()val
function = Function(
    "transfer",
    listOf(
        Address(toAddress),
        Uint256(amountRaw)
    ),
    emptyList()
)
    val data = FunctionEncoder.encode(function)
    val gasPrice = Convert.toWei("5",
Convert.Unit.GWEI).toBigInteger()
    val gasLimit = BigInteger.valueOf(100_000)
    val tx = RawTransaction.createTransaction(
        nonce,
        gasPrice,
        gasLimit,
        PYUSD_CONTRACT,
        data
    )
    val signed = TransactionEncoder.signMessage(tx, credentials)
    val hex = Numeric.toHexString(signed)
    val response = web3.ethSendRawTransaction(hex).send()
    if (response.hasError()) {
        throw Exception("PYUSD send error:
${response.error.message}")
    }
}

```

```

    }
    val txHash = response.transactionHash
    waitForReceipt(web3, txHash)
    return txHash
}
fun getEthBalance(address: String, network: EthNetwork):
BigDecimal {
    val web3 = Web3j.build(HttpService(network.rpcUrl))
    val wei = web3.ethGetBalance(
        address,
        DefaultBlockParameterName.LATEST
    ).send().balance
    return Convert.fromWei(wei.toString(), Convert.Unit.ETHER)
}
fun getUsdtBalance(address: String, network: EthNetwork):
BigDecimal {
    val web3 = Web3j.build(HttpService(network.rpcUrl))
    val contractAddress = "0xCaC524BcA292aaade2DF8A05cC58F0a65B1B3bB9"
    // PYUSD / USDT-like
    val function = Function(
        "balanceOf",
        listOf(Address(address)),
        emptyList()
    )
    val encodedFunction = FunctionEncoder.encode(function)
    val response = web3.ethCall(
        Transaction.createEthCallTransaction(
            address,
            contractAddress,
            encodedFunction
        ),
        DefaultBlockParameterName.LATEST
    ).send()
    val value = response.value
    val balance = Numeric.toBigInt(value)
    // USDT = 6 decimals
    return BigDecimal(balance)
        .divide(BigDecimal.TEN.pow(6))
}
}

```

Файл EthereumTransactionChecker.java

```

package com.example.coldbitwallet;

import android.content.Context;
import android.content.SharedPreferences;
import android.util.Log;
import com.example.coldbitwallet.wallet.ethereum.EthereumManager;
import org.json.JSONArray;
import org.json.JSONObject;
import java.io.BufferedReader;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		147

```

import java.io.InputStreamReader;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.net.HttpURLConnection;
import java.net.URL;

public class EthereumTransactionChecker {
    private static final String TAG = "ETH_CHECKER";
    private static final String API_KEY =
"57K3YZ7ZFT5K6RQH24D7YA8TW7RIVNYVZ3";
    public static void checkIncoming(Context context, String
address, Callback callback) {
        new Thread(() -> {
            try {
                String urlStr =
                    "https://api.etherscan.io/v2/api" +
                    "?chainid=11155111" +
                    "&module=account" +
                    "&action=txlist" +
                    "&address=" + address +
                    "&startblock=0" +
                    "&endblock=99999999" +
                    "&sort=desc" +
                    "&apikey=" + API_KEY;
                URL url = new URL(urlStr);
                HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
                conn.setRequestMethod("GET");

                BufferedReader reader = new BufferedReader(
                    new InputStreamReader(conn.getInputStream())
                );
                StringBuilder result = new StringBuilder();
String line;
                while ((line = reader.readLine()) != null) {
                    result.append(line);
                }
                reader.close();
                String response = result.toString();
                Log.d(TAG, "RAW RESPONSE: " + response);
                JSONObject json = new JSONObject(response);
                // Проверка статуса
                String status = json.optString("status");
                if (!"1".equals(status)) {
                    Log.e(TAG, "API ERROR: " +
json.optString("result"));
                    return;
                }
                Object resultObj = json.get("result");
                // Защита от краша
                if (!(resultObj instanceof JSONArray)) {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		148

```

        Log.e(TAG, "Result is not array: " +
resultObj.toString());
        return;
    }
    JSONArray txs = (JSONArray) resultObj;
    for (int i = 0; i < txs.length(); i++) {
        JSONObject tx = txs.getJSONObject(i);
        String txHash = tx.getString("hash");
        String to = tx.getString("to").toLowerCase();
        String from = tx.getString("from");
        // фильтр failed транзакций
        String isError = tx.optString("isError", "0");
        if ("1".equals(isError)) continue;
    if (address.toLowerCase().equals(to)) {
        String value = tx.getString("value"); // wei
        BigInteger wei = new BigInteger(value);
        // точный ETH
        BigDecimal eth = new BigDecimal(wei)
            .divide(new
BigDecimal("10000000000000000000"));
        if (!isTxSaved(context, txHash)) {
            Log.d(TAG, "NEW INCOMING ETH TX: " +
txHash);
            TransactionHistoryManager.saveTransaction(
                context,
                "ETH",
                eth.doubleValue(),
                from,
                false // получено
            );
            BigDecimal balance =
EthereumManager.INSTANCE.getEthBalance(
                address,
                EthereumManager.EthNetwork.SEPOLIA
            );
            BalanceManager.getInstance().setBalance("eth",
balance.doubleValue());
            saveTx(context, txHash);
        }
    }
    }
    } catch (Exception e) {
        Log.e(TAG, "Error checking ETH txs", e);
    }
    if (callback != null) {
        new
android.os.Handler(android.os.Looper.getMainLooper())
            .post(callback::onFinished);
    }
    }).start();
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		149

```

    public static void checkIncomingERC20(Context context, String
address, Callback callback) {
        new Thread(() -> {
            try {
                Log.d(TAG, "checkIncomingERC20 started for address:
" + address);
                String urlStr = "https://api.etherscan.io/v2/api" +
                    "?chainid=11155111" +
                    "&module=account" +
                    "&action=tokenTx" +
                    "&address=" + address +
                    "&startblock=0" +
                    "&endblock=99999999" +
                    "&sort=desc" +
                    "&apikey=" + API_KEY;
                URL url = new URL(urlStr);
                HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
                conn.setRequestMethod("GET");
                BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                StringBuilder result = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    result.append(line);
                }
                reader.close();
                Log.d(TAG, "ERC20 API response: " +
result.toString());
                JSONObject json = new JSONObject(result.toString());
                String status = json.optString("status");
                if (!"1".equals(status)) {Log.e(TAG, "ERC20 API
error: " + json.optString("result"));
                    return;
                }
                JSONArray txs = json.optJSONArray("result");
                if (txs == null) {
                    Log.e(TAG, "ERC20 result is not array");
                    return;
                }
                Log.d(TAG, "Found " + txs.length() + " ERC20
transactions");
                for (int i = 0; i < txs.length(); i++) {
                    JSONObject tx = txs.getJSONObject(i);
                    String to = tx.getString("to").toLowerCase();
                    String tokenSymbol =
tx.getString("tokenSymbol");
                    String txHash = tx.getString("hash");
                    String from = tx.getString("from");
                    Log.d(TAG, "Tx " + i + ": hash=" + txHash + "
to=" + to + " address=" + address.toLowerCase());

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						150
Изм.	Лист	№ докум.	Подпись	Дата		

```

        if (!address.toLowerCase().equals(to)) {
            Log.d(TAG, "Skipping: to != address");
            continue;
        }
        if (isTxSaved(context, txHash)) {
            Log.d(TAG, "Skipping: already saved");
            continue;
        }
        BigInteger value = new
BigInteger(tx.getString("value"));
        int decimals =
Integer.parseInt(tx.getString("tokenDecimal"));
        BigDecimal amount = new
BigDecimal(value).divide(BigDecimal.TEN.pow(decimals));
        Log.d(TAG, "NEW ERC20 TX: " + tokenSymbol + " "
+ amount + " from " + from);
        TransactionHistoryManager.saveTransaction(
            context,
            BigDecimal balance =
EthereumManager.INSTANCE.getEthBalance(
                address,
                EthereumManager.EthNetwork.SEPOLIA
            );
            BalanceManager.getInstance().setBalance("eth",
balance.doubleValue());

            saveTx(context, txHash);
        }

        } catch (Exception e) {
            Log.e(TAG, "ERC20 check error", e);
        }
        if (callback != null) {
            new
android.os.Handler(android.os.Looper.getMainLooper()).post(callback:
:onFinished);
        }
    }).start();
}
private static boolean isTxSaved(Context context, String txid) {
    SharedPreferences prefs =
context.getSharedPreferences("eth_txs", Context.MODE_PRIVATE);
    return prefs.contains(txid);
}
private static void saveTx(Context context, String txid) {
    SharedPreferences prefs = context.getSharedPreferences("eth_txs",
Context.MODE_PRIVATE);
    prefs.edit().putBoolean(txid, true).apply();
}
public interface Callback {
    void onFinished();
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		151

```

    }
}

```

Файл GetActivity.java

```

package com.example.coldbitwallet;

import android.graphics.Bitmap;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class GetActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_get);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        // Применение цвета текста
        int[] textElements = {
            R.id.Get, R.id.warningText, R.id.tokenText,
R.id.addressText
        };
        for (int id : textElements) {
            TextColorManager.applySavedTextColor(this, id);
        }

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });
        ImageView qrImageView = findViewById(R.id.qrImageView);
        TextView addressTextView = findViewById(R.id.addressText);
        TextView tokenTextView = findViewById(R.id.tokenText);

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		152

```

        TextView warningTextView = findViewById(R.id.warningText);
        // Получаем токен из интента
        String token = getIntent().getStringExtra("selected_token");
        if (token == null || token.isEmpty()) {
            Toast.makeText(this,
getString(R.string.toast_no_detect_token),
Toast.LENGTH_SHORT).show();
            finish();
            return;
        }
        // Выбираем цвет фона в зависимости от токена
        int backgroundColorRes;
        switch (token.toUpperCase()) {
            case "BTC":
                backgroundColorRes = R.color.BTCColor;
                break;
            case "ETH":
                backgroundColorRes = R.color.ETHColor;
                break;
            case "TON":
                backgroundColorRes = R.color.TONColor;
                break;
            case "USDT":
                backgroundColorRes = R.color.USDTColor;
                break;
            case "SOL":
                backgroundColorRes = R.color.SOLColor;
                break;
            default:
                backgroundColorRes = R.color.TopBackground;
                break;
        }
        // Получаем адрес из BalanceManager
        String address =
BalanceManager.getInstance().getAddress(token);
        if (address == null || address.isEmpty()) {
            Toast.makeText(this,
getString(R.string.toast_no_address_token, token),
Toast.LENGTH_SHORT).show();
            finish();
            return;
        }
        // Генерируем и отображаем QR-код с нужным фоном и текстом
        Bitmap qrBitmap = QRCodeGenerator.generateQRCode(this,
address, backgroundColorRes, token);
        if (qrBitmap != null) {
            qrImageView.setImageBitmap(qrBitmap);
        } else {
            Toast.makeText(this,
getString(R.string.toast_no_generate_qr_code),
Toast.LENGTH_SHORT).show();

```

```

    }
    // Применяем цвет фона к контейнеру
    LinearLayout tokenAddressContainer =
findViewById(R.id.tokenAddressContainer);

tokenAddressContainer.setBackgroundResource(R.drawable.rounded_bg);
    // Устанавливаем токен в TextView
tokenTextView.setText(getString(R.string.address_token_text,
token));
    // Устанавливаем адрес в TextView
addressTextView.setText(address);
    // Устанавливаем предупреждение с подстановкой токена
warningTextView.setText(getString(R.string.warning_token_hint,
token));
}
public void ClickCancel(View view) {
    PressEffectHelper.apply(this, view, "sss");
    finish();
}
}

```

Файл HDWalletManager.java

```

package com.example.coldbitwallet.wallet

import android.content.Context
import android.os.Build
import android.util.Base64
import androidx.annotation.RequiresApi
import org.bitcoinj.crypto.DeterministicKey
import org.bitcoinj.crypto.HDKeyDerivation
import org.bitcoinj.crypto.MnemonicCode
import java.security.KeyStore
import java.security.SecureRandom
import javax.crypto.Cipher
import javax.crypto.spec.SecretKeySpec

object HDWalletManager {
    // MNEMONIC
    @JvmStatic
    fun generateMnemonic(): List<String> {
        // Используем конструктор без параметров - он сам загрузит
        английский словарь из ресурсов
        val mnemonicCode = MnemonicCode()
        val entropy = ByteArray(32)
        SecureRandom().nextBytes(entropy)
        return mnemonicCode.toMnemonic(entropy)
    }
    @JvmStatic
    fun mnemonicToSeed(mnemonic: List<String>, passphrase: String =
    ""): ByteArray {
        return MnemonicCode.toSeed(mnemonic, passphrase)
    }
}

```

```

    }
    // SAVE SEED (ENCRYPT)
    @RequiresApi(Build.VERSION_CODES.FROYO)
    @JvmStatic
    fun saveSeed(context: Context, seed: ByteArray) {
        val prefs = context.getSharedPreferences("MyPrefs",
Context.MODE_PRIVATE)
        val aesKey = ByteArray(32).also {
SecureRandom().nextBytes(it) }
        val aesCipher = Cipher.getInstance("AES")
        aesCipher.init(Cipher.ENCRYPT_MODE, SecretKeySpec(aesKey,
"AES"))
        val encryptedSeed = aesCipher.doFinal(seed)
        val keyStore = KeyStore.getInstance("AndroidKeyStore")
        keyStore.load(null)
        val publicKey =
keyStore.getCertificate("seed_rsa_key").publicKey
        val rsaCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding")
        rsaCipher.init(Cipher.ENCRYPT_MODE, publicKey)
        val encryptedAesKey = rsaCipher.doFinal(aesKey)
        prefs.edit()
            .putString("encrypted_seed",
Base64.encodeToString(encryptedSeed, Base64.DEFAULT))
            .putString("encrypted_aes_key",
Base64.encodeToString(encryptedAesKey, Base64.DEFAULT))
            .apply()
    }
    // GET SEED (DECRYPT)
    @RequiresApi(Build.VERSION_CODES.FROYO)
    @JvmStatic
    fun getSeed(context: Context): ByteArray {
        val prefs = context.getSharedPreferences("MyPrefs",
Context.MODE_PRIVATE)
        val encryptedSeed =
Base64.decode(prefs.getString("encrypted_seed", null),
Base64.DEFAULT)
        val encryptedAesKey =
Base64.decode(prefs.getString("encrypted_aes_key", null),
Base64.DEFAULT)
        val keyStore = KeyStore.getInstance("AndroidKeyStore")
        keyStore.load(null)
        val privateKey = keyStore.getKey("seed_rsa_key", null)
        val rsaCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding")
        rsaCipher.init(Cipher.DECRYPT_MODE, privateKey)
        val aesKey = rsaCipher.doFinal(encryptedAesKey)
        val aesCipher = Cipher.getInstance("AES")
        aesCipher.init(Cipher.DECRYPT_MODE, SecretKeySpec(aesKey, "AES"))
        return aesCipher.doFinal(encryptedSeed)
    }
    // HDWalletMaster
    class HDWalletMaster(

```

```

        val masterKey: DeterministicKey
    ) {
        constructor(seed: ByteArray) :
this(HDKeyDerivation.createMasterPrivateKey(seed))
    }
    @JvmStatic
    fun createMasterFromMnemonic(
        mnemonic: List<String>,
        passphrase: String = ""
    ): HDWalletMaster {
        val seed = mnemonicToSeed(mnemonic, passphrase)
        return HDWalletMaster(seed)
    }
}

```

Файл HistoryAction.java

