Версия 1.4 (processing)

▼ О проекте Allyo

Allyo Wallet – это программный криптокошелёк с повышенным уровнем безопасности, в котором каждый адрес жёстко привязан к верифицированной личности пользователя через госслужбу ЕСИА. Кошелёк не является биржевым или платежным сервисом, его основное предназначение – безопасное хранение и перемещение цифровых активов между **подтверждёнными пользователями**. Главные особенности продукта – **«защищённые счета»** (smart accounts) с набором встроенных политик безопасности (таймлоки, мультиподпись, лимиты, белые списки адресов, эскроу и др.), **управляемое восстановление доступа** в случае утраты ключей, расширенные **антифрод-механизмы**, а также **юридически значимая** привязка личности к адресу и полная протоколизация действий. Такой подход обеспечивает однозначную идентификацию владельца каждой транзакции и формирует доказательную базу для разрешения споров и расследований. Данный документ описывает требования к **минимально жизнеспособному продукту (MVP)** Allyo Wallet, включая необходимые функции, техническую архитектуру и выбор технологий. MVP должен продемонстрировать основные ценности продукта – безопасность, модульность, возможность кроссплатформенной работы – и заложить основу для дальнейшего

развития, соблюдая все актуальные требования законодательства РФ в сфере персональных данных и AML.

> **При разработке функционала ИИ должен учитывать принципы безопасности, модульности и кроссплатформенности. Все решения должны быть масштабируемыми, поддерживать гибкую интеграцию через API и обеспечивать единый, интуитивно понятный пользовательский опыт независимо от устройства и платформы.**

▼ Функциональные требования MVP

Основной функционал MVP разделён на несколько компонентов, описанных ниже. Каждая функция направлена на обеспечение безопасности средств и соответствие регуляторным нормам.

Идентификация через ЕСИА:

Реализовать авторизацию пользователей через систему госуслуг ЕСИА по протоколу OpenID Connect/OAuth 2.0. При входе или регистрации пользователь перенаправляется на ЕСИА и входит под своей учётной записью. После успешной аутентификации сервис получает от ЕСИА подтверждённые атрибуты личности – ФИО, СНИЛС, ИНН, при необходимости паспортные данные, а также уровень подтверждения учётной записи.

> **“Эта интеграция обеспечивает вход под настоящим именем пользователя, что выгодно отличает ЕСИА от OAuth-сервисов соцсетей.”**

На этапе MVP в целях упрощения будет реализован **stub/эмулятор ЕСИА**, возвращающий тестовые данные пользователя, но архитектура должна позволять легко подключить боевую среду ЕСИА позднее. После идентификации система должна выпускать **аттестацию привязки личности к крипто-адресу** – например, регистрировать в системе (или смарт-контракте) запись, однозначно связывающую удостоверенную личность с генерируемым адресом кошелька. Эта аттестация необходима для последующей проверки прав собственности и разрешения споров.

> ***(Примечание: В реальной среде интеграция с ЕСИА требует получения доступа через техпортал ЕСИА; при разработке нужно заложить возможность подключения по OIDC с использованием клиентского сертификата, redirect URL и необходимых scope, таких как openid, fullname и т.п.)***

«Защищённые счета» (смарт-аккаунты): Для каждого пользователя кошелька необходимо создавать специальный **смарт-контракт (smart account)** в блокчейне, который будет владеть его средствами вместо обычного EO-адреса. Эти смарт-аккаунты реализуют встроенные политики безопасности и контроля доступа:

- *Таймлоки и отложенные переводы:* возможность устанавливать задержку (например, 24 часа) на вывод средств, особенно при переводе на новые (неизвестные) адреса. Это позволит обнаружить и предотвратить несанкционированные транзакции до их завершения.
- ~~*Мультиподпись (m-of-n) и подтверждения:* поддержка сценария, когда для крупных или нестандартных транзакций требуется подтверждение несколькими участниками. Например, кошелёк может быть настроен как 2-из-3: основная подпись пользователя + подпись доверенного лица или второго устройства. Мультиподпись повышает безопасность и затрудняет кражу средств одним скомпрометированным ключом.~~
- *Белые списки адресов:* пользователь может указать список доверенных адресов получателей. Переводы на **не** включённые в список адреса могут либо запрещаться, либо требовать задержки/подтверждения. Такая функция позволяет реализовать **allowlisting** для повышения контроля .
- *Адресный эскроу:* поддержка режима escrow – средства переводятся на промежуточный смарт-контракт между двумя пользователями и освобождаются только при выполнении определённых условий (например, обе стороны подтвердили транзакцию, либо с истечением времени средства возвращаются отправителю). В MVP достаточно показать примитивный сценарий:

транзакция может быть помещена в эскроу-режим вручную и затем вручную разблокирована второй стороной.

Смарт-контракты должны быть разработаны для каждой поддерживаемой платформы (для EVM-сетей – Solidity, для Solana – Rust программы). Желательно опереться на проверенные шаблоны (например, OpenZeppelin Contracts, Gnosis Safe) для реализации мультисига, таймлоков и т.д., чтобы снизить риски уязвимостей. Использование смарт-аккаунтов позволит реализовать ряд расширенных функций безопасности, недоступных обычным EOA-адресам , таких как социальное восстановление, 2FA и др. (см. ниже).