```

package com.example.coldbitwallet;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.Locale;
import android.util.TypedValue;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import static android.view.ViewGroup.LayoutParams.WRAP_CONTENT;
import android.widget.LinearLayout;
import android.widget.ImageView;
import android.widget.TextView;
import android.view.Gravity;
import android.view.ViewGroup;
import android.graphics.Typeface;
import android.widget.Toast;
import androidx.activity.OnBackPressedCallback;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

public class HistoryAction extends AppCompatActivity {
    private int ColorText;
    private DoubleBackClickManager doubleBackPressHandler;
    private boolean emptyMessageShown = false; // Флаг для
предотвращения дублирования сообщения
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		156

```

        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_history_action);
        doubleBackPressedHandler = new DoubleBackClickManager(this);
        // Применение сохранённого фона
        ThemeManager.applySavedBackground(this, R.id.main);
        // Применение цвета текста
        int[] textElements = {
            R.id.textHome, R.id.textHistory
        };
        for (int id : textElements)
        {TextColorManager.applySavedTextColor(this, id);
        }
        ColorText = TextColorManager.getSavedTextColor(this);
        SystemBarColorHelper.applyColors(this, true);
        // Получаем контейнер для истории
        LinearLayout historyLayout =
        findViewById(R.id.historyLayout);
        String btcAddress =
        BalanceManager.getInstance().getAddress("btc");
        String ethAddress =
        BalanceManager.getInstance().getAddress("eth");
        String pyusdAddress =
        BalanceManager.getInstance().getAddress("eth"); // USDT (ERC-20) на
        том же ETH-адресе
        String solAddress =
        BalanceManager.getInstance().getAddress("sol");
        // Передаём колбэк refreshHistory во все методы проверки
        if (btcAddress != null && !btcAddress.isEmpty()) {
            BitcoinTransactionChecker.checkIncoming(this,
            btcAddress, this::refreshHistory);
        }
        if (ethAddress != null && !ethAddress.isEmpty()) {
            EthereumTransactionChecker.checkIncoming(this,
            ethAddress, this::refreshHistory);
        }
        if (pyusdAddress != null && !pyusdAddress.isEmpty()) {
            EthereumTransactionChecker.checkIncomingERC20(this,
            pyusdAddress, this::refreshHistory);
        }
        // Первоначальное отображение истории (из уже сохранённого
        файла)
        displayTransactionHistory(historyLayout);
        getOnBackPressedDispatcher().addCallback(this, new
        OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {

            doubleBackPressedHandler.onBackPressed(R.string.toast_exit_app);
            }
        });
    }
}

```

```

        // Вызывается из асинхронных чекеров после добавления новых
        транзакций в файл.
        private void refreshHistory() {
            runOnUiThread(() -> {
                LinearLayout historyLayout =
findViewById(R.id.historyLayout);
                if (historyLayout != null) {
                    displayTransactionHistory(historyLayout);
                }
            });
        }
        private void displayTransactionHistory(LinearLayout
historyLayout) {
            try {
                File historyFile =
TransactionHistoryManager.getHistoryFile(this);
                if (!historyFile.exists()) {
                    if (!emptyMessageShown) {
                        showEmptyHistoryMessage(historyLayout);
                        emptyMessageShown = true;
                    }
                    return;
                }
                BufferedReader reader = new BufferedReader(new
FileReader(historyFile));
                String directionLine;
                String detailsLine;
                historyLayout.removeAllViews();
                while ((directionLine = reader.readLine()) != null &&
                    (detailsLine = reader.readLine()) != null) {
                    reader.readLine(); // Пустая строка разделителя
                    // Парсим строку directionLine для разделения типа и
адреса
                    String[] directionParts = directionLine.split(" ");
                    String type = directionParts[0].trim(); // Тип
перевода (Отправлено / Получено)
                    String address = directionParts.length > 1 ?
directionParts[1].trim() : ""; // Адрес кошелька
                    // Парсим строку detailsLine для получения данных о
токене и сумме
                    String[] parts = detailsLine.split(", ");
                    if (parts.length < 3) continue;
                    String token = parts[0].replace("Token: ",
"".trim());
                    String amount = parts[1].replace("Amount: ",
"".trim());
                    String date = parts[2].replace("Date: ", "").trim();
                    // Создание блока для истории
                    LinearLayout block = new LinearLayout(this);
                    block.setOrientation(LinearLayout.VERTICAL);
                    block.setPadding(24, 24, 24, 24);

```

```

        block.setBackgroundResource(R.drawable.rounded_bg);
// скругленный фон
        // 1. Адрес кошелька
        TextView addressView = new TextView(this);
        addressView.setText(address);
        addressView.setTextSize(10f);
        addressView.setTypeface(null, Typeface.BOLD);
        addressView.setTextColor(ColorText);
        block.addView(addressView);
        // --- 2 строка: иконка токена + количество
        (прижатое к правому краю)
        LinearLayout iconAmountRow = new LinearLayout(this);

        iconAmountRow.setOrientation(LinearLayout.HORIZONTAL);
        iconAmountRow.setLayoutParams(new
        LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            ViewGroup.LayoutParams.WRAP_CONTENT
        ));
        iconAmountRow.setGravity(Gravity.CENTER_VERTICAL);
        iconAmountRow.setPadding(0, 8, 0, 8);
        // Иконка токена
        ImageView icon = new ImageView(this);
        int iconRes = getTokenIconResource(token);
        icon.setImageResource(iconRes);
        int sizeInDp = (int) TypedValue.applyDimension(
            TypedValue.COMPLEX_UNIT_DIP,
            22,
            getResources().getDisplayMetrics()
        );
        LinearLayout.LayoutParams iconParams = new
        LinearLayout.LayoutParams(
            sizeInDp,
            sizeInDp
        );
        iconParams.setMarginEnd((int)
        TypedValue.applyDimension(
            TypedValue.COMPLEX_UNIT_DIP,
            8,
            getResources().getDisplayMetrics()
        ));
        icon.setLayoutParams(iconParams);
        // Количество токенов (прижатое к правому краю)
        TextView amountText = new TextView(this);
        try {
            double amountValue = Double.parseDouble(amount);
            DecimalFormatSymbols symbols = new
            DecimalFormatSymbols(Locale.US);
            DecimalFormat formatter = new
            DecimalFormat("0.#####", symbols);

```

```

        String formattedAmount =
formatter.format(amountValue);
        amountText.setText(formattedAmount);
    } catch (NumberFormatException e) {
        amountText.setText(amount);
    }
    amountText.setTextSize(16f);
amountText.setTypeface(null, Typeface.BOLD);
    amountText.setTextColor(ColorText);
    amountText.setGravity(Gravity.END); // Выравнивание
текста по правому краю
    LinearLayout.LayoutParams amountParams = new
LinearLayout.LayoutParams(
        0, // Ширина будет вычисляться по весу
        ViewGroup.LayoutParams.WRAP_CONTENT,
        1f // Вес 1 для растягивания
    );
    amountParams.gravity = Gravity.END; // Прижимаем
весь элемент к правому краю
    amountText.setLayoutParams(amountParams);
    // Добавляем элементы
    iconAmountRow.addView(icon);
    iconAmountRow.addView(amountText);
    block.addView(iconAmountRow);
    // 3. Тип транзакции + дата (по краям)
    LinearLayout typeDateRow = new LinearLayout(this);
    typeDateRow.setOrientation(LinearLayout.HORIZONTAL);
    typeDateRow.setGravity(Gravity.CENTER_VERTICAL);
    // Тип транзакции (Отправлено / Получено)
    TextView typeText = new TextView(this);
    // Локализация текста (Отправлено / Получено)
    String localizedType;
    if ("Отправлено".equals(type)) {
        localizedType =
getString(R.string.transaction_sent);
    } else if ("Получено".equals(type)) {
        localizedType =
getString(R.string.transaction_received);
    } else {
        localizedType = type;
    }
    typeText.setText(localizedType);
    typeText.setTextSize(12f);
    typeText.setTextColor(ColorText);
    // Дата
    TextView dateText = new TextView(this);
    dateText.setText(date);
    dateText.setTextSize(12f);
    dateText.setTextColor(ColorText);
    dateText.setGravity(Gravity.END);

```

```

        LinearLayout.LayoutParams typeParams = new
LinearLayout.LayoutParams(0, WRAP_CONTENT, 1f);
        LinearLayout.LayoutParams dateParams = new
LinearLayout.LayoutParams(WRAP_CONTENT, WRAP_CONTENT);
        typeText.setLayoutParams(typeParams);
        dateText.setLayoutParams(dateParams);
        typeDataRow.addView(typeText);
        typeDataRow.addView(dateText);
        block.addView(typeDataRow);
        // Отступ между блоками
        LinearLayout.LayoutParams blockParams = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
WRAP_CONTENT);
        blockParams.setMargins(0, 0, 0, 24);

        historyLayout.addView(block, 0, blockParams);
    }
    reader.close();
    if (historyLayout.getChildCount() == 0) {
        if (!emptyMessageShown) {
            showEmptyHistoryMessage(historyLayout);
            emptyMessageShown = true;
        }
    } else {
        emptyMessageShown = false; // Есть транзакции,
сбрасываем флаг
    }
} catch (IOException e) {
    e.printStackTrace();
    Toast.makeText(this,
getString(R.string.toast_error_read_history),
Toast.LENGTH_SHORT).show();
}
}

private int getTokenIconResource(String token) {
    switch (token.toUpperCase()) {
        case "BTC":
            return R.drawable.btc_icon;
        case "ETH":
            return R.drawable.eth_icon;
        case "TON":
            return R.drawable.ton_icon;
        case "USDT":
            return R.drawable.usdt_icon;
        case "PYUSD":
            return R.drawable.usdt_icon;
        case "SOL":
            return R.drawable.sol_icon;
        default:
            return R.drawable.sent;
    }
}

```

```

    }
    private void showEmptyHistoryMessage(LinearLayout layout) {
        // Очищаем макет и добавляем одно сообщение
        layout.removeAllViews();
        TextView emptyView = new TextView(this);
        emptyView.setText(getString(R.string.empty_history_text));
        emptyView.setTextColor(ColorText);
        emptyView.setTextSize(16f);
        emptyView.setGravity(Gravity.CENTER);
        layout.addView(emptyView);
    }
    public void onHomeClicked(View view) {
        PressEffectHelper.apply(this, view, "sss");
        Intent intent = new Intent(HistoryAction.this,
MainWindows.class);
        startActivity(intent);
        finish();
    }
    public void onHistoryClicked(View view) {
        PressEffectHelper.apply(this, view, "sss");
    }
    public void DeleteHistoryTransaction(View view) {
        PressEffectHelper.apply(this, view, "sss");

        new AlertDialog.Builder(this)
            .setTitle(getString(R.string.confirm_text1))
            .setMessage(getString(R.string.confirm_text2))
            .setPositiveButton(getString(R.string.yes_text),
(dialog, which) -> {
TransactionHistoryManager.clearTransactionHistory(this);
                // Очистка визуальной истории
                LinearLayout historyLayout =
findViewById(R.id.historyLayout);
                historyLayout.removeAllViews();
                Toast.makeText(this,
getString(R.string.toast_delete_history),
Toast.LENGTH_SHORT).show();
                // Сбрасываем флаг и показываем сообщение о
пустой истории
                emptyMessageShown = false;
                showEmptyHistoryMessage(historyLayout);
                emptyMessageShown = true;
            })
            .setNegativeButton(getString(R.string.no_text),
(dialog, which) -> dialog.dismiss())
            .show();
        }
    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

```

```

        if (doubleBackPressedHandler != null) {
            doubleBackPressedHandler.cancel();
        }
    }
}

```

Файл JustSend.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.os.Bundle;
import android.text.InputFilter;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class JustSend extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_just_send);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        // Применение сохранённого фона
        ThemeManager.applySavedBackground(this, R.id.main);
        // Применение цвета текста
        int[] textElements = {
            R.id.NextButtonSendByScanner, R.id.Recipient,
R.id.AddressWalletText, R.id.HintText
        };
        for (int id : textElements) {
            TextColorManager.applySavedTextColor(this, id);
        }
        // Применение сохраненного цвета для кнопок
        Button nextButton =
findViewById(R.id.NextButtonSendByScanner);
        PressEffectHelper.apply(this, nextButton, "spec");

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		163

```

        Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
        return insets;
    });
    EditText editText = findViewById(R.id.AdressWalletText);
    // Блокируем ручной ввод \n
    editText.setFilters(new InputFilter[]{
        (source, start, end, dest, dstart, dend) -> {
            for (int i = start; i < end; i++) {
                if (source.charAt(i) == '\n') return "";
            }
            return null;
        }
    });
}
public void ClickNext(View view) {
    PressEffectHelper.apply(this, view, "spec");
    EditText addressEditText =
findViewById(R.id.AdressWalletText);
    String currentAddress =
addressEditText.getText().toString().replace("\n", "").trim(); //
удаляем \n
    // Проверка на пустой адрес или адрес с меньшим количеством
символов, чем 10
    if (currentAddress.isEmpty() || currentAddress.length() <
10) {
        // Отображаем Toast с сообщением об ошибке
        Toast.makeText(this,
getString(R.string.toast_incorrect_address),
Toast.LENGTH_SHORT).show();
        VibrationHelper.triggerVibration(this);
        return; // Завершаем выполнение метода
    }
    // Если адрес корректен, продолжаем с intent
    Intent intent = new Intent(JustSend.this,
AssetSelectionSet.class);
    intent.putExtra("SCAN_RESULT", currentAddress);
    startActivity(intent);
}
public void ClickCancel(View view) {
    PressEffectHelper.apply(this, view, "sss");
    finish();
}
}

```

Файл LanguageSelection.java

```

package com.example.coldbitwallet;

import android.content.Intent;

```

```

import android.content.SharedPreferences;
import android.content.res.ColorStateList;
import android.content.res.Resources;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;

public class LanguageSelection extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_language_selection);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        // Применение сохранённого фона
        ThemeManager.applySavedBackground(this, R.id.main);
        // Применение цвета текста
        int[] textElements = {
            R.id.select_language_text, R.id.SystemLanguage,
R.id.RussianLanguage, R.id.EndlishLanguage,
            R.id.SpanishLanguage, R.id.HindiLanguage,
R.id.ChineseLanguage
        };

        for (int id : textElements) {
            TextColorManager.applySavedTextColor(this, id);
        }
        LinearLayout languageListLayout =
findViewById(R.id.language_list_layout);
        int textColor = TextColorManager.getSavedTextColor(this); //
Получаем сохранённый цвет текста
        GradientDrawable dividerDrawable = new GradientDrawable();
        dividerDrawable.setColor(textColor); // Цвет разделителя =
цвет текста

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		165

```

        dividerDrawable.setSize(0, 2); // Толщина линии (высота) 2px
        languageListLayout.setDividerDrawable(dividerDrawable);
        // Настроим отображение системного языка в TextView
        TextView systemLangTextView =
findViewById(R.id.SystemLanguage);
        String systemLangName = getSystemLanguageName();
        String label = getString(R.string.language_system) + " (" +
systemLangName + ")";
        systemLangTextView.setText(label);
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });
        highlightCurrentLanguage();
    }
    private String getSystemLanguageName() {
        Locale systemLocale =
Resources.getSystem().getConfiguration().locale;
        return
systemLocale.getDisplayLanguage(systemLocale).substring(0,
1).toUpperCase() +

systemLocale.getDisplayLanguage(systemLocale).substring(1); // с
заглавной буквы
    }
    public void ClickCancel(View view) {
        PressEffectHelper.apply(this, view, "sss");
        finish();
    }
    private void changeLanguage(String langCode, View view) {
        PressEffectHelper.apply(this, view, "sss");
        // Устанавливаем локаль
        LocaleHelper.setLocale(this, langCode);

        // Сохраняем выбранный язык в SharedPreferences
        SharedPreferences prefs = getSharedPreferences("settings",
MODE_PRIVATE);
        prefs.edit().putString("language", langCode).apply();
        // Перезапускаем MainActivity
        Intent intent = new Intent(this, MainWindow.class); // Здесь
MainWindow - это ваше главное окно
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
        // Закрываем текущую активность
        finish();
    }
}

```

```

        public void ClickSystemLanguage(View view) {
            changeLanguage("system", view);
        }
        public void ClickRussian(View view) {
            changeLanguage("ru", view);
        }
        public void ClickEnglish(View view) {
            changeLanguage("en", view);
        }
        public void ClickChinese(View view) {
            changeLanguage("zh", view);
        }
        public void ClickHindi(View view) {
            changeLanguage("hi", view);
        }

        public void ClickSpanish(View view) {
            changeLanguage("es", view);
        }

        private void highlightCurrentLanguage() {
            String currentLang = getSharedPreferences("settings",
MODE_PRIVATE)
                .getString("language", "system");

            int savedTextColor =
TextColorManager.getSavedTextColor(this); // Получаем сохранённый
цвет текста

            Map<String, Integer> langToCircle = new HashMap<>();
            langToCircle.put("system", R.id.system_language_indicator);
            langToCircle.put("ru", R.id.russian_language_indicator);
            langToCircle.put("en", R.id.english_language_indicator);
            langToCircle.put("es", R.id.spanish_language_indicator);
            langToCircle.put("hi", R.id.hindi_language_indicator);
            langToCircle.put("zh", R.id.chinese_language_indicator);

            for (Map.Entry<String, Integer> entry :
langToCircle.entrySet()) {
                ImageView circle = findViewById(entry.getValue());
                if (circle != null) {
                    if (entry.getKey().equals(currentLang)) {
                        circle.setVisibility(View.VISIBLE);

circle.setImageTintList(ColorStateList.valueOf(savedTextColor));
                    } else {
                        circle.setVisibility(View.INVISIBLE);
                    }
                }
            }
        }
    }

```