Управляемое восстановление доступа:

В случае утраты пользователем доступа к своему кошельку (например, потеря устройства с ключами) должна быть предусмотрена многоуровневая процедура восстановления. Поскольку каждая учётная запись привязана к личности через ЕСИА, предлагается следующий подход (соответствует концепции *social recovery*):

1. **Повторная верификация личности** – пользователь проходит процедуру входа через ЕСИА на новом устройстве, подтверждая свою личность тем же способом, что при регистрации.
2. **Вызов механизма восстановления** – после успешной идентификации система инициирует процесс смены ключа владельца в смарт-аккаунте пользователя. Реализовать это можно, например, через специальную функцию в смарт-контракте кошелька: если поступает подпись от доверенного **«восстановительного ключа»** (заранее прописанного) или от центра восстановления, и прилагаются доказательства верификации личности, контракт перевыпускает права владельца на новый открытый ключ.

> В MVP можно упростить: предусмотреть в контракте второго совладельца (guardian) – сервисный адрес восстановления, который совместно с подтверждённой личностью пользователя может пересчитать ключи.

- ~~**«Холодный период» после восстановления** – после смены ключа кошелёк входит в режим ограничения: например, *N* дней нельзя выводить средства или лимиты урезаны. Это защитит от злоупотреблений, если восстановление было инициировано злоумышленником.~~

Такая схема повторной верификации и смарт-контракта с изменяемым владельцем реализует **social recovery**, когда при утере приватного ключа публичный ключ кошелька может быть изменён на новый через мультисиг с участием заранее доверенных сторон . В нашем случае роль доверенной стороны выполняет сервис, удостоверяющий личность через ЕСИА и/или контакт доверия, назначенный пользователем (например, доверенный аккаунт другого человека или нотариальный ключ). В MVP достаточно реализовать базовый сценарий (например, recovery через сервисный ключ после проверки паспорта/ESIA на бэкенде).

Антифрод-механизмы: Allyo Wallet должен предоставлять пользователю и системе средства раннего обнаружения и предотвращения подозрительной активности:

- ~~Режим **«холодный кошелёк»**: пользователь может пометить свой счёт как холодный – тогда все исходящие транзакции приостанавливаются или требуют длительного таймлока. Это полезно, например, на время длительного отсутствия или если подозревается компрометация.~~
- **Алерты о рисках:** система анализирует действия (например, попытка вывода всех средств сразу, перевод на адрес, помеченный как мошеннический, частые переводы подряд и т.п.) и высылает предупреждения владельцу (через push-уведомление, email). В MVP можно реализовать простые эвристики: сравнение суммы транзакции с порогом, мониторинг известных опасных адресов (черный список).
- **Заморозка по требованию:** в случае подозрения на взлом пользователь через интерфейс может мгновенно приостановить все исходящие операции по своему кошельку. Это реализуется вызовом функции смарт-контракта, которая временно блокирует

переводы (флаг паузы) до разблокировки владельцем или через поддержку.

- ~~**Sandbox-режим для новых действий:** например, если аккаунт ранее не проводил определённые операции (смарт-контракт взаимодействия), можно первые такие операции выполнять с задержкой и вручную подтверждать. В рамках MVP достаточно заложить возможность маркировать транзакции определённого типа флагом «требует ручного подтверждения».~~

  Эти меры соответствуют требованиям 115-ФЗ о контроле и блокировке подозрительных операций. Закон требует, чтобы платформа отслеживала нетипичные транзакции и при необходимости ограничивала доступ к сервису . Allyo Wallet в MVP должен продемонстрировать минимум: ручную заморозку счетов и уведомления о крупных транзакциях.

Протоколирование и юридическая значимость действий: Все значимые действия пользователей и системы должны **протоколироваться** с сохранением в неизменяемом журнале. В системе необходимо вести журналы: входов в систему, создания кошелька, выполнения транзакций (со всеми параметрами), изменения настроек безопасности, попыток восстановления, с указанием времени, ID пользователя и прочих метаданных. Каждый лог-запись должна иметь метку времени и цифровую подпись (например, подпись серверным ключом или хеш, сохраняемый отдельно) для обеспечения целостности. Цель – сформировать **доказательную базу**, которую можно предъявить при внутреннем разбирательстве или в суде, подтверждая кто, когда и что сделал. В MVP достаточно реализовать хранение логов в базе данных и выгрузку отчёта по ним. В перспективе можно добавить хеширование цепочек записей (для неизменности) или запись важных событий в публичный блокчейн для независимой верификации меток времени. Для пользователей должна быть возможность получить **«выписку»** – человекочитаемый отчёт обо всех операциях своего кошелька за период, подписанный ЭЦП сервиса. Это придаст юридическую определённость владения активами и транзакциям. Такой функционал соответствует требованиям

законодательства: напр., 152-ФЗ требует регистрировать все действия с персонданными , а 115-ФЗ – хранить данные по транзакциям и при необходимости предоставлять их регуляторам . В MVP предусмотрим простой экспорт журнала (CSV/PDF) с электронной подписью сервиса.

Поддержка нескольких блокчейн-сетей

MVP должен работать как мультицепочечный кошелёк. Первоначально заявлена поддержка **Ethereum, Binance Smart Chain, Solana** – для смарт-контрактов, а также **Bitcoin** как UTXO-платформы. Реализация:

- Для **Ethereum/BSC (EVM-сети)**: будут использоваться единые смарт-контракты (Solidity) деплоящиеся в каждую сеть. Логика smart account разрабатывается один раз и повторно разворачивается в Ethereum и BSC. Взаимодействие с этими сетями через стандартные библиотеки Web3/Ethers.js на бэкенде и/или клиенте.
- Для **Solana:** необходимо разработать аналогичную программу (smart contract) на Rust, реализующую кошелёк с описанными функциями (мультисиг, таймлок и пр.). В MVP можно ограничиться базовым multisig-программой и time-lock аккаунтом на Solana. Взаимодействие – через Solana Web3 SDK (JavaScript) или REST API ноды.
- Для **Bitcoin:** смарт-контракты в привычном понимании недоступны, поэтому защищённый счёт может быть реализован в виде **мультиподписного адреса** (P2SH) или через скрипты с timelock. Например, при создании биткоин-кошелька генерируется адрес типа 2-из-3 (пользователь, доверенный контакт, сервис-восстановление). Для вывода средств потребуется по крайней мере 2 из 3 подписей или ожидание таймлока, если вторую подпись не предоставили. В MVP реализуем генерацию multi-sig адресов для Bitcoin и базовые операции (просмотр баланса, создание транзакции с несколькими подписями). Взаимодействие с Bitcoin – через подключение к собственному узлу Bitcoin Core (RPC) либо с использованием библиотеки (например, bitcoinjs-lib) на сервере.

Поддержка сразу четырёх сетей делает архитектуру сложной, поэтому особое внимание уделим модульности: логика работы с каждой сетью будет инкапсулирована в отдельном модуле/сервисе с единым интерфейсом. Это позволит при необходимости добавить новые сети или заменить реализацию (например, переключиться с эмулятора на боевой узел) без изменения остального кода.

Интеграция с dApp (WalletConnect)

Для удобства пользователей Allyo Wallet планируется совместимым с dApp-приложениями через протокол **WalletConnect 2.0**. Это значит, что мобильное приложение кошелька сможет подключаться к внешним dApp (децентрализованным приложениям) по QR-коду/ссылке, выступая аналогом MetaMask/TrustWallet. В рамках MVP интеграция с WalletConnect может быть частичной: включаем библиотеку WalletConnect в мобильный клиент и тестируем соединение с одним-двумя популярными dApp (например, Uniswap на тестнете). **Важно:** никакой собственной “цифровой валюты” (токена) платформа не предлагает и не требует – WalletConnect используется только для подписания транзакций пользователем во внешних приложениях. Также, если dApp требует передачу персональных данных, можно предусмотреть OIDC провайдер на базе нашего сервиса (но это за рамками MVP). Интеграция WalletConnect должна быть модульной, чтобы её можно было легко включить/отключить или обновить версию протокола.

Клиентские приложения (Web и Mobile): MVP должен предоставить единый пользовательский опыт на веб-платформе и мобильных устройствах:

- **Веб-клиент:** Веб-приложение (доступ через браузер) для управления кошельком. Функциональность: вход через ЕСИА (в MVP – имитация), просмотр балансов по всем поддержанным сетям, инициирование переводов, отображение журнала операций, настройка политик безопасности (включая управление списком доверенных адресов, лимитов, включение заморозки и т.д.), запуск процесса восстановления. Веб-интерфейс должен быть адаптивным и безопасным (при вводе ключевых фраз или приватных ключей –

использовать шифрование на стороне клиента, предотвращать XSS/CSRF и пр.).

- **Мобильный клиент:** Нативное или кроссплатформенное мобильное приложение с аналогичным функционалом. Приоритетны платформы iOS и Android. Можно использовать **React Native** для ускорения разработки и общего кода с веб (логика на TypeScript), либо Flutter – по выбору, здесь доверяем выбору исполнителя. Главное – обеспечить одинаковый набор возможностей и схожий UI/UX на мобильном и вебе. В частности, мобильное приложение будет ключевым для WalletConnect (см. выше) и для push-уведомлений (алерты безопасности).

  Дизайн интерфейсов должен быть интуитивно понятным, с упором на простоту выполнения безопасных операций. Несмотря на сложность функций (мультиподпись, политики), для пользователя они должны проявляться в виде простых переключателей и подсказок. Единый дизайн-гайд должен применяться на всех платформах.

- **Административный веб-интерфейс (минимально):** Хотя основной продукт – пользовательский кошелёк, в MVP целесообразно реализовать простой интерфейс админки (или хотя бы API) для мониторинга системы: просмотр логов действий, управление списками блокировки (например, адреса в чёрном списке), ручная помощь в восстановлении аккаунта и пр. Доступ в админку – только для ограниченного круга операторов (с сильной аутентификацией, например, через VPN и 2FA). Это связано с требованиями 115-ФЗ об оперативном реагировании на инциденты (заморозка, отчётность) и 152-ФЗ о контроле доступа к данным.