```
}
```

Файл LocalHelper.java

```
package com.example.coldbitwallet;

import android.content.Context;
import android.content.res.Configuration;
import android.content.res.Resources;
import java.util.Locale;

public class LocaleHelper {
    public static void setLocale(Context context, String
languageCode) {
        Locale locale;
        if (languageCode == null || languageCode.equals("system")) {
            locale =
Resources.getSystem().getConfiguration().locale; // ЯЗЫК СИСТЕМЫ
        } else {
            locale = new Locale(languageCode);
        }

        Locale.setDefault(locale);
        Configuration config = new Configuration();
        config.setLocale(locale);
        context.getResources().updateConfiguration(config,
context.getResources().getDisplayMetrics());
    }
}
```

Файл PressEffectHelper.java

```
package com.example.coldbitwallet;

import android.content.Context;
import android.content.res.ColorStateList;
import android.graphics.drawable.Drawable;
import android.graphics.drawable.GradientDrawable;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import androidx.core.content.ContextCompat;

public class PressEffectHelper {
    // Основной метод, куда передаем тип и сам элемент
    public static void apply(Context context, View view, String
type) {
        if (view == null || type == null) return;
        int pressedColor = ContextCompat.getColor(context,
R.color.TopBackground);
        int defaultColor = ThemeManager.getSavedColor(context);
        int ButColor =
ButtonsColorManager.getSavedButtonColor(view.getContext());
        // В зависимости от типа обрабатываем
```

					ПГУ 1.090301.29.001 ПЗ	Лист
						168
Изм.	Лист	№ докум.	Подпись	Дата		

```

        switch (type) {
            case "but": // для Button без фона с закруглением
                applyTintEffect((Button) view, pressedColor,
defaultColor);
                break;
            case "imgb": // для ImageButton без закругления
                applyTintEffect((ImageButton) view, pressedColor,
defaultColor);
                break;
            case "draw": // для GradientDrawable
                Drawable background = view.getBackground();
                if (background instanceof GradientDrawable) {
                    applyGradientDrawableEffect(view,
(GradientDrawable) background.mutate(), pressedColor, defaultColor);
                }
                break;
            case "спец": // для Button с цветным фоном
                applyTintEffectSpecial((Button) view,
pressedColor, ButColor);
                break;
            default:
                // Если тип не подошел, то фолбэк на стандартный
эффект для любых View
                applySimpleColorEffect(view, pressedColor,
defaultColor);
                break;
        }
    }
    // Метод для кнопки (Button)
    private static void applyTintEffect(Button button, int
pressedColor, int defaultColor) {
        button.setBackgroundTintList(ColorStateList.valueOf(pressedColor));
        button.postDelayed(() ->

        button.setBackgroundTintList(ColorStateList.valueOf(defaultColor)),
100);
    }
    // Метод для кнопки (ImageButton)
    private static void applyTintEffect(ImageButton imageButton, int
pressedColor, int defaultColor) {
        imageButton.setBackgroundTintList(ColorStateList.valueOf(pressedColo
r));
        imageButton.postDelayed(() ->

        imageButton.setBackgroundTintList(ColorStateList.valueOf(defaultColo
r)), 100);
    }
    // Метод для GradientDrawable

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		169

```

        private static void applyGradientDrawableEffect(View view,
GradientDrawable drawable, int pressedColor, int defaultColor) {
            drawable.setColor(pressedColor);
            view.setBackground(drawable);
            view.postDelayed(() -> {
                drawable.setColor(defaultColor);
                view.setBackground(drawable);
            }, 100);
        }
        // Новый метод для Button с цветным фоном
        private static void applyTintEffectSpecial(Button button, int
pressedColor, int normalColor) {
            ColorStateList colorStateList = new ColorStateList(
                new int[][] {
                    new int[] { android.R.attr.state_pressed },
                    new int[] { -android.R.attr.state_pressed }
                },
                new int[] {
                    pressedColor,
                    normalColor
                }
            );
            button.setBackgroundTintList(colorStateList);
            button.refreshDrawableState();
        }
        // Метод для стандартного фона (fallback для всех других
случаев)
        private static void applySimpleColorEffect(View view, int
pressedColor, int defaultColor) {
            view.setBackgroundColor(pressedColor);
            view.postDelayed(() ->
view.setBackgroundColor(defaultColor), 100);
        }
    }

```

Файл PriceManager.java

```

package com.example.coldbitwallet;

import android.content.Context;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkCapabilities;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import com.example.coldbitwallet.wallet.ethereum.EthereumManager;
import org.jetbrains.annotations.NotNull;
import org.json.JSONObject;
import java.io.IOException;
import java.math.BigInteger;
import java.net.URLEncoder;

```

```

import java.nio.charset.StandardCharsets;
import java.util.concurrent.TimeUnit;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class PriceManager {
    private static final String TAG = "PriceManager";
    private static final String BASE_URL =
"http://192.168.1.100:7000/api/balances";
    private static final String PREFS_NAME = "wallet_data";
    private static final OkHttpClient client = new
OkHttpClient.Builder()
        .connectTimeout(10, TimeUnit.SECONDS)
        .readTimeout(10, TimeUnit.SECONDS)
        .build();

    public interface WalletCallback {
        void onSuccess();
        void onError(String error);
    }

    // SERVER REQUEST
    public static void fetchBalances(Context context,
                                     String btc,
                                     String eth,
                                     String ton,
                                     String sol,
                                     WalletCallback callback) {
        if (isEmpty(btc, eth, ton, sol)) {
            postError(callback, "Нет адресов для запроса");
            return;
        }
        try {
            String url = buildUrl(btc, eth, ton, sol);
            Log.d(TAG, "Request URL: " + url);

            Request request = new Request.Builder()
                .url(url)
                .get()
                .build();
            client.newCall(request).enqueue(new Callback() {
                @Override
                public void onFailure(Call call, IOException e) {
                    Log.e(TAG, "Request failed: ", e);
                    postError(callback, "Ошибка соединения с
сервером");
                }
                @Override
                public void onResponse(Call call, Response response)
            {

```

```

        try (Response res = response) {
            if (!res.isSuccessful()) {
                postError(callback, "Ошибка сервера: " +
res.code());

                return;
            }
            String body = res.body().string();
            Log.d(TAG, "Server response: " + body);
            JSONObject json = new JSONObject(body);
            saveToPrefs(context, json);
            postSuccess(callback);
        } catch (Exception e) {
            Log.e(TAG, "Parsing error: ", e);
            postError(callback, "Ошибка обработки
данных");
        }
    }
});
} catch (Exception e) {
    Log.e(TAG, "URL error: ", e);
    postError(callback, "Ошибка формирования запроса");
}
}

public static void fetchPyusdBalance(Context context, String
address, Runnable onDone) {
    new Thread(() -> {
        try {
            @NotNull BigInteger balance =
EthereumManager.INSTANCE.getErc20Balance(
                address,

                "0xCaC524BcA292aaade2DF8A05cC58F0a65B1B3bB9", // PYUSD
                EthereumManager.EthNetwork.SEPOLIA
            );
            SharedPreferences prefs =
                context.getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
            prefs.edit()
                .putString("usdt_balance",
balance.toString())
                .apply();

        } catch (Exception e) {
            Log.e("PriceManager", "PYUSD load error", e);
        }
        new Handler(Looper.getMainLooper()).post(onDone);
    }).start();
}

// SAVE DATA
private static void saveToPrefs(Context context, JSONObject
json) {

```

```

        SharedPreferences prefs =
            context.getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        double btcBalance = json.optDouble("btc", 0.0);
        double ethBalance = json.optDouble("eth", 0.0);
        double tonBalance = json.optDouble("ton", 0.0);
        double solBalance = json.optDouble("sol", 0.0);
        // USDT приходит в микро-единицах
        double usdtBalanceRaw = json.optDouble("usdt", 0.0);
        double usdtBalance = usdtBalanceRaw / 1_000_000.0;

        Log.d(TAG, "Saving balances - BTC: " + btcBalance + ", ETH:
" + ethBalance +
            ", TON: " + tonBalance + ", SOL: " + solBalance);
        editor.putString("btc_balance", String.valueOf(btcBalance));
        editor.putString("eth_balance", String.valueOf(ethBalance));
        editor.putString("ton_balance", String.valueOf(tonBalance));
        editor.putString("sol_balance", String.valueOf(solBalance));
        // НОРМАЛЬНЫЙ USDT
        editor.putString("usdt_balance",
String.valueOf(usdtBalance));
        // Цены
        double btcPrice = json.optDouble("BTCPrice", 0.0);
        double ethPrice = json.optDouble("ETHPrice", 0.0);
        double tonPrice = json.optDouble("TONPrice", 0.0);
        double solPrice = json.optDouble("SOLPrice", 0.0);
        Log.d(TAG, "Saving prices - BTC: " + btcPrice + ", ETH: " +
ethPrice +
            ", TON: " + tonPrice + ", SOL: " + solPrice);
        editor.putFloat("btc_price", (float) btcPrice);
        editor.putFloat("eth_price", (float) ethPrice);
        editor.putFloat("ton_price", (float) tonPrice);
        editor.putFloat("sol_price", (float) solPrice);
        editor.apply();
    }
    // LOCAL ACCESS
    public static double getBalance(Context context, String token) {
        SharedPreferences prefs =
            context.getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);
        String value = prefs.getString(token + "_balance", "0");
        Log.d(TAG, "getBalance(" + token + ") = " + value);
        double balance = Double.parseDouble(value);
        return balance;
    }
    public static double getPrice(Context context, String token) {
        SharedPreferences prefs =
            context.getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE);

```

```

        float value = prefs.getFloat(token + "_price", 0f);
        Log.d(TAG, "getPrice(" + token + ") = " + value);
return value;
    }
    // HELPERS
    private static boolean isEmpty(String... values) {
        for (String value : values) {
            if (value != null && !value.isEmpty()) return false;
        }
        return true;
    }
    private static String buildUrl(String btc, String eth, String
ton, String sol) throws Exception {
        return BASE_URL +
            "?btc=" + URLEncoder.encode(safe(btc),
StandardCharsets.UTF_8.name()) +
            "&eth=" + URLEncoder.encode(safe(eth),
StandardCharsets.UTF_8.name()) +
            "&ton=" + URLEncoder.encode(safe(ton),
StandardCharsets.UTF_8.name()) +
            "&sol=" + URLEncoder.encode(safe(sol),
StandardCharsets.UTF_8.name());
    }
    private static String safe(String value) {
        return value == null ? "" : value;
    }
    private static void postSuccess(WalletCallback callback) {
        new
Handler(Looper.getMainLooper()).post(callback::onSuccess);
    }

    private static void postError(WalletCallback callback, String
error) {
        new Handler(Looper.getMainLooper()).post(() ->
callback.onError(error));
    }
    // INTERNET CHECK
    public static boolean isInternetAvailable(Context context) {
        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        if (cm != null) {
            NetworkCapabilities capabilities =
cm.getNetworkCapabilities(cm.getActiveNetwork());
            return capabilities != null &&
capabilities.hasCapability(NetworkCapabilities.NET_CAPABILITY_INTERN
ET);
        }
        return false;
    }
}

```

```
}
```

Файл QRCodeGenerator.java

```
package com.example.coldbitwallet;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import androidx.core.content.ContextCompat;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.WriterException;
import com.google.zxing.qrcode.QRCodeWriter;

public class QRCodeGenerator {
    public static Bitmap generateQRCode(Context context, String
content, int backgroundColorResId, String tokenName) {
        QRCodeWriter writer = new QRCodeWriter();
        try {
            int size = 512;
            // Генерация QR-кода
            com.google.zxing.common.BitMatrix bitMatrix =
writer.encode(content, BarcodeFormat.QR_CODE, size, size);
            int darkColor = Color.BLACK; // Цвет элементов QR-кода
            int lightColor = ContextCompat.getColor(context,
backgroundColorResId); // Цвет фона QR-кода
            // Создаем изображение для QR-кода
            Bitmap qrBitmap = Bitmap.createBitmap(size, size,
Bitmap.Config.RGB_565);
            for (int x = 0; x < size; x++) {
                for (int y = 0; y < size; y++) {
                    qrBitmap.setPixel(x, y, bitMatrix.get(x, y) ?
darkColor : lightColor);
                }
            }
            // Создаём Bitmap для текста
            int textHeight = 100; // Высота области для текста
            Bitmap finalBitmap = Bitmap.createBitmap(size, size +
textHeight, Bitmap.Config.RGB_565);
            // Рисуем QR-код на новом Bitmap
            Canvas canvas = new Canvas(finalBitmap);
            canvas.drawBitmap(qrBitmap, 0, 0, null);

            // Добавляем фон для текста (динамический)
            int textBackgroundColor =
ContextCompat.getColor(context, backgroundColorResId);
            canvas.drawRect(0, size, size, size + textHeight, new
Paint() {{
                setColor(textBackgroundColor); // Фон для текста
            }});
        } catch (WriterException e) {
            e.printStackTrace();
        }
    }
}
```

```

        // Добавляем текст под QR-кодом
        Paint paint = new Paint();
        paint.setColor(Color.WHITE); // Цвет текста
        paint.setTextSize(30); // Размер текста
        paint.setTextAlign(Paint.Align.CENTER); // Выравнивание
текста по центру
        // Получаем текст из ресурсов
        String qrText = context.getString(R.string.scan_qr_text,
tokenName);
        // Разбиваем текст на строки
        float width = paint.measureText(qrText); // Ширина
текста
        float lineHeight = paint.getTextSize() * 1.5f; // Высота
строки
        float x = size / 2; // Позиция X (по центру)
        float y = size + textHeight / 2; // Позиция Y для первой
строки

        // Разбиваем текст по строкам
        if (width > size) {
            // Если текст слишком длинный, делим его на
несколько строк
            String[] lines = qrText.split("(?<=\\G.{32})"); //
Разделение на строки по 32 символов
            for (String line : lines) {
                canvas.drawText(line, x, y, paint);
                y += lineHeight; // Перенос на следующую строку
            }
        } else {
            // Если текст помещается, выводим его в одну строку
            canvas.drawText(qrText, x, y, paint);
        }
        return finalBitmap;
    } catch (WriterException e) {
        e.printStackTrace();
        return null;
    }
}
}

```

Файл SeedPhraswActivity.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.os.Bundle;

import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
import androidx.activity.EdgeToEdge;

```

```

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import java.util.ArrayList;

public class SeedPhraseActivity extends AppCompatActivity {
    private ArrayList<String> seedWords;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_seed_phrase);

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
            (v, insets) -> {
                Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
                v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
                return insets;
            });
        seedWords =
getIntent().getStringArrayListExtra("seedWords");
        // Применение сохраненного цвета для кнопок
        Button nextButton =
findViewById(R.id.NextButtonCreateNewWalletStage2);
        PressEffectHelper.apply(this, nextButton, "spec");
        LinearLayout leftColumn = findViewById(R.id.leftColumn);
        LinearLayout rightColumn = findViewById(R.id.rightColumn);
        if (seedWords != null && seedWords.size() == 24) {
            for (int i = 0; i < 24; i++) {
                TextView wordView = new TextView(this);
                wordView.setText((i + 1) + ". " + seedWords.get(i));
                wordView.setTextSize(16);
                wordView.setTextColor(ContextCompat.getColor(this,
R.color.TextColor));
                wordView.setPadding(0, 8, 0, 8);
                if (i < 12) {
                    leftColumn.addView(wordView);
                } else {
                    rightColumn.addView(wordView);
                }
            }
        }

        public void ClickCancel(View view) {
            PressEffectHelper.apply(this, view, "sss");
            finish();
        }
    }
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		177

```

    }
    public void ClickNextButtonCreateNewWalletStage3(View view) {
        Intent intent = new Intent(this, SeedPhraseActivity2.class);
        intent.putStringArrayListExtra("seedWords", seedWords);
        startActivity(intent);
    }
}