▼ Архитектура и решения

Архитектура MVP выбирается модульной, с разделением на несколько слоёв: **клиентский уровень (Web/Mobile)**, **серверный уровень (Backend/API)** и **блокчейн-уровень**. Ниже описаны основные компоненты и взаимодействие между ними:

- **Клиентские приложения:** Веб-приложение (React/TypeScript) и мобильное приложение (React Native либо Flutter) – отвечают за отображение интерфейса пользователю и выполнение локальных операций (например, хранение ключей, подписание транзакций). Клиенты взаимодействуют с сервером по защищённому API (HTTPS) для операций, требующих серверной логики: авторизация через ЕСИА, получение аттестации личности, запрос списка событий/логов, отправка транзакции (в части координации multisig, если нужно), и т.д. При этом приватные ключи пользователя **не передаются** на сервер: клиент сам подписывает транзакции локально и отправляет только нужные данные (подписанные хеши и пр.). Мобильное приложение дополнительно реализует WalletConnect-связь с dApps и может получать push-уведомления с бэкенда (например, об алертах).
- **Серверная часть (Backend API):** Центральный веб-сервис, который оркестрирует работу системы. Основные обязанности сервера:
    - **Аутентификация и управление пользователями:** редирект на ЕСИА (или эмулятор) для логина, приём OAuth-кода, обмен на id_token/access_token и получение данных пользователя. Создание локальной записи пользователя в базе (с привязкой к его уникальному идентификатору ЕСИА, например, oid или СНИЛС). Выдача пользователю JWT сессии для последующей авторизации в API.
    - **Взаимодействие с блокчейнами:** сервер поддерживает подключение к каждому требуемому блокчейну (либо через запущенные ноды, либо через провайдеры). Он выполняет такие задачи, как: генерация смарт-контракта кошелька для нового пользователя (деплой контракта в сети), обработка входящих транзакций (подписанных пользователем) – например, если транзакция требует нескольких подписей, сервер может временно сохранять первую подпись и ожидать вторую, или инициировать выполнение смарт-контракт функции. Для Bitcoin сервер координирует создание multi-sig адресов и сбор подписей. Также сервер подписывает **аттестацию** личности к

адресу (например, формирует и подписывает JSON Web Token или сертификат, связывающий адрес с данными пользователя).

- **Бизнес-логика политик безопасности:** сервер хранит настройки политики для каждого аккаунта (лимиты, whitelist и др.) и при получении запроса на транзакцию или другое действие проверяет соответствие политикам. Например, если пользователь пытается вывести больше дневного лимита, backend отклонит или поместит транзакцию в отложенные. Некоторые политики реализованы на смарт-контракте (on-chain, напр. timelock), а некоторые можно дублировать на backend для безопасности.
- **Логи и мониторинг:** запись всех операций в базу данных журнала. Реализация механизма immutable-логов: например, сервер помимо записи в БД может писать хеши лог-записей в отдельный файл или в блокчейн (в MVP можно ограничиться БД, но структура должна предусматривать неизменяемость – например, не предоставлять операций удаления/редактирования логов). Также backend анализирует операции на подозрительность (anti-fraud rules) и при срабатывании правил отправляет уведомления или помечает аккаунт.
- **API:** предоставление REST/JSON API или GraphQL для взаимодействия с клиентами. Основные эндпойнты: регистрация/логин (через ЕСИА), получение профиля пользователя (аттестованные данные), создание/просмотр адресов кошелька, запрос баланса (взаимодействуя с блокчейном или кэшем), отправка транзакции (с необходимыми параметрами, подписями), запрос журналов и настроек, управление настройками безопасности, запуск процесса восстановления и пр. API должен быть документирован для возможной интеграции сторонних сервисов (в дальнейшем, например, B2B-клиенты смогут вызывать API для управления кошельками через свой софт).
- **Хранение данных:** связана с БД, см. ниже. Сервер отвечает за безопасное хранение персональных данных (минимально

необходимого набора: ФИО, возможно СНИЛС/ИНН, ID учётки ЕСИА, email/телефон для уведомлений) – всё это должно храниться согласно 152-ФЗ (в зашифрованном виде, на серверах в РФ, с разграничением прав доступа) .

- **База данных:** Реляционная БД (например, PostgreSQL) будет использована для хранения данных пользователей, привязок к адресам, политик безопасности, логов действий, и прочей служебной информации. Выбор PostgreSQL обоснован её надёжностью и поддержкой сложных транзакций – это важно для ведения аудируемых логов. Все чувствительные поля (ПДн) должны быть зашифрованы на уровне приложения или с помощью расширений БД. Также настроены резервное копирование и репликация (во избежание потери данных, что критично для восстановление доступа и логов). БД развёртывается на серверах, физически расположенных в РФ, удовлетворяя требованию локализации персональных данных.
- **Блокчейн-узлы / провайдеры:** Для взаимодействия с каждым поддерживаемым блокчейном предполагается настроить подключение:
  - Ethereum/BSC: либо запуск собственных узлов (Geth/Nethermind для Ethereum, BSC Node) в режиме light/full, либо использование надёжного провайдера (Infura/Alchemy) на этапе MVP. Подключение через web3 провайдеры, доступные из backend.
  - Solana: запуск solana-validator (тестнет) или использование API (QuickNode, Solana RPC).
  - Bitcoin: собственный Bitcoin Core node (testnet/mainnet), с включённым RPC. Для MVP возможно использовать testnet.

  Узлы должны быть синхронизированы и доступны для отправки транзакций и чтения состояния (балансов, контрактов). В перспективе, для отказоустойчивости, можно заложить несколько узлов или fallback на сторонние API.
- **Интеграция с ЕСИА (stub):** Поскольку на этапе MVP нет подключения к боевому контуру ЕСИА, будет реализован *мини-*

сервис-эмулятор (может работать как часть backend или отдельный mock-сервер). При попытке аутентификации он имитирует поведение ЕСИА: переадресует на страничку ввода условных данных, затем возвращает авторизационный код, который backend обменяет на фиктивный id_token содержащий в payload требуемые поля (ФИО, и др.). Структура должна соответствовать OpenID Connect (JWT ID Token). Благодаря такой замене можно отладить весь пользовательский сценарий SSO без реальной интеграции. Тем не менее, архитектура должна предполагать легкое переключение на реальную ЕСИА: URL авторизации, проверку подписи JWT (для ЕСИА нужны ГОСТ-алгоритмы, возможно придётся использовать библиотеку CryptoPro/BouncyCastle), получение доп. атрибутов через API ЕСИА . **Важно:** хранить ключи и секреты, связанные с интеграцией (client_id, сертификаты), в конфигурации сервера (например, в .env с защищённым доступом).

Диаграмма взаимодействия основных компонентов:

- Пользователь в приложении инициирует действие (например, перевод).
- Клиентское приложение при необходимости отправляет запрос на Backend (REST API).
- Backend проверяет права, политики, логирует событие, взаимодействует с БД (например, фиксирует транзакцию pending).
- Backend формирует транзакцию для блокчейна: если EVM – может вызвать функцию смарт-контракта (через web3); если требуется подпись пользователя – отправляет обратно challenge.
- Клиент подписывает транзакцию приватным ключом и возвращает подпись.
- Backend отправляет транзакцию в сеть. После подтверждения – обновляет состояние (баланс, лог).
- При входе: OAuth редирект → ЕСИА (stub) → назад с кодом → обмен на токен → создание сессии.

Архитектура MVP будет реализована сперва как относительно **монолитное приложение** (для скорости разработки), но с хорошим разделением модулей внутри. Например, модуль BlockchainService будет иметь реализации для каждой сети, модуль EsiaAuthService – заглушку, которую потом заменят на реальный OIDC, модуль WalletPolicy – применим политики. Кодовая база структурируется так, чтобы в будущем без больших переделок можно было вынести модули в отдельные сервисы (микросервисы) при росте нагрузки или требований. Таким образом, принципы **масштабируемости и модульности** будут соблюдены с самого начала.

▼ Технологии

Для реализации описанной функциональности предлагается следующий стек технологий, учитывающий требования по кросс-платформенности, безопасности и быстроте разработки:

- **Frontend (Web):** фреймворк **React** (JavaScript/TypeScript) для построения одностраничного приложения. React обеспечивает гибкость и большой выбор библиотек. Для управления состоянием можно использовать Redux Toolkit или Zustand. В качестве UI-библиотеки – что-то лёгкое, например, MUI или AntDesign, с настройкой под нашу дизайн-тему. Билд и разработка через CRA или Vite, либо Next.js если потребуется SSR (но, вероятно, для кошелька SSR не обязателен). TypeScript предпочтителен для надёжности типизации. Все криптографические операции (генерация ключей, подпись транзакций) на фронте будут выполняться через проверенные библиотеки: ethers.js для Ethereum/BSC, @solana/web3.js для Solana, bitcoinjs-lib для Bitcoin и т.п.
- **Mobile:** кроссплатформенный **React Native** (Expo или bare) на TypeScript, чтобы переиспользовать бизнес-логику из веба и обеспечить единый UX. RN имеет библиотеки для WalletConnect (например, WalletConnect SDK) и возможность использовать те же крипто-библиотеки (ethers, solana web3) через JS-бридж. Альтернативно можно применить **Flutter** (Dart) – он тоже позволяет писать под iOS/Android из одного кода, но тогда логику придётся

дублировать с вебом. С учётом акцента на единый стек, выбор React/TS для обоих клиентов выглядит оптимальным. На мобильном приложении нужно особое внимание уделить хранению секретов: применять SecureStorage (Keychain на iOS, Keystore на Android) для хранения приватных ключей/сид-фраз, с дополнительной защитой PIN/биометрией.

- **Backend:** среда выполнения **Node.js** с **TypeScript** – для реализации API сервера. Node хорошо подходит благодаря обилию библиотек для работы с блокчейнами (web3, solana, bitcoin), а TypeScript придаст надёжность коду. В качестве фреймворка – **NestJS** либо легковесный **Express/Koa**. NestJS предлагает из коробки модульность (controllers, providers), что созвучно нашим требованиям модульной архитектуры, а также поддерживает валидаторы DTO, Guards для авторизации, что пригодится для JWT auth. Основные библиотеки:
  - Библиотека Web3 (ethers.js или web3.js) для взаимодействия с Ethereum/BSC.
  - Solana web3 SDK (@solana/web3.js) для Solana.
  - Bitcoin: либо bitcoinjs, либо использование RPC через библиотеку bitcoin-core npm.
  - Библиотека для JWT (например, jsonwebtoken) для выпуска токенов сессии и, возможно, для реализации самого OIDC stub.
  - OAuth2 / OIDC клиент – можно использовать openid-client для подготовки запросов к ЕСИА (в реальном случае) . В MVP же – свой контроллер, имитирующий редирект и callback.
  - Библиотека для ГОСТ-криптографии (при интеграции с реальной ЕСИА) – например, крипто-провайдер через CryptoPro DSS или gost-crypto (на этапе MVP не нужна, но упоминается на будущее).
  - СУБД: драйвер **PostgreSQL** (pg) или ORM (TypeORM/Prisma). Можно начать с TypeORM для быстрого создания сущностей: таблицы User, Wallet, Transaction, Log, etc.