```

Файл SeedPhraswActivity2.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;

public class SeedPhraseActivity2 extends AppCompatActivity {
    private ArrayList<String> seedWords;
    private List<Integer> indicesToCheck;
    private EditText editableWord1, editableWord2, editableWord3,
    editableWord4;
    private TextView wordNumber1, wordNumber2, wordNumber3,
    wordNumber4;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_seed_phrase2);
        SystemBarColorHelper.applyColors(this, true);
        seedWords =
        getIntent().getStringArrayListExtra("seedWords");
        editableWord1 = findViewById(R.id.wordInput1);
        editableWord2 = findViewById(R.id.wordInput2);
        editableWord3 = findViewById(R.id.wordInput3);
        editableWord4 = findViewById(R.id.wordInput4);
        wordNumber1 = findViewById(R.id.wordNumber1);
        wordNumber2 = findViewById(R.id.wordNumber2);
        wordNumber3 = findViewById(R.id.wordNumber3);
        wordNumber4 = findViewById(R.id.wordNumber4);
        TextView wordPositionsText =
        findViewById(R.id.WordPositionsText);
    }
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		178

```

        // Применение сохраненного цвета для кнопок
        Button nextButton = findViewById(R.id.NextCreatePinCode);
        PressEffectHelper.apply(this, nextButton, "спец");
        // Генерируем случайные индексы для слов
        generateRandomIndices();
        // Показываем номера слов в формате "Введите слово №1, №2,
№3 и №4"
        String prompt = getString(R.string.seed_words_prompt,
            indicesToCheck.get(0) + 1, indicesToCheck.get(1) +
1,
            indicesToCheck.get(2) + 1, indicesToCheck.get(3) +
1);

        wordPositionsText.setText(prompt);
        // Устанавливаем подсказки в TextView
        wordNumber1.setHint(indicesToCheck.get(0) + 1 + ":");
        wordNumber2.setHint(indicesToCheck.get(1) + 1 + ":");
        wordNumber3.setHint(indicesToCheck.get(2) + 1 + ":");
        wordNumber4.setHint(indicesToCheck.get(3) + 1 + ":");
        wordNumber1.setHintTextColor(Color.LTGRAY);
        wordNumber2.setHintTextColor(Color.LTGRAY);
        wordNumber3.setHintTextColor(Color.LTGRAY);
        wordNumber4.setHintTextColor(Color.LTGRAY);
        // Добавляем слушатели на редактируемые поля
        setupTextWatcher(editableWord1, 0);
        setupTextWatcher(editableWord2, 1);
        setupTextWatcher(editableWord3, 2);
        setupTextWatcher(editableWord4, 3);
    }

    private void setupTextWatcher(EditText editText, final int
index) {
        editText.setFocusableInTouchMode(true);
        editText.setClickable(true);
        editText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence,
int start, int count, int after) {}
            @Override
public void onTextChanged(CharSequence charSequence, int start, int
before, int count) {}
            @Override
            public void afterTextChanged(Editable editable) {}
        });
    }

    private void generateRandomIndices() {
        indicesToCheck = new ArrayList<>();
        Random random = new Random();
        while (indicesToCheck.size() < 4) {
            int index = random.nextInt(24); // 0 to 23
            if (!indicesToCheck.contains(index)) {
                indicesToCheck.add(index);
            }
        }
    }

```

```

    }
    Collections.sort(indicesToCheck); // Чтобы были по порядку
}
public void ClickConfirmSeedWords(View view) {
    boolean allCorrect = true;
    // Проверяем введенные пользователем слова
    String[] userInputs = new String[4];
    userInputs[0] =
editableWord1.getText().toString().trim().toLowerCase();
    userInputs[1] =
editableWord2.getText().toString().trim().toLowerCase();
    userInputs[2] =
editableWord3.getText().toString().trim().toLowerCase();
    userInputs[3] =
editableWord4.getText().toString().trim().toLowerCase();
    for (int i = 0; i < 4; i++) {
        String correctWord =
seedWords.get(indicesToCheck.get(i)).toLowerCase();
        if (!userInputs[i].equals(correctWord)) {
            allCorrect = false;
        }
    }
    if (allCorrect) {
        // Переход к следующей активности
        Intent intent = new Intent(this, CreatePinCode.class);
        startActivity(intent);
    } else {
        VibrationHelper.triggerVibration(this);
        Toast.makeText(this,
R.string.toast_error_enter_seed_phrase, Toast.LENGTH_SHORT).show();
    }
}
public void ClickCancel(View view) {
    PressEffectHelper.apply(this, view, "sss");
    finish();
}
}

```

Файл SendByScanner.java

```

package com.example.coldbitwallet;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;
import android.text.InputFilter;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		180

```

import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class SendByScanner extends AppCompatActivity {
    private String scanResult;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_send_by_scanner);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        // Применение сохранённого фона
        ThemeManager.applySavedBackground(this, R.id.main);
        // Применение цвета текста
        int[] textElements = {
            R.id.NextButtonSendByScanner, R.id.Recipient,
R.id.HintText, R.id.AdressWalletText
        };
        for (int id : textElements) {
            TextColorManager.applySavedTextColor(this, id);
        }
        // Применение сохраненного цвета для кнопок
        Button nextButton =
findViewById(R.id.NextButtonSendByScanner);
        PressEffectHelper.apply(this, nextButton, "spec");
        scanResult = getIntent().getStringExtra("SCAN_RESULT");
        // Если строка начинается с "ton://transfer/", отрезаем этот
префикс
        if (scanResult != null &&
scanResult.startsWith("ton://transfer/")) {
            scanResult = scanResult.replace("ton://transfer/", "");
        }
        // Отображаем в TextView
        TextView recipientTextView =
findViewById(R.id.AdressWalletText);
        if (scanResult != null && !scanResult.isEmpty()) {
            StringBuilder formattedText = new StringBuilder();
            int maxLineLength = 28;
            for (int i = 0; i < scanResult.length(); i +=
maxLineLength) {
                int end = Math.min(i + maxLineLength,
scanResult.length());
                formattedText.append(scanResult, i,
end).append("\n");
            }
        }
    }
}

```

```

        if (formattedText.length() > 0 &&
formattedText.charAt(formattedText.length() - 1) == '\n') {
            formattedText.deleteCharAt(formattedText.length() -
1);
        }
        recipientTextView.setText(formattedText.toString());
        EditText editText = findViewById(R.id.AdressWalletText);
        editText.setFilters(new InputFilter[]{
            (source, start, end, dest, dstart, dend) -> {
                for (int i = start; i < end; i++) {
                    if (source.charAt(i) == '\n') return "";
                }
                return null;
            }
        });
    }

    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
        Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
        return insets;
    });
}

    public void ClickNext(View view) {
        PressEffectHelper.apply(this, view, "spec");
        EditText addressEditText =
findViewById(R.id.AdressWalletText);
        String currentAddress =
addressEditText.getText().toString().replace("\n", "").trim(); //
удаляем \n
        // Проверка на пустой адрес или адрес с меньшим количеством
СИМВОЛОВ,
чем 10
        if (currentAddress.isEmpty() || currentAddress.length() <
10) {
            // Отображаем Toast с сообщением об ошибке
            Toast.makeText(this,
getString(R.string.toast_incorrect_address),
Toast.LENGTH_SHORT).show();
            VibrationHelper.triggerVibration(this);
            return; // Завершаем выполнение метода
        }
        // Если адрес корректен, продолжаем с intent
        Intent intent = new Intent(SendByScanner.this,
AssetSelectionSet.class);
        intent.putExtra("SCAN_RESULT", currentAddress);
        startActivity(intent);
    }
}

```

```

        public void ClickCancel(View view) {
            PressEffectHelper.apply(this, view, "sss");
            finish();
        }
    }
}

```

Файл SettingActivity.java

```

package com.example.coldbitwallet;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Switch;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.google.android.material.bottomsheet.BottomSheetDialog;

public class SettingsActivity extends AppCompatActivity {
    private int TextColor;
    private Switch hideBalanceSwitch;
    private static final int REQUEST_CODE_CONFIRM_PIN = 1; // Целое
число для requestCode
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings",
MODE_PRIVATE).getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_settings);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        // Применение сохранённого фона
        ThemeManager.applySavedBackground(this, R.id.main);
        // Список всех id элементов
        int[] textViewIds = {
            // Системные штуки
            R.id.SettingsText, R.id.languageText,
R.id.themeText, R.id.textColor, R.id.hideBalance, R.id.resetPinCode,
            R.id.colorButton, R.id.exitFromAccount
        };
    }
}

```

```

        // Применяем цвет ко всем id
        for (int id : textViewIds) {
            TextColorManager.applySavedTextColor(this, id);
        }
        TextColor = TextColorManager.getSavedTextColor(this);
        // Восстанавливаем состояние переключателя
        hideBalanceSwitch = findViewById(R.id.hideBalanceSwitch);
        SharedPreferences prefs = getSharedPreferences("settings",
MODE_PRIVATE);
        boolean isBalanceHidden = prefs.getBoolean("hide_balance",
false); // false по умолчанию
        hideBalanceSwitch.setChecked(isBalanceHidden);
        hideBalanceSwitch.getTrackDrawable().setTint(TextColor);
        // Красим ползунок переключателя в зависимости от состояния
        if (hideBalanceSwitch.getThumbDrawable() != null) {
            if (isBalanceHidden) {
                // Если включен – цвет текста
                hideBalanceSwitch.getThumbDrawable().setTint(TextColor);
            } else {
                // Если выключен – системный серый цвет
                int defaultColor = ContextCompat.getColor(this,
android.R.color.darker_gray);
                hideBalanceSwitch.getThumbDrawable().setTint(defaultColor);
            }
        }

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });
    }
    public void ClickCancel(View view) {
        PressEffectHelper.apply(this, view, "sss");
        finish();
    }
    public void ClickLanguage(View view) {
        PressEffectHelper.apply(this, view, "draw");
        Intent intent = new Intent(SettingsActivity.this,
LanguageSelection.class);
        startActivity(intent);
    }
    public void ClickThemes(View view) {
        PressEffectHelper.apply(this, view, "draw");
        Intent intent = new Intent(SettingsActivity.this,
ThemesSetting.class);

```

```

        startActivity(intent);
    }
    public void ClickTextColor(View view) {
        PressEffectHelper.apply(this, view, "draw");
        Intent intent = new Intent(SettingsActivity.this,
TextColorActivity.class);
        startActivity(intent);
    }
    public void ClickColorButton(View view) {
        PressEffectHelper.apply(this, view, "draw");
        Intent intent = new Intent(SettingsActivity.this,
ButtonColorActivity.class);
        startActivity(intent);
    }
    public void ClickResetPinCode(View view) {
        // Применение эффекта нажатия
        PressEffectHelper.apply(this, view, "draw");
        // Запуск активности ConfirmPinCode для ввода пин-кода с
указанием причины
        Intent intent = new Intent(SettingsActivity.this,
ConfirmPinCode.class);
        intent.putExtra("reason", "change_pin"); // Добавляем
информацию о причине
        startActivityForResult(intent, 1); // 1 - это requestCode, который
будет использоваться для обработки результата
    }
    public void ClickHideBalance(View view) {
        PressEffectHelper.apply(this, view, "draw");

        hideBalanceSwitch = findViewById(R.id.hideBalanceSwitch);
        boolean isCurrentlyHidden = hideBalanceSwitch.isChecked();
// Текущее состояние
        if (!isCurrentlyHidden) {
            // Сейчас баланс ОТКРЫТ, пользователь хочет скрыть –
разрешаем сразу
            hideBalanceSwitch.setChecked(true);
            hideBalanceSwitch.getTrackDrawable().setTint(TextColor);
            if (hideBalanceSwitch.getThumbDrawable() != null) {
hideBalanceSwitch.getThumbDrawable().setTint(TextColor);
            }
            SharedPreferences prefs =
getSharedPreferences("settings", MODE_PRIVATE);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putBoolean("hide_balance", true);
            editor.apply();
            returnToMainWindow(this); // Можно сразу вернуться
        } else {
            Intent intent = new Intent(this, ConfirmPinCode.class);
            intent.putExtra("ACTION", "TOGGLE_HIDE_BALANCE");
            startActivityForResult(intent, 1001);
        }
    }

```

```

    }
}
@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 1001) { // Пришли сюда из
ConfirmPinCode для сокрытия/показа баланса
        if (resultCode == RESULT_OK) {
            String action = data != null ?
data.getStringExtra("ACTION") : null;
            if ("TOGGLE_HIDE_BALANCE".equals(action)) {
                hideBalanceSwitch =
findViewById(R.id.hideBalanceSwitch);
                // Пользователь подтвердил PIN – можно показать
баланс
                hideBalanceSwitch.setChecked(false); // Снимаем
скрытие

hideBalanceSwitch.getTrackDrawable().setTint(TextColor);
                if (hideBalanceSwitch.getThumbDrawable() !=
null) {
                    int defaultColor =
ContextCompat.getColor(this, android.R.color.darker_gray);
hideBalanceSwitch.getThumbDrawable().setTint(defaultColor);
                }
                // Сохраняем новое состояние
                SharedPreferences prefs =
getSharedPreferences("settings", MODE_PRIVATE);
                SharedPreferences.Editor editor = prefs.edit();
                editor.putBoolean("hide_balance", false);
                editor.apply();
                returnToMainWindow(this);
            } else {
                returnToMainWindow(this);
            }
        }
    } else if (requestCode == 1) { // Пришли сюда из
ConfirmPinCode для смены PIN-кода или перехода на MainWindow
        if (resultCode == RESULT_OK) {
            SharedPreferences preferences =
getSharedPreferences("MyPrefs", MODE_PRIVATE);
            String savedPin = preferences.getString("user_pin",
"");

            String enteredPin = data != null ?
data.getStringExtra("entered_pin") : "";
            if (enteredPin.equals(savedPin)) {
                String reason = data != null ?
data.getStringExtra("reason") : "";
                if ("change_pin".equals(reason)) {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		186

```

        Intent intent = new
Intent(SettingsActivity.this, CreatePinCode.class);
        startActivity(intent);
    } else {
        returnToMainWindow(this);
    }
    } else {
        Toast.makeText(this, R.string.error_pin_code,
Toast.LENGTH_SHORT).show();
    }
    }
    } else if (requestCode == 2) { // Подтверждение PIN для
выхода из аккаунта
        if (resultCode == RESULT_OK) {
            String action = data != null ?
data.getStringExtra("ACTION") : null;
            if ("EXIT".equals(action)) {
                // Очистка SharedPreferences
                getSharedPreferences("MyPrefs",
MODE_PRIVATE).edit().clear().apply();
                getSharedPreferences("theme_prefs",
MODE_PRIVATE).edit().clear().apply();
                getSharedPreferences("text_color_prefs",
MODE_PRIVATE).edit().clear().apply();
                getSharedPreferences("button_color_prefs",
MODE_PRIVATE).edit().clear().apply();
                // Очистка истории транзакций

TransactionHistoryManager.clearTransactionHistory(this);
                // Перезапуск приложения
                Intent intent = new
Intent(getApplicationContext(), CreateOrConnectWallet.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent);
                finishAffinity();
            }
        }
    }
    }
    public void ClickExitFromAccount(View view) {
        PressEffectHelper.apply(this, view, "draw");
        View bottomSheetView =
getLayoutInflater().inflate(R.layout.bottom_sheet_exit, null);
        BottomSheetDialog bottomSheetDialog = new
BottomSheetDialog(this);
        bottomSheetDialog.setContentView(bottomSheetView);
        bottomSheetDialog.show();
        Button btnConfirmExit =
bottomSheetView.findViewById(R.id.btnConfirmExit);

```

```

        Button btnCancel =
bottomSheetView.findViewById(R.id.btnCancel);
        btnConfirmExit.setOnClickListener(v -> {
            bottomSheetDialog.dismiss();

            Intent intent = new Intent(this, ConfirmPinCode.class);
            intent.putExtra("ACTION", "EXIT");
            startActivityForResult(intent, 2);
        });
        btnCancel.setOnClickListener(v ->
bottomSheetDialog.dismiss());
    }
    public void returnToMainWindow(Context context) {
        Intent intent = new Intent(context, MainWindow.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(intent);
    if (context instanceof Activity) {
        ((Activity) context).finish();
    }
    }
}

```

Файл SolanaManager.kt

```

package com.example.coldbitwallet.wallet.solana

import android.util.Base64
import android.util.Log
import org.sol4k.Keypair
import org.json.JSONObject
import okhttp3.MediaType.Companion.toMediaType
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody
import java.security.MessageDigest
import java.nio.ByteBuffer

object SolanaManager {
    private const val TAG = "Solana"
    private val client = OkHttpClient()
    private const val RPC_URL = "https://api.devnet.solana.com"
    // DERIVE KEY
    private fun derivePrivateKey(seed: ByteArray, index: Int):
ByteArray {
        val digest = MessageDigest.getInstance("SHA-256")
        val combined = seed + byteArrayOf(index.toByte())
        return digest.digest(combined).copyOfRange(0, 32)
    }
    @JvmStatic
    fun getAddress(seed: ByteArray, index: Int = 0): String {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						188
Изм.	Лист	№ докум.	Подпись	Дата		