- Логирование: Winston или Pino для структурированных логов. Логи действий могут дублироваться в консоль и в БД (в таблицу Log). Кроме того, настроить audit-trail: например, использовать middleware, который при каждом запросе API записывает кто-что сделал.
- Безопасность: helmet (для базовых security-заголовков), rate-limiter (ограничить частоту важных запросов, чтобы избежать brute-force).

Backend будет развёрнут в среде Linux. Возможно использование **Docker** контейнеров для каждого компонента (Node.js API, PostgreSQL, фронтенд) для удобства деплоя и масштабирования.

- **Smart Contracts:**
  - EVM-сети: язык **Solidity** (версия 0.8.x). Использование фреймворка **Hardhat** для компиляции, тестирования и деплоя контрактов. В контракте кошелька будем применять библиотеки OpenZeppelin (Ownable, MultiSig if available, TimelockController и т.п.) , чтобы не писать с нуля критичные части. Контракт должен включать функции: приём/отправка ETH/BSC-native, управление списком guardians/доверенных лиц, установка таймлоков, выполнение транзакций с задержкой, экстренная пауза (pause) и смена владельца (для recovery).
  - Solana: контракт (программа) на **Rust**. Можно использовать **Anchor framework**, чтобы упростить разработку (Anchor предоставляет декларативный подход к Solana programs). В MVP реализуем ключевые инструкции: инициализация счета с владельцем и опциями (мультисиг), перевод с проверкой подписей M-of-N, возможно отдельная инструкция Pause/Unpause. Тестирование – с Anchor framework (Mocha/Chai in JS or native tests in Rust).
  - Bitcoin: смарт-контрактов нет, но скрипты: будем использовать стандартные опкоды. Multi-sig адрес – через создание P2WSH-скрипта с OP_CHECKMULTISIG. Таймлок – возможно с OP_CHECKLOCKTIMEVERIFY (например, сделать скрипт,

позволяющий вывести средства либо с 2 подписями сразу, либо с 1 подписью пользователя, но не раньше чем через N блоков). Для MVP ручной скрипт можно не писать, а воспользоваться функционалом Bitcoin Core: createmultisig RPC для адреса, и опция nLockTime для задержки при формировании транзакции. Это область для эксперимента, поэтому часть функций (например, timelock) на Bitcoin можно отразить организационно (не проводить ручное одобрение до времени).

Примечание по тестированию контрактов: необходимо покрыть контрактные функции юнит-тестами (Hardhat/Anchor tests) насколько возможно, чтобы убедиться в правильности ограничений (например, что без второй подписи перевод не проходит, что после паузы перевод невозможен и т.п.). Безопасность смарт-контрактов – критична.

- **Безопасность разработки:** Все части проекта должны разрабатываться с учётом безопасного программирования. Например, на фронте – избегать хранения JWT в небезопасных местах (лучше HttpOnly Cookie или SecureStore на мобиле), вводить подтверждения на опасные действия (необратимые операции должны требовать дополнительного подтверждения пользователя). На бэкенде – проводить валидацию всех входящих данных (например, параметры транзакции), проверять права доступа (JWT должен проверяться на каждый запрос, важные действия – с дополнительным подтверждением второго фактора, если включено). В коде смарт-контрактов – учитывать проверки на переполнение, использовать require для всех предположений, ограничивать функции onlyOwner/onlyGuardian. При деплое контрактов обязательно зафиксировать их адреса и версии в репозитории, чтобы было понятно, с каким кодом работает MVP.
- **Инфраструктура и развёртывание:** Для MVP достаточно облачного сервера (или локального) в РФ. Можно использовать, например, VK Cloud, Selectel или Yandex Cloud, чтобы гарантировать физическое размещение в России (соответствие закону о ПДн). Развёртывание Docker-контейнерами или напрямую (Node + PM2, PostgreSQL на

управляемой БД). Обязательно настроить SSL (HTTPS) для всех внешних соединений. Мониторинг: подключить простой мониторинг (метрики нагрузки, логирование ошибок через Sentry либо аналог). Резервирование: ежедневный бэкап БД на защищённое хранилище.