```

        val keypair = Keypair.fromSecretKey(derivePrivateKey(seed,
index))
        val address = keypair.publicKey.toBase58()
        Log.d(TAG, "Generated address[$index]: $address")
        return address
    }
    // BALANCE
    @JvmStatic
    fun getBalance(address: String): Double {
        Log.d(TAG, "getBalance → $address")
        val json = ""
            {
                "jsonrpc":"2.0",
                "id":1,
                "method":"getBalance",
                "params":["$address"]
            }
        """.trimIndent()
        val body = RequestBody.create(
            "application/json".toMediaType(),
            json
        )
        val response = client.newCall(
            Request.Builder()
                .url(RPC_URL)
                .post(body)
                .build()
        ).execute()
        val raw = response.body?.string() ?: ""
        Log.d(TAG, "Balance RAW response: $raw")
        val obj = JSONObject(raw)
        val lamports = obj.getJSONObject("result").getLong("value")
        val sol = lamports / 1_000_000_000.0
        Log.d(TAG, "Balance parsed: $sol SOL")
        return sol
    }
    // SEND SOL
    @JvmStatic
    fun sendSol(seed: ByteArray, toAddress: String, amountSol:
Double): String {
        Log.d(TAG, "sendSol START → $amountSol SOL to $toAddress")
        val keypair = Keypair.fromSecretKey(derivePrivateKey(seed,
0))

        val fromPubkey = keypair.publicKey.toBase58()
        Log.d(TAG, "From address: $fromPubkey")
        val lamports = (amountSol * 1_000_000_000).toLong()
        Log.d(TAG, "Lamports: $lamports")
        val blockhash = getLatestBlockhash()
        Log.d(TAG, "Blockhash: $blockhash")
        val message = buildTransferMessage(
            fromPubkey,

```

```

        toAddress,
        lamports,
        blockhash
    )
    Log.d(TAG, "Message size: ${message.size} bytes")
    val signature = keypair.sign(message)
    Log.d(TAG, "Signature generated")
    val tx = message + signature
    val base64Tx = Base64.encodeToString(tx, Base64.NO_WRAP)
    Log.d(TAG, "TX base64 length: ${base64Tx.length}")
    val json = """
        {
            "jsonrpc": "2.0",
            "id": 1,
            "method": "sendTransaction",
            "params": [
                "$base64Tx",
                {
                    "encoding": "base64"
                }
            ]
        }
    """.trimIndent()
    val body = RequestBody.create(
        "application/json".toMediaType(),
        json
    )
    Log.d(TAG, "Sending transaction to RPC...")
    val response = client.newCall(
        Request.Builder()
            .url(RPC_URL)
            .post(body)
            .build()
    ).execute()
    val raw = response.body?.string() ?: ""
    Log.d(TAG, "RPC RESPONSE: $raw")
    val result = JSONObject(raw)
    if (result.has("error")) {
        val err =
result.getJSONObject("error").getString("message")
        Log.e(TAG, "TX ERROR: $err")
        throw Exception(err)
    }
    val txHash = result.getString("result")
    Log.d(TAG, "TX SUCCESS → $txHash")
    return txHash
}
// BLOCKHASH
private fun getLatestBlockhash(): String {
    Log.d(TAG, "getLatestBlockhash")
    val json = """

```

```

        {
            "jsonrpc": "2.0",
            "id": 1,
            "method": "getLatestBlockhash"
        }
        """.trimIndent()
        val response = client.newCall(
            Request.Builder()
                .url(RPC_URL)

.post(RequestBody.create("application/json".toMediaType(), json))
                .build()
        ).execute()
        val raw = response.body?.string() ?: ""
        Log.d(TAG, "Blockhash RAW: $raw")
        val obj = JSONObject(raw)
        val blockhash = obj
            .getJSONObject("result")
            .getJSONObject("value")
            .getString("blockhash")
        Log.d(TAG, "Blockhash parsed: $blockhash")
        return blockhash
    }
    // MESSAGE BUILDER
    private fun buildTransferMessage(
        from: String,
        to: String,
        lamports: Long,
        blockhash: String
    ): ByteArray {
        Log.d(TAG, "buildTransferMessage")
        val fromBytes = from.decodeBase58()
        val toBytes = to.decodeBase58()
        val buffer = ByteBuffer.allocate(1024)
        buffer.put(1)
        buffer.put(fromBytes)
        buffer.put(toBytes)
        buffer.putLong(lamports)
        buffer.put(blockhash.toByteArray())
        val msg = buffer.array().filter { it.toInt() != 0 }
    }.toByteArray()
        Log.d(TAG, "Message built size=${msg.size}")

        return msg
    }
    // BASE58
    private val ALPHABET =
        "123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

    private fun String.decodeBase58(): ByteArray {
        var num = 0L

```

```

        for (c in this) {
            num = num * 58 + ALPHABET.indexOf(c)
        }
        val bytes = ByteArray(32)
        for (i in 31 downTo 0) {
            bytes[i] = (num and 0xff).toByte()
            num = num shr 8
        }
        return bytes
    }
}

```

Файл SolanaTransactionChecker.java

```

package com.example.coldbitwallet;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class SolanaTransactionChecker {
    private static final String TAG = "SOLANA_CHECKER";
    private static final String RPC_URL =
"https://api.testnet.solana.com";
    // CHECK INCOMING SOL
    // =====
    public static void checkIncomingBalance(Context context, String
address, Callback callback) {
        new Thread(() -> {
            try {
                String request = "{\n" +
                    "  \"jsonrpc\": \"2.0\",\n" +
                    "  \"id\": 1,\n" +
                    "  \"method\": \"getBalance\",\n" +
                    "  \"params\": [\"" + address + "\"]\n" +
                    "}";
                HttpURLConnection conn = (HttpURLConnection)
                    new
URL("https://api.testnet.solana.com").openConnection();
                conn.setRequestMethod("POST");
                conn.setDoOutput(true);
                conn.setRequestProperty("Content-Type",
"application/json");
                conn.getOutputStream().write(request.getBytes());
                BufferedReader reader = new BufferedReader(

```

```

        new InputStreamReader(conn.getInputStream())
    );
    StringBuilder result = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
        result.append(line);
    }
    JSONObject json = new JSONObject(result.toString());
    long lamports =
json.getJSONObject("result").getLong("value");
    double sol = lamports / 1_000_000_000.0;
    double old =
context.getSharedPreferences("sol_balance", Context.MODE_PRIVATE)
        .getFloat(address, 0);
    if (sol > old) {
        TransactionHistoryManager.saveTransaction(
            context,
            "SOL",
            sol - old,
            "on-chain",
            false
        );
    }
    context.getSharedPreferences("sol_balance",
Context.MODE_PRIVATE)
        .edit()
        .putFloat(address, (float) sol)
        .apply();

    } catch (Exception e) {
        Log.e("SOLANA", "balance check error", e);
    }
    if (callback != null) {
        new Handler(Looper.getMainLooper())
            .post(callback::onFinished);
    }
    }).start();
}
// GET FULL TX
private static JSONObject getTransaction(String signature) {
    try {
        String request = String.format(
            "{\n" +
                "  \"jsonrpc\": \"2.0\", \n" +
                "  \"id\": 1, \n" +
                "  \"method\": \"getTransaction\", \n" +
                "  \"params\": [\n" +
                "    \"%s\", \n" +
                "    {\n" +
                "      \"encoding\": \"json\", \n"

```

```

        "
        \"maxSupportedTransactionVersion\":0\n" +
        "    }\n" +
        "  ]\n" +
        "}",
        signature
    );
    HttpURLConnection conn = (HttpURLConnection)
        new URL(RPC_URL).openConnection();
    conn.setRequestMethod("POST");
    conn.setDoOutput(true);
    conn.getOutputStream().write(request.getBytes());
    BufferedReader reader = new BufferedReader(
        new InputStreamReader(conn.getInputStream())
    );
    StringBuilder result = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
        result.append(line);
    }
    reader.close();
    return new JSONObject(result.toString());
} catch (Exception e) {
    Log.e(TAG, "GET TX ERROR", e);
    return null;
}
}
// CACHE
private static boolean isTxSaved(Context context, String txid) {
    SharedPreferences prefs =
        context.getSharedPreferences("sol_txs",
Context.MODE_PRIVATE);
    return prefs.contains(txid);
}
private static void saveTx(Context context, String txid) {
    SharedPreferences prefs =
        context.getSharedPreferences("sol_txs",
Context.MODE_PRIVATE);
    prefs.edit().putBoolean(txid, true).apply();
}
// CALLBACK
public interface Callback {
    void onFinish();
}
}

```

Файл SystemBarColorHelper.java

```

package com.example.coldbitwallet;

import android.app.Activity;
import android.graphics.Color;

```

```

import android.view.Window;
import android.view.WindowManager;
import androidx.core.content.ContextCompat;

public class SystemBarColorHelper {
    public static void applyColors(Activity activity, boolean
darkenStatusForSomeColors) {
        Window window = activity.getWindow();
window.clearFlags(WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS
);

window.addFlags(WindowManager.LayoutParams.FLAG_DRAWS_SYSTEM_BAR_BAC
KGROUND);
        int savedColor = ThemeManager.getSavedColor(activity);
        int paleGold = ContextCompat.getColor(activity,
R.color.PaleGold);
        int white = ContextCompat.getColor(activity,
R.color.TextColor);
        int Aqua = ContextCompat.getColor(activity,
R.color.Aquamarine);
        int darkenedColor = darkenColor(savedColor, 0.6f);
        // Статус-бар
        if (darkenStatusForSomeColors && (savedColor == paleGold ||
savedColor == white || savedColor == Aqua)) {
            window.setStatusBarColor(darkenedColor);
        } else {
            window.setStatusBarColor(savedColor);
        }
        window.setNavigationBarColor(darkenedColor);
    }
    public static int darkenColor(int color, float factor) {
        int r = Math.round(Color.red(color) * factor);
        int g = Math.round(Color.green(color) * factor);
        int b = Math.round(Color.blue(color) * factor);
        return Color.rgb(
            Math.min(r, 255),
            Math.min(g, 255),
            Math.min(b, 255)
        );
    }
}

```

Файл TextColorActivity.java

```

package com.example.coldbitwallet;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		195

```

import android.view.View;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class TextColorActivity extends AppCompatActivity {
    private View selectedTextView = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_text_color);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        TextColorManager.applySavedTextColor(this,
R.id.text_color_text);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
    return insets;
});
    int savedTextColor =
TextColorManager.getSavedTextColor(this);
highlightSelectedTextBlock(savedTextColor);
}
    private int getOriginalTextColor(int viewId) {
        if (viewId == R.id.blueTextBlock) {
            return ContextCompat.getColor(this, R.color.Blue);
        } else if (viewId == R.id.greenTextBlock) {
            return ContextCompat.getColor(this, R.color.Green);
        } else if (viewId == R.id.blackTextBlock) {
            return ContextCompat.getColor(this, R.color.Black);
        } else if (viewId == R.id.whiteTextBlock) {
            return ContextCompat.getColor(this, R.color.TextColor);
        } else if (viewId == R.id.pinkTextBlock) {
            return ContextCompat.getColor(this, R.color.Pink);
        } else if (viewId == R.id.purpleTextBlock) {
            return ContextCompat.getColor(this, R.color.Purple);
        } else if (viewId == R.id.grayTextBlock) {
            return ContextCompat.getColor(this, R.color.Gray);
        } else if (viewId == R.id.redTextBlock) {
            return ContextCompat.getColor(this, R.color.Red);
        }
    }
}

```

```

    } else if (viewId == R.id.OrangeTextBlock) {
        return ContextCompat.getColor(this, R.color.Orange);
    } else if (viewId == R.id.PaleTextBlock) {
        return ContextCompat.getColor(this, R.color.PaleGold);
    } else if (viewId == R.id.yellowTextBlock) {
        return ContextCompat.getColor(this, R.color.Yellow);
    } else if (viewId == R.id.brownTextBlock) {
        return ContextCompat.getColor(this, R.color.Brown);
    } else if (viewId == R.id.darkGrayTextBlock) {
        return ContextCompat.getColor(this, R.color.darkGray);
    } else if (viewId == R.id.AquamarineTextBlock) {
        return ContextCompat.getColor(this, R.color.Aquamarine);
    } else if (viewId == R.id.trueMainTextBlock) {
        return ContextCompat.getColor(this,
R.color.MainBackground);
    } else {
        return ContextCompat.getColor(this, R.color.TextColor);
    }
}

private void highlightSelectedTextBlock(int savedTextColor) {
    int[] textColorBlockIds = {
        R.id.brownTextBlock, R.id.blueTextBlock,
R.id.redTextBlock, R.id.greenTextBlock, R.id.purpleTextBlock,
        R.id.whiteTextBlock, R.id.grayTextBlock,
R.id.PaleTextBlock, R.id.OrangeTextBlock, R.id.blackTextBlock,
        R.id.pinkTextBlock, R.id.trueMainTextBlock,
R.id.yellowTextBlock, R.id.darkGrayTextBlock,
        R.id.AquamarineTextBlock
    };
    for (int id : textColorBlockIds) {
        View block = findViewById(id);
        if (block != null && getOriginalTextColor(id) ==
savedTextColor) {
            boolean isBlack = (id == R.id.blackTextBlock);
            selectTextColorBlock(block, isBlack);
            break;
        }
    }
}

public void ClickCancel(View view) {
    PressEffectHelper.apply(this, view, "sss");
    finish();
}

private void selectTextColorBlock(View block, boolean
isBlackBlock) {
    if (selectedTextView != null) {
        selectedTextView.setBackgroundColor(getOriginalTextColor(selectedTex
tView.getId())); // Сброс цвета
    }
    selectedTextView = block;
}

```

```

        GradientDrawable background = new GradientDrawable();
        background.setColor(getOriginalTextColor(block.getId()));
// Цвет блока
        background.setStroke(8, isBlackBlock ? Color.WHITE :
Color.BLACK); // Обводка
        block.setBackground(background);
    }
    private void applyTextColor(View view, int colorResId, boolean
isBlackTheme) {
        int normal = ContextCompat.getColor(this, colorResId);
        int savedThemeColor = ThemeManager.getSavedColor(this); //
Получаем текущий цвет темы
        int savedButtonColor =
ButtonsColorManager.getSavedButtonColor(this); // Получаем текущий
цвет кнопок
        if (normal == savedThemeColor) {
            Toast.makeText(this,
getString(R.string.same_background_and_text_warning),
Toast.LENGTH_SHORT).show();
            VibrationHelper.triggerVibration(this);
            return; // Не сохраняем цвет текста
        } else if (normal == savedButtonColor) {
            Toast.makeText(this,
getString(R.string.same_button_and_text_warning),
Toast.LENGTH_SHORT).show();
            VibrationHelper.triggerVibration(this);
            return; // Не сохраняем цвет текста
        }
        TextColorManager.saveTextColor(this, normal);
        selectTextColorBlock(view, isBlackTheme);
        returnToMainWindow(this);
    }
    public void ClickBlue(View view) {
        applyTextColor(view, R.color.Blue, false);
    }
    public void ClickGreen(View view) {
        applyTextColor(view, R.color.Green, false);
    }
    public void ClickBlack(View view) {
        applyTextColor(view, R.color.Black, true);
    }
    public void ClickWhite(View view) {
        applyTextColor(view, R.color.TextColor, false);
    }
    public void ClickPink(View view) {
        applyTextColor(view, R.color.Pink, false);
    }
    public void ClickPurple(View view) {
        applyTextColor(view, R.color.Purple, false);
    }
    public void ClickGray(View view) {

```

```

        applyTextColor(view, R.color.Gray, false);
    }
    public void ClickRed(View view) {
        applyTextColor(view, R.color.Red, false);
    }
    public void ClickOrange(View view) {
        applyTextColor(view, R.color.Orange, false);
    }
    public void ClickPale(View view) {
        applyTextColor(view, R.color.PaleGold, false);
    }
    public void ClickYellow(View view) {
        applyTextColor(view, R.color.Yellow, false);
    }
    public void ClickBrown(View view) {
        applyTextColor(view, R.color.Brown, false);
    }
    public void ClickDarkGray(View view) {
        applyTextColor(view, R.color.darkGray, false);
    }
    public void ClickMain(View view) {
        applyTextColor(view, R.color.MainBackground, false);
    }
    public void ClickAquamarine(View view) {
        applyTextColor(view, R.color.Aquamarine, false);
    }
    public void returnToMainWindow(Context context) {
        Intent intent = new Intent(context, MainWindow.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(intent);

        if (context instanceof Activity) {
            ((Activity) context).finish();
        }
    }
}

```

Файл TextColorManager.java

```

package com.example.coldbitwallet;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.core.content.ContextCompat;

public class TextColorManager {

```

```

        private static final String PREFS_NAME = "text_color_prefs";
        private static final String KEY_TEXT_COLOR =
"selected_text_color";

        public static void applySavedTextColor(Activity activity, int...
textViewIds) {
            int color = getSavedTextColor(activity);
            for (int id : textViewIds) {
                TextView textView = activity.findViewById(id);
                if (textView != null) {
                    textView.setTextColor(color);
                }
            }
        }

        public static void saveTextColor(Context context, int color) {
            SharedPreferences prefs =
context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
            prefs.edit().putInt(KEY_TEXT_COLOR, color).apply();
        }

        public static int getSavedTextColor(Context context) {
            SharedPreferences prefs =
context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
            if (prefs.contains(KEY_TEXT_COLOR)) {
                return prefs.getInt(KEY_TEXT_COLOR, Color.BLACK);
            } else {
                return ContextCompat.getColor(context,
R.color.TextColor);
            }
        }

        public static void applySavedTextColorToView(Context context,
View view) {
            int color = getSavedTextColor(context);
            if (view instanceof TextView) {
                ((TextView) view).setTextColor(color);
            }
            if (view instanceof ViewGroup) {
                ViewGroup group = (ViewGroup) view;
                for (int i = 0; i < group.getChildCount(); i++) {
                    applySavedTextColorToView(context,
group.getChildAt(i));
                }
            }
        }
    }
}

```

Файл ThemeManager.java

```

package com.example.coldbitwallet;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		200

```

import android.view.View;
import androidx.core.content.ContextCompat;

public class ThemeManager {
    private static final String PREFS_NAME = "theme_prefs";
    private static final String KEY_COLOR = "selected_color";
    public static void applySavedBackground(Activity activity, int
layoutId) {
        View rootLayout = activity.findViewById(layoutId);
        if (rootLayout != null) {
            int color = getSavedColor(activity);
            rootLayout.setBackgroundColor(color);
        }
    }
    // Для Dialog / BottomSheet / любого View
    public static void applySavedBackgroundToView(Context context,
View view) {
        if (view == null) return;
        int color = getSavedColor(context);
        view.setBackgroundColor(color);
    }
    public static void saveColor(Context context, int color) {
        SharedPreferences prefs =
context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
        prefs.edit().putInt(KEY_COLOR, color).apply();
    }
    public static int getSavedColor(Context context) {
        SharedPreferences prefs =
context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
        if (prefs.contains(KEY_COLOR)) {
            return prefs.getInt(KEY_COLOR, R.color.TextColor);
        } else {
            // Вернём реальный цвет из ресурсов по умолчанию
            return ContextCompat.getColor(context,
R.color.MainBackground);
        }
    }
}

```

Файл ThemesSetting.java

```

package com.example.coldbitwallet;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		201

```

import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class ThemesSetting extends AppCompatActivity {
    ConstraintLayout mainLayout;
    private View selectedView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_themes_setting);
        // Затемнение нижний и верхней панели
        SystemBarColorHelper.applyColors(this, true);
        // Применение сохранённого цвета текста
        TextColorManager.applySavedTextColor(this,
R.id.selectThemesText);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
    Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
    v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
    return insets;
});
    mainLayout = findViewById(R.id.main);
    // Применяем сохранённый цвет через ThemeManager
    int savedColor = ThemeManager.getSavedColor(this);
    mainLayout.setBackgroundColor(savedColor);
    highlightSelectedBlock(savedColor);
}
    private int getOriginalColor(int viewId) {
        if (viewId == R.id.blueBlock) {
            return ContextCompat.getColor(this, R.color.Blue);
        } else if (viewId == R.id.greenBlock) {
            return ContextCompat.getColor(this, R.color.Green);
        } else if (viewId == R.id.blackBlock) {
            return ContextCompat.getColor(this, R.color.Black);
        } else if (viewId == R.id.whiteBlock) {
            return ContextCompat.getColor(this, R.color.TextColor);
        } else if (viewId == R.id.pinkBlock) {
            return ContextCompat.getColor(this, R.color.Pink);
        } else if (viewId == R.id.purpleBlock) {
            return ContextCompat.getColor(this, R.color.Purple);
        } else if (viewId == R.id.grayBlock) {
            return ContextCompat.getColor(this, R.color.Gray);
        } else if (viewId == R.id.redBlock) {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		202

```

        return ContextCompat.getColor(this, R.color.Red);
    } else if (viewId == R.id.OrangeBlock) {
        return ContextCompat.getColor(this, R.color.Orange);
    } else if (viewId == R.id.PaleBlock) {
        return ContextCompat.getColor(this, R.color.PaleGold);
    } else if (viewId == R.id.yellowBlock) {
        return ContextCompat.getColor(this, R.color.Yellow);
    } else if (viewId == R.id.brownBlock) {
        return ContextCompat.getColor(this, R.color.Brown);
    } else if (viewId == R.id.darkGrayBlock) {
        return ContextCompat.getColor(this, R.color.darkGray);
    } else if (viewId == R.id.AquamarineBlock) {
        return ContextCompat.getColor(this, R.color.Aquamarine);
    } else if (viewId == R.id.trueMainBlock) {
        return ContextCompat.getColor(this,
R.color.MainBackground);
    } else {
        return ContextCompat.getColor(this,
R.color.MainBackground);
    }
}

private void selectColorBlock(View block, boolean isBlackBlock)
{
    // Снимаем выделение с предыдущего блока (если он был
выбран)
    if (selectedView != null) {

selectedView.setBackgroundColor(getOriginalColor(selectedView.getId(
))); // Сброс фона
    }
    // Сохраняем новый выбранный блок
    selectedView = block;
    // Устанавливаем цвет и обводку
    GradientDrawable background = new GradientDrawable();
    background.setColor(getOriginalColor(block.getId())); //
Устанавливаем цвет блока
    background.setStroke(8, isBlackBlock ? Color.WHITE :
Color.BLACK); // Устанавливаем обводку
    block.setBackground(background); // Применяем фоновую
обводку
}

private void highlightSelectedBlock(int savedColor) {
    int[] blockIds = {
        R.id.brownBlock, R.id.blueBlock, R.id.redBlock,
R.id.greenBlock, R.id.purpleBlock, R.id.whiteBlock,
        R.id.grayBlock, R.id.PaleBlock, R.id.OrangeBlock,
R.id.blackBlock, R.id.pinkBlock, R.id.trueMainBlock,
        R.id.yellowBlock, R.id.darkGrayBlock,
R.id.AquamarineBlock
    };
    for (int id : blockIds) {

```

```

        View block = findViewById(id);
        if (block != null && getOriginalColor(id) == savedColor)
        {
            boolean isBlack = (id == R.id.blackBlock);
            selectColorBlock(block, isBlack);
            break;
        }
    }

    private void applyThemeColor(View view, int colorResId, boolean
isBlackTheme) {
        int normal = ContextCompat.getColor(this, colorResId);
        int savedTextColor =
TextColorManager.getSavedTextColor(this); // Получаем сохранённый
цвет текста
        int savedButtonColor =
ButtonsColorManager.getSavedButtonColor(this); // Получаем текущий
цвет кнопок
        if (normal == savedTextColor) {
            Toast.makeText(this,
getString(R.string.same_text_and_background_warning),
Toast.LENGTH_SHORT).show();
            VibrationHelper.triggerVibration(this);
            return;
        } else if (normal == savedButtonColor) {
            Toast.makeText(this,
getString(R.string.same_button_and_background_warning),
Toast.LENGTH_SHORT).show();
            VibrationHelper.triggerVibration(this);
            return;
        }
        mainLayout.setBackgroundColor(normal);
        ThemeManager.saveColor(this, normal);
        selectColorBlock(view, isBlackTheme);
        returnToMainWindow(this);
    }

    public void ClickCancel(View view) {
        PressEffectHelper.apply(this, view, "sss");
        finish();
    }

    public void ClickBlue(View view) {
        applyThemeColor(view, R.color.Blue, false);
    }

    public void ClickGreen(View view) {
        applyThemeColor(view, R.color.Green, false);
    }

    public void ClickBlack(View view) {
        applyThemeColor(view, R.color.Black, true);
    }

    public void ClickWhite(View view) {
        applyThemeColor(view, R.color.TextColor, false);
    }

```

```

    }
    public void ClickPink(View view) {
        applyThemeColor(view, R.color.Pink, false);
    }
    public void ClickPurple(View view) {
        applyThemeColor(view, R.color.Purple, false);
    }
    public void ClickGray(View view) {
        applyThemeColor(view, R.color.Gray, false);
    }
    public void ClickRed(View view) {
        applyThemeColor(view, R.color.Red, false);
    }
    public void ClickOrange(View view) {
        applyThemeColor(view, R.color.Orange, false);
    }
    public void ClickPale(View view) {
        applyThemeColor(view, R.color.PaleGold, false);
    }
    public void ClickYellow(View view) {
        applyThemeColor(view, R.color.Yellow, false);
    }
    public void ClickBrown(View view) {
        applyThemeColor(view, R.color.Brown, false);
    }
    public void ClickDarkGray(View view) {
        applyThemeColor(view, R.color.darkGray, false);
    }
    public void ClickAquamarine(View view) {
        applyThemeColor(view, R.color.Aquamarine, false);
    }
    public void ClickMain(View view) {
        applyThemeColor(view, R.color.MainBackground, false);
    }
    public void returnToMainWindow(Context context) {
        Intent intent = new Intent(context, MainWindow.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(intent);
        if (context instanceof Activity) {
            ((Activity) context).finish();
        }
    }
}

```

Файл TonManager.kt

```

package com.example.coldbitwallet.wallet.ton

import org.ton.api.pk.PrivateKeyEd25519
import org.ton.contract.wallet.WalletV4R2Contract

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		205

```

object TonManager {

    @JvmStatic
    fun getAddress(seed: ByteArray, bounceable: Boolean = true,
testnet: Boolean = false): String {
        // Первые 32 байта - приватный ключ
        val privateKey = PrivateKeyEd25519(seed.copyOfRange(0, 32))
        val walletId = 0
        val address = WalletV4R2Contract.address(privateKey,
walletId)
        return address.toString(
userFriendly = true,
            urlSafe = true,
            bounceable = bounceable,
            testOnly = testnet
        )
    }
}

```

Файл TransactionAmount.java

```

package com.example.coldbitwallet;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Build;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Base64;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.example.coldbitwallet.wallet.HDWalletManager;
import com.example.coldbitwallet.wallet.bitcoin.BitcoinManager;
import com.example.coldbitwallet.wallet.ethereum.EthereumManager;
import com.google.android.material.bottomsheet.BottomSheetBehavior;
import com.google.android.material.bottomsheet.BottomSheetDialog;
import org.jetbrains.annotations.NotNull;
import org.web3j.utils.Convert;
import java.math.BigDecimal;

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		206

```

import java.math.BigInteger;
import java.security.KeyStore;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.util.List;
import java.util.Locale;
import java.util.concurrent.CompletableFuture;
import javax.crypto.Cipher;

public class TransactionAmount extends AppCompatActivity {

    private static final String TAG = "TransactionAmount";
    private String scanResult;
    private String selectedToken;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        String lang = getSharedPreferences("settings", MODE_PRIVATE)
            .getString("language", "system");
        LocaleHelper.setLocale(this, lang);
        setContentView(R.layout.activity_transaction_amount);
        SystemBarColorHelper.applyColors(this, true);
        ThemeManager.applySavedBackground(this, R.id.main);
        ThemeManager.applySavedBackground(this, R.id.mainTX);
        int[] textViewIds = {
R.id.NextButtonSendByScanner, R.id.txSum, R.id.AvailableTokens,
R.id.txHash, R.id.txTitle
        };
        for (int id : textViewIds) {
            TextColorManager.applySavedTextColor(this, id);
        }
        Button nextButton =
findViewById(R.id.NextButtonSendByScanner);
        PressEffectHelper.apply(this, nextButton, "spec");

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
(v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
                systemBars.right, systemBars.bottom);
            return insets;
        });
        scanResult = getIntent().getStringExtra("SCAN_RESULT");
        selectedToken =
getIntent().getStringExtra("selected_token");
        TextView availableText = findViewById(R.id.AvailableTokens);
        EditText amountInput = findViewById(R.id.TransactionAmount);
        availableText.setOnClickListener(v -> {

```

```

        double balance = PriceManager.getBalance(this,
selectedToken.toLowerCase());
        DecimalFormat df = new DecimalFormat("0.#####",
            new DecimalFormatSymbols(Locale.US));
        String formatted = df.format(balance);
        amountInput.setText(formatted);
        amountInput.setSelection(formatted.length());
    });
    updateAvailableBalance();
}
@Override
protected void onResume() {
    super.onResume();
    selectedToken =
getIntent().getStringExtra("selected_token");
    TextView tokenText = findViewById(R.id.TokenText);
    ConstraintLayout layout =
findViewById(R.id.constraintLayout3);
    if (selectedToken != null) {
        tokenText.setText(selectedToken);
        int colorRes = R.color.MainBackground;
        switch (selectedToken.toLowerCase()) {
            case "btc": colorRes = R.color.BTCColor; break;
            case "eth": colorRes = R.color.ETHColor; break;
            case "sol": colorRes = R.color.SOLColor; break;
            case "ton": colorRes = R.color.TONColor; break;
            case "usdt":
                colorRes = R.color.USDTColor;
                break;
        }
        int color = ContextCompat.getColor(this, colorRes);
        if (layout.getBackground() instanceof
android.graphics.drawable.GradientDrawable) {
            ((android.graphics.drawable.GradientDrawable)
layout.getBackground())
                .setColor(color);
        }
    }
    // ОБНОВЛЕНИЕ БАЛАНСА
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        CompletableFuture.runAsync(() -> {
            try {
                String ethAddress =
BalanceManager.getInstance().getAddress("eth");
                String btcAddress =
BalanceManager.getInstance().getAddress("btc");
                // ETH баланс
                if (ethAddress != null && !ethAddress.isEmpty())
            {

```

```

        BigDecimal ethBalance =
EthereumManager.INSTANCE.getEthBalance(
            ethAddress,
            EthereumManager.EthNetwork.SEPOLIA
        );
        BalanceManager.getInstance().setBalance(
            "eth",
            ethBalance.doubleValue()
        );
    }
    // BTC
    if (btcAddress != null && !btcAddress.isEmpty())
    {
        // пока оставляем как есть или через
        BitcoinManager
        {
            runOnUiThread(this::updateAvailableBalance);
        } catch (Exception e) {
            Log.e(TAG, "Balance update error", e);
        }
    }
    // Input limiter
    EditText amountInput = findViewById(R.id.TransactionAmount);
    amountInput.addTextChangedListener(new TextWatcher() {
        public void beforeTextChanged(CharSequence s, int st,
int c, int a) {}
        public void onTextChanged(CharSequence s, int st, int b,
int c) {}
        public void afterTextChanged(Editable s) {
            String input = s.toString();
            if (input.contains(".")) {
                int i = input.indexOf(".");
                if (input.length() - i - 1 > 6) {
                    s.delete(i + 7, input.length());
                }
            }
        }
    });
}
private void updateAvailableBalance() {
    if (selectedToken == null) return;
    double balance = PriceManager.getBalance(this,
selectedToken.toLowerCase());
    DecimalFormat df = new DecimalFormat("0.#####",
        new DecimalFormatSymbols(Locale.US));
    String text = getString(R.string.availableToken,
        df.format(balance), selectedToken);
    ((TextView)
findViewById(R.id.AvailableTokens)).setText(text);
}

```

```

private void performSend() {
    EditText amountInput = findViewById(R.id.TransactionAmount);
    TextView tokenText = findViewById(R.id.TokenText);
    String token = tokenText.getText().toString().trim();
    String amountStr = amountInput.getText().toString().trim();
    if (token.isEmpty() || amountStr.isEmpty()) {
        Toast.makeText(this,
getString(R.string.toast_sum_transaction),
Toast.LENGTH_SHORT).show();
        return;
    }
    double amount;
    try {
        amount = Double.parseDouble(amountStr);
    } catch (Exception e) {
        Toast.makeText(this,
getString(R.string.toast_incorrect_format),
Toast.LENGTH_SHORT).show();
        return;
    }

    String recipient = (scanResult != null) ? scanResult.trim()
: "";
    if (recipient.isEmpty()) {
        Toast.makeText(this, "Адрес не найден",
Toast.LENGTH_SHORT).show();
        return;
    }
    // КЛЮЧЕВОЕ
    refreshBalanceAndSend(token, amount, recipient);
}

private void refreshBalanceAndSend(String token, double
amount, String recipient) {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        CompletableFuture.runAsync(() -> {
            try {
                List<String> mnemonic =
getMnemonicFromStorage();