Выбранный стек и архитектура обеспечат выполнение принципов:

- **Кроссплатформенность:** благодаря React/ReactNative большая часть логики повторно используется, UX унифицирован.
- **Масштабируемость:** Node.js backend легко масштабировать горизонтально (stateless, можно несколько инстансов под балансировщиком), БД PostgreSQL – масштабируется вертикально или через шардирование при необходимости. Разделение по сервисам (ESIA, blockchain) позволяет выносить их отдельно.
- **Гибкая интеграция:** наличие документированного API позволит в будущем интегрировать кошелёк с другими сервисами (например, интернет-банки для отображения баланса, или корпоративные системы для массового управления кошельками).
- **Безопасность:** контролируемая среда исполнения смарт-контрактов, верификация пользователей, шифрование данных – всё это заложено с MVP, что облегчит получение необходимых лицензий/одобрений в регуляторных органах.

▼ Требования безопасности и соответствия законодательству

Проект должен удовлетворять требованиям российских нормативных актов, связанным с хранением цифровых активов, персональных данных и противодействием незаконным операциям. В MVP эти требования учитываются следующим образом:

- **Соответствие 152-ФЗ (персональные данные):** все персональные данные пользователей (ФИО, идентификаторы ЕСИА, контакты) хранятся и обрабатываются в дата-центрах, расположенных на территории РФ. Реализуются необходимые организационные и технические меры защиты ПДн: шифрование данных в покое и при передаче, контроль доступа к БД (ограничение круга

администраторов, использование роли с минимумом привилегий), журналирование действий с ПДн , регулярное резервное копирование. Перед запуском системы должна быть проведена классификация ИСПДн и выполнены требования по обеспечению уровня защиты (например, установка сертифицированных средств защиты, межсетевых экранов, СЗИ от НСД, если необходимо, согласно приказам ФСТЭК). Пользователи при регистрации дают согласие на обработку ПДн, уведомляются о политике хранения. Также обеспечим права субъектов данных – например, возможность удаления аккаунта (прекращение обработки ПДн, кроме тех, которые обязаны храниться по закону). Отдельно стоит отметить, что связка «верифицированная личность – криптоадрес» сама по себе может рассматриваться как персональные данные, поэтому она тоже хранится в защищённом виде (может быть хэш от идентификатора, публикуемый в блокчейн для аттестации, без раскрытия ФИО).

- **Соответствие 115-ФЗ (AML/KYC):** платформа изначально предусматривает **полную идентификацию каждого клиента** через ЕСИА, что удовлетворяет требованию KYC (знать своего клиента) . Все транзакции пользователей, особенно на крупные суммы, отслеживаются и при необходимости помечаются для проверки. В MVP будет заложен порог (скажем, эквивалент 600k ₽ в криптовалюте), при превышении которого операция требует ручного рассмотрения либо автоматической приостановки (как упоминалось в антифрод). Такие пороги можно будет настроить согласно актуальным нормативам (напр., сейчас порог 15 000 ₽ для идентификации операций). Система должна уметь формировать отчёт о подозрительных операциях для предоставления в Росфинмониторинг. В частности, предусмотрено хранение расширенной информации о транзакциях (адреса отправителя/получателя, суммы, даты, айди пользователя) сверх требований блокчейна, чтобы облегчить подготовку формы отчёта. В случае поступления официального запроса или выявления факта финансирования терроризма/отмывания, сервис обязан ограничить или заморозить счёт пользователя – функция заморозки у нас есть

(Anti-fraud). Для соответствия требованиям, в политике использования будет прописано, что сервис может приостанавливать обслуживание в таких случаях. Кроме того, **санкционные списки**: нужно интегрировать проверку адресов на принадлежность к санкционным/нелегальным (например, известные тёмные рынки). В MVP это можно реализовать в виде простого списка запрещённых адресов/меток. По мере развития возможно сотрудничество с аналитическими сервисами блокчейн-транзакций (Chainalysis, Crystal и т.д.) для автоматизации AML-проверок.

- **Разграничение доступа и роли:** кроме пользователей, есть администраторы/операторы системы, которые могут выполнять особые действия (помочь в восстановлении, просмотреть логи, разблокировать счёт). Для них должны быть отдельные защищённые учётные записи с усиленной аутентификацией (двухфакторной). Весь их доступ также логируется. Это частично реализуется через упомянутый административный интерфейс.
- **Криптографическая защита:** Все соединения – только по HTTPS с современными протоколами (TLS 1.3). Приватные ключи пользователей никогда не отправляются на сервер; серверные ключи (например, ключ для подписи журналов или для выпуска аттестаций) хранятся в защищённом хранилище (может использоваться HSM на стороне сервера или по крайней мере файл в зашифрованном виде с ограниченным доступом). Для генерации случайных чисел и ключей использовать криптографически стойкие генераторы (Web Crypto API на фронте, Crypto модуль в Node.js на бэке). Пароли (если будут, например, PIN-код кошелька) – хранить только в хэшированном виде (bcrypt/argon2).
- **Соответствие другим нормам:** Учитываем требования 152-ФЗ по уведомлению Роскомнадзора (если система обрабатывает ПДн, оператор обязан уведомиться). Также при запуске боевой версии, если будет обрабатываться значительный объём средств, возможно потребуется получение разрешений (хотя формально кошелёк не является оператором электронных денежных средств).

Однако, позиционирование проекта избегает прямой работы с «цифровой валютой» как платёжным средством для широкой публики, что должно вывести его из-под действия законов о платежных системах. Тем не менее, юридическая служба должна будет проанализировать необходимость получения статуса оператора информации по ЦФА или регистрации в ЦБ, и т.д. – при MVP это вне зоны разработки, но технически мы готовы обеспечить любые необходимые доработки (например, интеграцию с гос. блокчейн-платформой или предоставление данных в ГосСУБД при необходимости).