                HDWalletManager.HDWalletMaster master =

HDWalletManager.createMasterFromMnemonic(mnemonic, "");
                String ethAddress =
EthereumManager.INSTANCE.getAddress(master, 0);
                String btcAddress =
BitcoinManager.INSTANCE.getAddress(mnemonic, "", 0);
                double balance = 0;
                switch (token.toLowerCase()) {
                    case "eth": {
                        BigDecimal ethBalance =
EthereumManager.INSTANCE.getEthBalance(
ethAddress,

```

```

EthereumManager.EthNetwork.SEPOLIA
    );
    balance = ethBalance.doubleValue();

BalanceManager.getInstance().setBalance("eth", balance);
    break;
}
case "usdt": {
    BigInteger raw =
EthereumManager.INSTANCE.getErc20Balance(
        ethAddress,

"0xCaC524BcA292aaade2DF8A05cC58F0a65B1B3bB9",

EthereumManager.EthNetwork.SEPOLIA
    );
    balance = new BigDecimal(raw)
        .divide(new
BigDecimal("1000000"))
        .doubleValue();

BalanceManager.getInstance().setBalance("usdt", balance);
    break;
}
case "btc": {
    BigDecimal btcBalance =
BitcoinManager.INSTANCE.getBtcBalance(btcAddress);
    balance = btcBalance.doubleValue();

BalanceManager.getInstance().setBalance("btc", balance);
    break;
}
default:
    balance =
BalanceManager.getInstance().getBalance(token);
    break;
}
double finalBalance = balance;
runOnUiThread(() -> {
    if (amount <= 0 || amount > finalBalance) {
Toast.makeText(this,

getString(R.string.toast_insufficient_funds),
        Toast.LENGTH_SHORT).show();
        return;
    }
    // отправка
    switch (token.toLowerCase()) {
        case "eth":
            sendEth(recipient, amount);

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		211

```

        break;
        case "usdt":
            sendPyusd(recipient, amount);
            break;
        case "btc":
            sendBtc(recipient, amount);
            break;
        case "sol":
            break;
        case "ton":
            break;
    }
    });
} catch (Exception e) {
    runOnUiThread(() ->
        Toast.makeText(this,
e.getMessage(),
            "Ошибка получения баланса: " +
            Toast.LENGTH_LONG).show()
        );
    }

});
}

private void sendBtc(String recipient, double amountBtc) {
    long amountSatoshis = (long) (amountBtc * 1e8);
    Log.d(TAG, "sendBtc called: recipient=" + recipient +
        ", amountSatoshis=" + amountSatoshis);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        CompletableFuture.supplyAsync(() -> {
            try {
                List<String> mnemonic =
getMnemonicFromStorage();
                Log.d(TAG, "Mnemonic loaded, size=" +
mnemonic.size());
                String txid =
BitcoinManager.INSTANCE.sendBitcoin(
                    mnemonic,
                    recipient,
                    amountSatoshis,
                    "",
                    0
                );
                Log.d(TAG, "sendBitcoin success, txid=" + txid);
                return txid;
            } catch (Exception e) {
                Log.e(TAG, "Error in sendBtc supplyAsync", e);
                throw new RuntimeException(e);
            }
        }).thenAccept(txid -> runOnUiThread(() -> {

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		212

```

        showTxSuccessSheet("BTC", txid); // Окно с хэшем
транзакции
        // История
        saveTransactionToHistory( "Отправлено", recipient,
"BTC", amountBtc );
    })).exceptionally(ex -> {
        Log.e(TAG, "sendBtc exceptionally caught", ex);
        runOnUiThread() ->
            Toast.makeText(this, "Ошибка BTC: " +
ex.getMessage(), Toast.LENGTH_LONG).show() );
return null;
    });
    } else {
        Toast.makeText(this, "API level too low",
Toast.LENGTH_SHORT).show();
    }
}
private void openHistoryScreen() {
    Intent intent = new Intent(TransactionAmount.this,
HistoryAction.class);
    startActivity(intent);
    finish(); // закрываем текущую активность
}
private void saveTransactionToHistory(String type, String
address, String token, double amount) {
    TransactionHistoryManager.saveTransaction(
        this,
        token,
        amount,
        address,
        type.equals("Отправлено")
    );
}
private List<String> getMnemonicFromStorage() throws Exception {
    Log.d(TAG, "getMnemonicFromStorage: start");
    SharedPreferences prefs = getSharedPreferences("MyPrefs",
MODE_PRIVATE);
    String encryptedSeedB64 = prefs.getString("encrypted_seed",
null);
    String encryptedAesKeyB64 =
prefs.getString("encrypted_aes_key", null);
    if (encryptedSeedB64 == null || encryptedAesKeyB64 == null)
    {
        throw new Exception("Missing encrypted seed or key");
    }
    byte[] encryptedSeed = Base64.decode(encryptedSeedB64,
Base64.DEFAULT);
    byte[] encryptedAesKey = Base64.decode(encryptedAesKeyB64,
Base64.DEFAULT);
    KeyStore keyStore = KeyStore.getInstance("AndroidKeyStore");
    keyStore.load(null);

```

```

        java.security.PrivateKey privateKey =
            (java.security.PrivateKey)
keyStore.getKey("seed_rsa_key", null);
        if (privateKey == null) {
            throw new Exception("Private key not found in
KeyStore");
        }
        Cipher rsaCipher =
Cipher.getInstance("RSA/ECB/PKCS1Padding");
        rsaCipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] aesKey = rsaCipher.doFinal(encryptedAesKey);
        // Используем CryptoHelper для расшифровки мнемоники
        String mnemonic = CryptoHelper.decryptWithAes(encryptedSeed,
aesKey);
        List<String> words =
java.util.Arrays.asList(mnemonic.split(" "));
        Log.d(TAG, "Mnemonic words count=" + words.size());
        return words;
    }

    private void sendEth(String recipient, double amountEth) {
        BigInteger amountWei = Convert.toWei(
            BigDecimal.valueOf(amountEth),
            Convert.Unit.ETHER
        ).toBigInteger();
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            CompletableFuture.supplyAsync(() -> {
                try {
                    List<String> mnemonic =
getMnemonicFromStorage();
                    HDWalletManager.HDWalletMaster master =

HDWalletManager.createMasterFromMnemonic(mnemonic, "");
                    String fromAddress =

EthereumManager.INSTANCE.getAddress(master, 0);
                    Log.d(TAG, "FROM: " + fromAddress);
                    return EthereumManager.INSTANCE.sendEthereum(
master,

                        recipient,
                        amountWei,
                        EthereumManager.EthNetwork.SEPOLIA,
                        0
                    );
                } catch (Exception e) {
                    throw new RuntimeException(e);
                }
            }).thenAccept(txHash -> runOnUiThread(() -> {
                showTxSuccessSheet("ETH", txHash);
                // ОБНОВЛЕНИЕ БАЛАНСА ПРАВИЛЬНО
                CompletableFuture.runAsync(() -> {
                    try {

```

```

        String address =
BalanceManager.getInstance().getAddress("eth");
        BigDecimal balance =
EthereumManager.INSTANCE.getEthBalance(
            address,
            EthereumManager.EthNetwork.SEPOLIA
        );
        double newBalance = balance.doubleValue();
        // 1. RAM кеш

BalanceManager.getInstance().setBalance("eth", newBalance);
        // 2. SharedPreferences кеш
        getSharedPreferences("MyPrefs",
MODE_PRIVATE)
            .edit()
            .putString("eth_balance",
String.valueOf(newBalance))
            .apply();
        } catch (Exception e) {
            Log.e(TAG, "Balance update error", e);
        }
    });
    saveTransactionToHistory(
        "Отправлено",
        recipient,
        "ETH",
        amountEth
    );
    })).exceptionally(ex -> {
        runOnUiThread(() ->
            Toast.makeText(this,
                "Ошибка: " + ex.getMessage(),
                Toast.LENGTH_LONG).show()
        );
        return null;
    });
}

private void sendPyusd(String recipient, double amountPyusd) {
    Log.d("PYUSD_SEND", "Recipient = " + recipient);
    Log.d("PYUSD_SEND", "Amount (input double) = " +
amountPyusd);
    BigDecimal amount = BigDecimal.valueOf(amountPyusd);
    Log.d("PYUSD_SEND", "Amount BigDecimal = " + amount);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        CompletableFuture.supplyAsync(() -> {
            try {
                List<String> mnemonic =
getMnemonicFromStorage();
                Log.d("PYUSD_SEND", "Mnemonic loaded: " +
(mnemonic != null ? mnemonic.size() : "null"));

```

```

        HDWalletManager.HDWalletMaster master =
HDWalletManager.createMasterFromMnemonic(mnemonic, "");
        String address =
BalanceManager.getInstance().getAddress("eth");
        Log.d("PYUSD_SEND", "FROM address = " +
address);
        BigDecimal beforeBalance =
EthereumManager.INSTANCE.getUsdtBalance(
            address,
            EthereumManager.EthNetwork.SEPOLIA
        );
        Log.d("PYUSD_SEND", "USDT BEFORE SEND = " +
beforeBalance);
        @NotNull String tx =
EthereumManager.INSTANCE.sendPYUSD(
            master,
            recipient,
            amount,
            EthereumManager.EthNetwork.SEPOLIA,
            0
        );
        Log.d("PYUSD_SEND", "TX HASH = " + tx);
        return tx;
    } catch (Exception e) {
        Log.e("PYUSD_SEND", "ERROR IN SEND", e);
        throw new RuntimeException(e);
    }
}).thenAccept(txHash -> runOnUiThread(() -> {
    Log.d("PYUSD_SEND", "TX CONFIRMED = " + txHash);
    showTxSuccessSheet("USDT", txHash);
    CompletableFuture.runAsync(() -> {
        try {
            String address =
BalanceManager.getInstance().getAddress("eth");
            Log.d("PYUSD_SEND", "Checking balance after
send...");
            BigDecimal balance =
EthereumManager.INSTANCE.getUsdtBalance(
                address,
                EthereumManager.EthNetwork.SEPOLIA
            );
            Log.d("PYUSD_SEND", "USDT AFTER SEND RAW = "
+ balance);
            Log.d("PYUSD_SEND", "USDT AFTER SEND DOUBLE
= " + balance.doubleValue());
            double newBalance = balance.doubleValue();
            // RAM
BalanceManager.getInstance().setBalance("usdt", newBalance);
            // PREFS

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						216
Изм.	Лист	№ докум.	Подпись	Дата		

```

        getSharedPreferences("MyPrefs",
MODE_PRIVATE)

        .edit()
        .putString("usdt_balance",
String.valueOf(newBalance))
        .apply();
        Log.d("PYUSD_SEND", "CACHE UPDATED OK = " +
newBalance);
    } catch (Exception e) {
        Log.e("PYUSD_SEND", "BALANCE UPDATE ERROR",
e);
    }
});
saveTransactionToHistory(
    "Отправлено",
    recipient,
    "PYUSD",
    amountPyusd
);
}).exceptionally(ex -> {
    Log.e("PYUSD_SEND", "FAILED COMPLETELY", ex);
    runOnUiThread() ->
        Toast.makeText(this,
            "Ошибка PYUSD: " + ex.getMessage(),
            Toast.LENGTH_LONG).show()
        );
    return null;
});
}

private void showTxSuccessSheet(String token, String txHash) {
    BottomSheetDialog dialog = new BottomSheetDialog(this);
    View view =
getLayoutInflater().inflate(R.layout.dialog_tx_success, null);
    TextView title = view.findViewById(R.id.txTitle);
    TextView hash = view.findViewById(R.id.txHash);
    Button btnOk = view.findViewById(R.id.btnOk);
    Button btnCopy = view.findViewById(R.id.btnCopy);
    dialog.setContentView(view);
    ThemeManager.applySavedBackgroundToView(this,
view.findViewById(R.id.mainTX));
    TextColorManager.applySavedTextColorToView(this, view);
    PressEffectHelper.apply(this, btnOk, "spec");
    PressEffectHelper.apply(this, btnCopy, "spec");
    title.setText(token + " " +
getString(R.string.sent_successfully));
    hash.setText(txHash);
    btnOk.setOnClickListener(v -> {
        dialog.dismiss();
        openHistoryScreen();
    });
}

```

```

        btnCopy.setOnClickListener(v -> {
            ClipboardManager clipboard =
                (ClipboardManager)
getSystemService(CLIPBOARD_SERVICE);
            ClipData clip = ClipData.newPlainText("txHash", txHash);
            clipboard.setPrimaryClip(clip);
        });
        dialog.setContentView(view);
        View bottomSheet =
dialog.findViewById(com.google.android.material.R.id.design_bottom_s
heet);
        if (bottomSheet != null) {
            BottomSheetBehavior<View> behavior =
                BottomSheetBehavior.from(bottomSheet);
            behavior.setPeekHeight(
                getResources().getDisplayMetrics().heightPixels
/ 2
            );
            behavior.setState(BottomSheetBehavior.STATE_EXPANDED);
        }
        dialog.show();
    }
    public void ClickSend(View view) {
        PressEffectHelper.apply(this, view, "spec");
        performSend();
    }
    public void ClickCancel(View view) {
        finish();
    }
}

```

Файл TransactionHistroeyManager.java

```

package com.example.coldbitwallet;

import android.content.Context;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class TransactionHistoryManager {
    private static final String FILE_NAME =
"transaction_history.txt";
    // Метод для записи транзакции в файл
    public static void saveTransaction(Context context, String
selectedToken, double amount, String address, boolean isSent) {
        String formattedDate = new SimpleDateFormat("dd.MM.yyyy
HH:mm:ss", Locale.getDefault()).format(new Date());

```

```

        String direction = isSent ? "Отправлено" : "Получено";
        String transactionRecord = direction + " " + address + "\n"
            + "Token: " + selectedToken + ", Amount: " + amount
            + ", Date: " + formattedDate + "\n\n";
        try {
            File file = new File(context.getExternalFilesDir(null),
FILE_NAME);
            FileOutputStream fos = new FileOutputStream(file, true);
// true – добавляем в конец
            OutputStreamWriter writer = new OutputStreamWriter(fos);
            writer.append(transactionRecord);
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // Метод для получения пути к файлу с историей
    public static File getHistoryFile(Context context) {
        return new File(context.getExternalFilesDir(null),
FILE_NAME);
    }
    // Метод для удаления файла с историей транзакций
    public static void clearTransactionHistory(Context context) {
        File file = new File(context.getExternalFilesDir(null),
FILE_NAME);
        if (file.exists()) {
            file.delete();
        }
    }
}

```

Файл VibrationHelper.java

```

package com.example.coldbitwallet;

import android.content.Context;
import android.os.Build;
import android.os.VibrationEffect;
import android.os.Vibrator;

public class VibrationHelper {

    public static void triggerVibration(Context context) {
        Vibrator vibrator = (Vibrator)
context.getSystemService(Context.VIBRATOR_SERVICE);
        if (vibrator != null) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                vibrator.vibrate(VibrationEffect.createOneShot(100,
VibrationEffect.DEFAULT_AMPLITUDE));
            } else {
                vibrator.vibrate(100);
            }
        }
    }
}

```

```

    }
}
}

```

Файл ServerForBetus.java

```

package com.mycompany.sereverforbetus;

import com.sun.net.httpserver.HttpServer;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpHandler;
import org.json.JSONObject;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.URI;
import java.net.http.*;
import java.time.Duration;
import java.util.concurrent.*;
import java.util.concurrent.atomic.AtomicReference;

public class SereverForBetus {
    private static final int PORT = 7000;
    private static final HttpClient client = HttpClient.newBuilder()
        .connectTimeout(Duration.ofSeconds(10))
        .build();

    // Только цены
    private static final AtomicReference<Double> BTC = new
AtomicReference<>(0.0);
    private static final AtomicReference<Double> ETH = new
AtomicReference<>(0.0);
    private static final AtomicReference<Double> TON = new
AtomicReference<>(0.0);
    private static final AtomicReference<Double> SOL = new
AtomicReference<>(0.0);
    public static void main(String[] args) throws Exception {
        // обновляем цены каждые 10 сек
        ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
        scheduler.scheduleAtFixedRate(SereverForBetus::updatePrices,
0, 10, TimeUnit.SECONDS);
        HttpServer server = HttpServer.create(new
InetSocketAddress("0.0.0.0", PORT), 0);
        // единственный endpoint
        server.createContext("/api/prices", new PriceHandler());
        server.setExecutor(Executors.newCachedThreadPool());
server.start();
        System.out.println("Price server started on port " + PORT);
    }
    // HANDLER
    static class PriceHandler implements HttpHandler {
        @Override