▼ Ограничения

Для ясного позиционирования следует обозначить, что **MVP не будет включать** ряд возможностей, намеренно остающихся за рамками первичной версии:

- **Отсутствие обменника/торговой функциональности:** Allyo Wallet **не предоставляет** услуг обмена криптовалюты на другую или на фиат, не выступает торговой площадкой P2P. Пользователи не смогут через наш интерфейс создавать ордера, искать встречные предложения и т.п. При необходимости обмен или покупка/продажа будут реализованы только через интеграцию с внешними партнёрами и **для ограниченного круга пользователей** (например, корпоративные клиенты) по отдельным договорам, чтобы не превращать сервис в публичную обменную точку. Это важно с юридической точки зрения – мы позиционируемся именно как кошелёк и сервис безопасности, а не как финансовый посредник.
- **Не принимаем криптовалюту в оплату товаров/услуг:** проект не реализует функционал мерчант-сервиса. То есть нельзя использовать Allyo Wallet прямо для оплаты в интернет-магазине и т.п. (если только магазин сам вручную не проверит транзакцию из нашего кошелька). Таким образом, мы не подпадаем под определение «оператора цифровой валюты для оплаты товаров» в контексте закона о ЦФА.
- **Нет собственного токена или внутренней валюты:** MVP не будет выпускать никакой utility-токен. Все операции производятся в

поддерживаемых существующих криптовалютах (ETH, BNB, BTC, SOL, либо токены ERC-20/SPL при добавлении). Это убирает риски, связанные с регулированием выпуска цифровых активов.

- **Ограниченная поддержка DeFi-интеграций:** кроме общего механизма WalletConnect, никакие конкретные DeFi-протоколы в MVP не интегрированы. Например, нет встроенных функций стейкинга, лендинга, свопов. Основная задача MVP – безопасный custody. Дополнительные сервисы могут быть добавлены потом по запросам пользователей.
- **Упрощённая интеграция ЕСИА:** как оговорено, используется заглушка. Поэтому юридически MVP не должен использоваться с реальными данными граждан, пока не будет подключена настоящая ЕСИА. Вся цепочка, связанная с ГОСТ-сертификатами, OAuth-клиентом в ЕСИА и согласованиями с Минцифры – не реализуется в MVP, а откладывается на этап подготовки промышленной версии. Это позволит быстрее запустить прототип для внутреннего тестирования.
- **Тестовые сети для блокчейнов (опционально):** В зависимости от стратегии, MVP можно развернуть сначала на тестовых сетях (Goerli для Ethereum, BTTC для BSC, Solana Devnet, Bitcoin testnet), чтобы не работать с реальными деньгами на этапе тестирования. В техзадании можно предусмотреть конфигурационный переключатель между использованием тестнета и mainnet. На демо-стенде, скорее всего, будут использоваться тестовые коины.
- **Ограничения по нагрузке и масштабу:** MVP рассчитан на ограниченное число пользователей (например, первые сотни или тысяча). Не требуется сразу обеспечить высокую отказоустойчивость или геораспределённость. Эти аспекты (настройка кластеров, авто-масштабирование) будут решаться по мере роста. Однако архитектурно мы стараемся не закладывать явных узких мест: БД можно вынести на кластер Postgres, Node-бэкенды масштабировать, etc.
- **Интерфейс на одном языке:** MVP будет, вероятно, только на русском языке (с учётом целевой аудитории РФ). Локализация на

англ. и другие языки пока не нужна.

- **UI/UX может быть базовым:** хотя стремимся к удобству, детальный дизайн и отточенный пользовательский опыт могут требовать нескольких итераций. В MVP заложим основы дизайна, но признаём, что он может измениться после сбора обратной связи.
- **Отсутствие внешнего аудита безопасности:** полноценный аудит смарт-контрактов и инфраструктуры – дорогостоящая и длительная процедура. В рамках MVP проведётся **внутреннее тестирование** и peer-review кода, но внешний аудит (Certik, MixBytes и пр.) не проводится. Он запланирован перед выходом в продуктовую эксплуатацию.

▼ Вывод

Представленный MVP призван продемонстрировать ключевые возможности Allyo Wallet – привязку адреса к подтверждённой личности и многоуровневую защиту активов – в действии. Используя современные технологии (React, React Native, Node.js, Solidity/Rust смарт-контракты) и ориентируясь на лучшие практики (мультиподпись, социальное восстановление, KYC/AML compliance), мы создадим основу продукта, который сочетает в себе удобство обычного криптокошелька с надёжностью банковского сейфа. Все принятые технические решения соответствуют принципам безопасности, модульности и масштабируемости, что позволит развивать систему без кардинальных переделок. После реализации MVP планируется этап тщательного тестирования, сбора обратной связи и доработки функционала перед запуском полноценного продукта на рынке. При дальнейшей работе важным будет получение необходимых согласований с регуляторами и проведение независимого аудита, чтобы Allyo Wallet стал не только инновационным, но и доверенным решением для хранения цифровых ценностей.