```

```

        public void handle(HttpExchange exchange) throws IOException
        {
            if
            (!exchange.getRequestMethod().equalsIgnoreCase("GET")) {
                exchange.sendResponseHeaders(405, -1);
                return;
            }
            JSONObject json = new JSONObject();
            json.put("BTC", BTC.get());
            json.put("ETH", ETH.get());
            json.put("TON", TON.get());
            json.put("SOL", SOL.get());

            sendJson(exchange, json);
        }
    }
    // PRICE UPDATE
    private static void updatePrices() {
        try {
            update("BTC",
                "https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT");
            update("ETH",
                "https://api.binance.com/api/v3/ticker/price?symbol=ETHUSDT");
            update("TON",
                "https://api.binance.com/api/v3/ticker/price?symbol=TONUSDT");
            update("SOL",
                "https://api.binance.com/api/v3/ticker/price?symbol=SOLUSDT");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private static void update(String symbol, String url) throws
    Exception {
        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .GET()
            .timeout(Duration.ofSeconds(10))
            .build();
        HttpResponse<String> response =
            client.send(request,
                HttpResponse.BodyHandlers.ofString());
        JSONObject json = new JSONObject(response.body());
        double price = json.getDouble("price");
        switch (symbol) {
            case "BTC" -> BTC.set(price);
            case "ETH" -> ETH.set(price);
            case "TON" -> TON.set(price);
            case "SOL" -> SOL.set(price);
        }
        System.out.println(symbol + " = " + price);
    }
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
						221
Изм.	Лист	№ докум.	Подпись	Дата		

```

// UTIL
private static void sendJson(HttpExchange exchange, JSONObject
json) throws IOException {
    byte[] response = json.toString().getBytes();
    exchange.getResponseHeaders().add("Content-Type",
"application/json");
    exchange.sendResponseHeaders(200, response.length);
    try (OutputStream os = exchange.getResponseBody()) {
        os.write(response);
    }
}
}

```

Файл Parsing.java

```

server.start();
    System.out.println("Price server started on port " + PORT);
}
// HANDLER
static class PriceHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange exchange) throws IOException
{
        if
(!exchange.getRequestMethod().equalsIgnoreCase("GET")) {
            exchange.sendResponseHeaders(405, -1);
            return;
        }
        JSONObject json = new JSONObject();
        json.put("BTC", BTC.get());
        json.put("ETH", ETH.get());
        json.put("TON", TON.get());
        json.put("SOL", SOL.get());

        sendJson(exchange, json);
    }
}
// PRICE UPDATE
private static void updatePrices() {
    try {
        update("BTC",
"https://api.binance.com/api/v3/ticker/price?symbol=BTCUSDT");
        update("ETH",
"https://api.binance.com/api/v3/ticker/price?symbol=ETHUSDT");
        update("TON",
"https://api.binance.com/api/v3/ticker/price?symbol=TONUSDT");
        update("SOL",
"https://api.binance.com/api/v3/ticker/price?symbol=SOLUSDT");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

					ПГУ 1.090301.29.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		222

```

        private static void update(String symbol, String url) throws
Exception {
            HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(url))
                .GET()
                .timeout(Duration.ofSeconds(10))
                .build();
            HttpResponse<String> response =
                client.send(request,
HttpResponse.BodyHandlers.ofString());
            JSONObject json = new JSONObject(response.body());
            double price = json.getDouble("price");
            switch (symbol) {
                case "BTC" -> BTC.set(price);
                case "ETH" -> ETH.set(price);
                case "TON" -> TON.set(price);
                case "SOL" -> SOL.set(price);
            }
            System.out.println(symbol + " = " + price);
        }
        // UTIL
        private static void sendJson(HttpExchange exchange, JSONObject
json) throws IOException {
            byte[] response = json.toString().getBytes();
            exchange.getResponseHeaders().add("Content-Type",
"application/json");
            exchange.sendResponseHeaders(200, response.length);
            try (OutputStream os = exchange.getResponseBody()) {
                os.write(response);
            }
        }
    }
}

```

Приложение В

(обязательное)

Разработка мультивалютного криптокошелька для платформы Android с использованием современных методов защиты криптографических ключей

Презентация

					ПГУ 1.090301.29.001 ПЗ	Лист
						224
Изм.	Лист	№ докум.	Подпись	Дата		

Разработка мультивалютного криптокошелька для платформы Android с использованием современных методов защиты криптографических ключей

Автор работы:

студент гр. 22ВВП2 Байков Алексей Владимирович

Научный руководитель:

к.т.н., доцент кф. ВТ Токарев Андрей Николаевич

Пенза 2026

Цель работы и объекты

Цель работы – разработать приложение, которое будет соответствовать стандартам безопасности и мультивалютности в формате некастодиального криптокошелька с возможностью осуществления исходящих транзакций, отслеживания поступлений и гибкой настройкой пользовательского интерфейса.

Объектами разработки являются:

- мобильное приложение для ОС Android, реализующие управление криптовалютными активами (просмотр баланса, история транзакций, отправка и получение средств), а также взаимодействие с блокчейн-сетями через API-запросы;
- криптографический модуль, обеспечивающий некастодиальное хранение приватных ключей: генерацию мнемонической фразы, создание и защиту ключей с использованием симметричного шифрования AES-256 в режиме CBC с PKCS5Padding и хранение зашифрованных данных.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.29.001 ПЗ

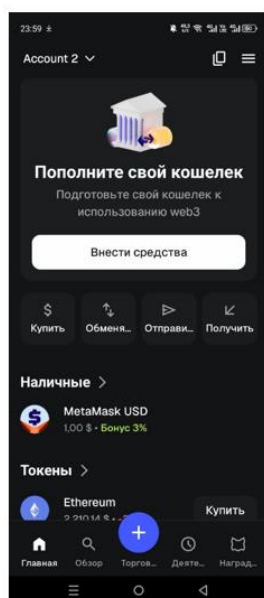
Лист

225

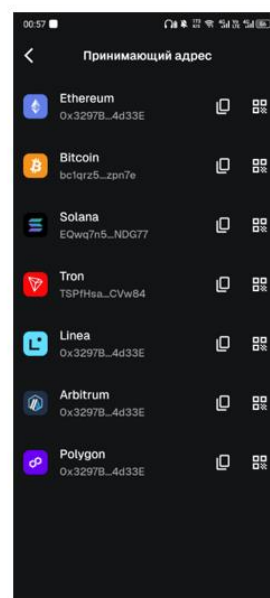
Актуальность

В условиях цифровизации мировой экономики криптовалюты стали неотъемлемой частью личных финансов миллионов людей. Биткоин, Эфириум и стейблкоины превратились в полноценный класс инвестиционных и расчетных средств: число пользователей в 2026 году превысило 700 млн, при этом более двух третей операций совершается с Android-устройств. В отличие от традиционных банков, криптовалютная экосистема основана на принципе некастодиальности — владелец самостоятельно несёт ответственность за хранение приватных ключей. Компрометация или утрата ключа влечёт безвозвратную потерю средств без возможности апелляции к третьей стороне.

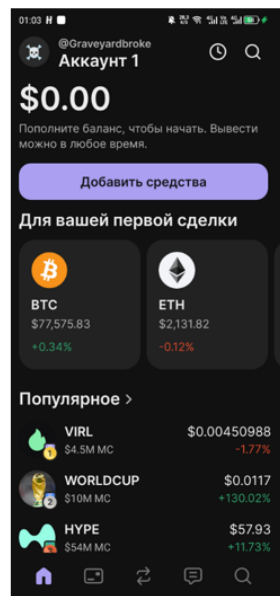
Аналоги



«MetaMask»



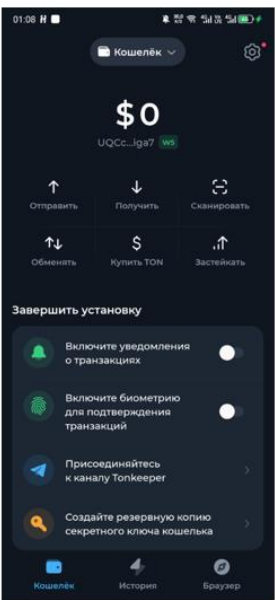
Аналоги



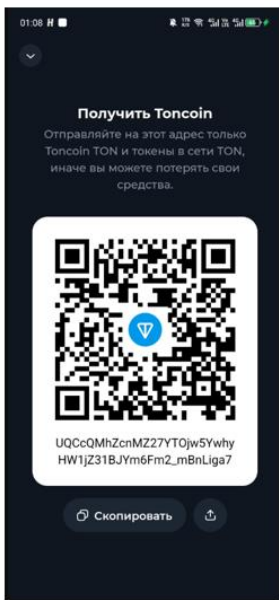
«Phantom»



Аналоги



«Tonkeeper»



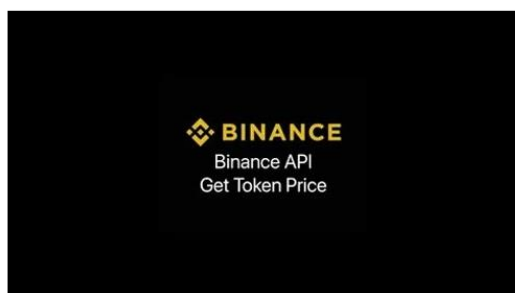
Постановка задачи

Приложение должно содержать следующие критерии:

- возможность создания нового кошелька или привязка существующего по seed фразе;
- реализация безопасного хранения и защиты конфиденциальных данных;
- реализация функционала отправки и получения криптовалюты с использованием QR-кода;
- возможность просмотра истории транзакций и ее очищение;
- обеспечение приватности отображения данных (скрытие финансовых показателей);
- поддержка мультиязычности (интернационализация);
- наличие инструментов управления учетной записью и безопасностью (кастомизация внешнего вида приложения и смена PIN);
- приложение должно обеспечивать корректную работу с базовыми криптоактивами (BTC, ETH, USDT);
- отображение актуальной стоимости активов с использованием API запросов с разбивкой по токенам и общей суммой;
- наличие сервера для централизованного управления API запросами.

Стек технологий

android
studio



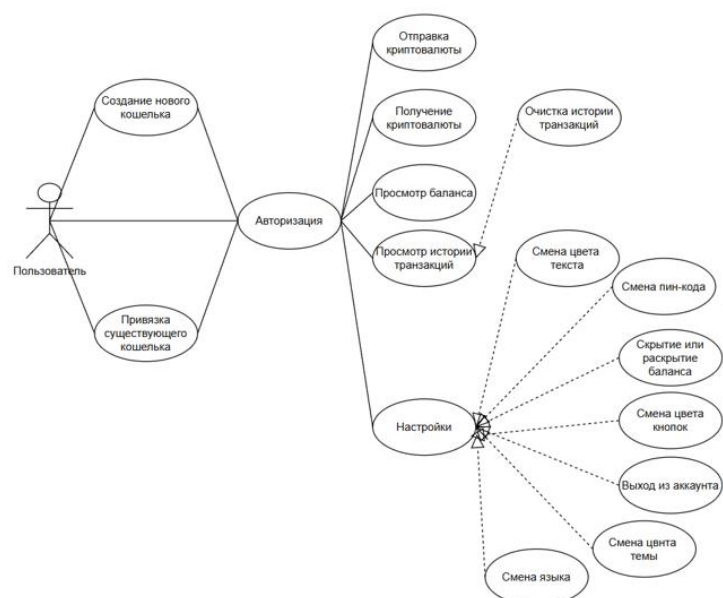
Kotlin

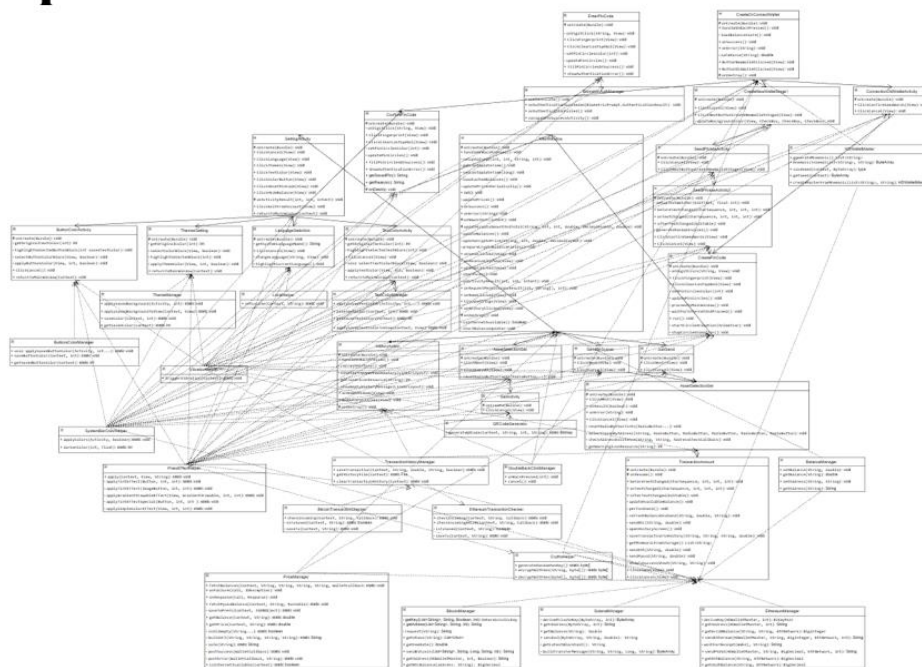
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.29.001 ПЗ

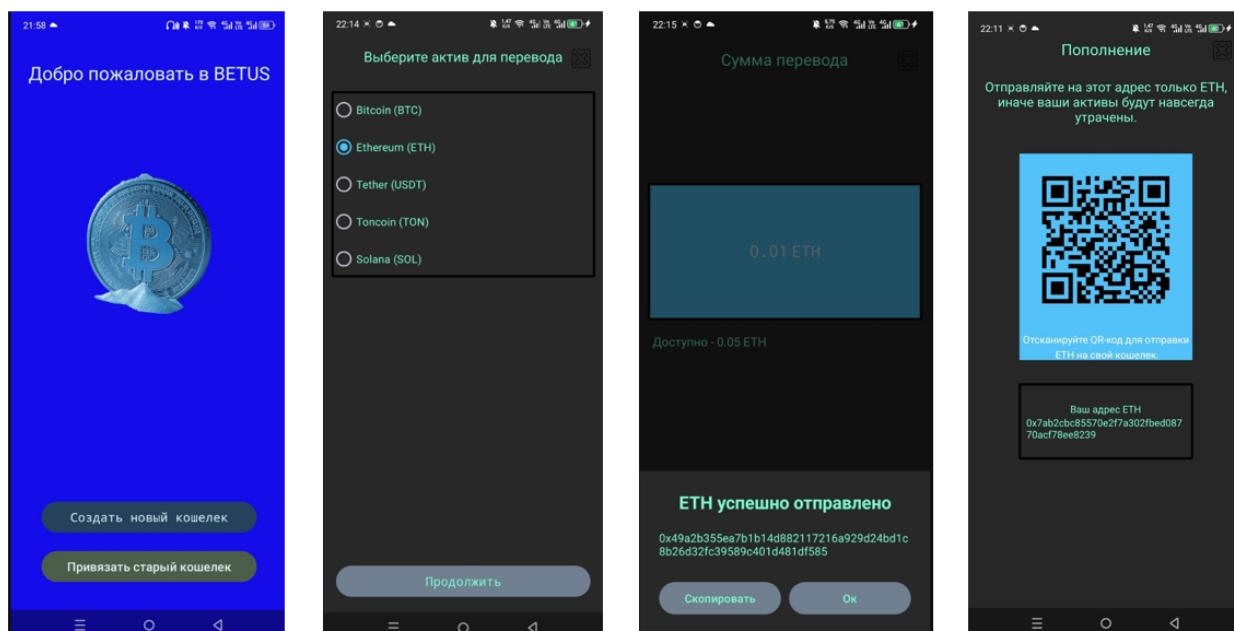
Лист

228





Демонстрация работы приложения



Заключение

В ходе выполнения данной выпускной квалификационной работы, было разработано Android приложение - некастодиальный мультивалютный криптокошелёк «BETUS». В процессе разработки были освоены современные подходы к созданию защищённых мобильных приложений в среде Android Studio с использованием языков Kotlin и Java, а также криптографических библиотек (например, BitcoinJ, Web3j) и системного хранилища ключей Android Keystore для изоляции приватных ключей.

Спасибо за внимание!



					ПГУ 1.090301.29.001 ПЗ	Лист
						231
Изм.	Лист	№ докум.	Подпись	Дата		