Содержание

[Введение 3](#_Toc69798258)

[1 Основная часть 4](#_Toc69798259)

[1.1 Постановка цели и формулировка задач 4](#_Toc69798260)

[1.2 Анализ и сравнительный обзор аналогов 5](#_Toc69798261)

[1.2.1 Приложение «Coins» 6](#_Toc69798262)

[1.2.2 Приложение «TrustWallet» 8](#_Toc69798263)

[1.3 Выбор инструментария и технологии проектирования 10](#_Toc69798264)

[1.4 Проектирование программного средства 12](#_Toc69798265)

[1.5 Программная реализация проекта 14](#_Toc69798266)

[Заключение 16](#_Toc69798267)

[Список используемых источников 17](#_Toc69798268)

[ПРИЛОЖЕНИЕ А 18](#_Toc69798269)

# **Введение**

Преддипломная практика проходила на предприятии ООО «ПикселПлекс Лабс» с 22 марта 2021 года по 16 апреля 2021 года.

ООО «ПикселПлекс Лабс» – ведущая компания в сфере блокчейн разработки, которая уже более 10 лет разрабатываем проекты как для всемирно-известных мировых брендов, так и для небольших, перспективных стартапов.

Работая по направлению блокчейн, компания уже поставили на рынок более 80-ти проектов. В портфолио PixelPlex входят разработанные внутри компании многофункциональные платформы, такие как Echo и Qtum, а также другие блокчейн решения для Fintech индустрии, включая обозреватели блоков, майнинговые платформы, крипто кошельки и децентрализованные биржи криптовалют.

PixelPlex также работает и в других передовых направлениях: AI & Machine Learning, Computer Vision, IoT, AR, VR, Mixed Reality. Компания всегда в курсе новейших разработок и технологий индустрии разработки ПО и использует только современный стек технологий в разработки своих проектах.

Коллектив компании состоят из большого числа молодых специалистов, которые только начинают свою карьеру, а также опытных специалистов с десятками годами практического опыта.

Предприятие активно мотивируем своих сотрудников на развитие и самореализацию посредством:

* наставничества;
* предоставления любой технической литературы, в том числе под заказ;
* спонсирования обучающих курсов
* участия в профессиональных конференциях;
* бесплатных корпоративных курсов английского языка

Преддипломная практика для специалистов высшей квалификации – это возможность продемонстрировать свою теоретическую подготовку и практические навыки, полученные в ходе обучения. Получение обратной связи со стороны компании или организации, принимающее студента на практику, приобретение опыта и знаний о текущих реалиях в изучаемой сфере, чтобы соответствовать современным требованиям рынка труда.

# **Основная часть**

* 1. **Постановка цели и формулировка задач**

С развитием электронных систем неоднократно возникали идеи создать электронный аналог наличных денег. Но камнем преткновения становилась потенциальная возможность двойного расходования одних и тех же средств. При оплате наличными двойного расходования никогда не возникает из-за того, что оплата сопровождается передачей денег и покупатель не может ещё раз их заплатить другому продавцу — у него этих денег уже нет. Но электронным системам органично присуща возможность копирования состояния, что позволяет сделать полные копии системы и затем произвести несколько платежей из одного и того же стартового состояния, то есть потратить одни и те же средства в разных направлениях. Проблема решалась лишь с помощью доверенных посредников, которые ведут учёт платежей и гарантируют оплаты исключительно в рамках наличия средств. Именно так работают все системы безналичных платежей — традиционно посредниками выступают банки или иные операторы платёжных систем.

Технология криптовалют изначально нацелена на отсутствие доверенного узла — того, чьи действия гарантированно истинны, и кто может подтвердить корректность чужих операций. Впервые эта проблема была решена в системе «Биткойн» за счёт искусственного усложнения внесения изменений в реестр истории операций. Для хранения информации транзакции объединяются в блоки, из которых формируется непрерывная цепочка. Непрерывность обеспечивается не столько нумерацией, сколько включением в текущий блок хеш-суммы предыдущего блока, что не позволяет изменить информацию в блоке без изменения хешей во всех последующих блоках.

Отсутствие у криптовалют какого-либо внутреннего или внешнего администратора приводит к тому, что банки, налоговые, судебные и иные государственные или частные органы не могут воздействовать на транзакции каких-либо участников платёжной системы. Передача криптовалют необратима – никто не может отменить, заблокировать, оспорить или принудительно совершить транзакцию.

Большинство криптовалют обеспечивают псевдоанонимность – все транзакции между всеми адресами общедоступны, но нет данных о владельцах адресов.

Все вышеперечисленные принципы подтолкнули общество к изучению и использованию различных систем, работающих с криптовалютами: бирж, кошельков, хранилищ приватных данных, анализаторов и других. В связи с повышение спроса и интереса к таким системам со стороны пользователей, появилась необходимость создания удобных и быстрых интерфейсов доступа к необходимым процессам. Самым удобным по мере доступности является мобильное устройство, поэтому предпочтительна разработка продуктов на мобильных операционных системах.

Исходя из этого, целью дипломного проекта является разработка мобильного приложения, представляющее из себя криптокошелек для платформы iOS.

Современный рынок предъявляет высокие требования к безопасности, удобству, быстродействию и внешнему виду мобильных приложений, поэтому дипломный проект должен иметь следующие функциональные возможности:

* приложение должно позволять создавать или импортировать криптокошельки Bitcoin, Ethereum, Litecoin криптовалют;
* функция отправки транзакций поддерживаемых криптовалют;
* возможность просмотра истории транзакций каждого кошелька;
* возможность добавление нескольких кошельков какой-либо поддерживаемой криптовалюты (мультиаккаунтность);
* возможность выбора рабочей сети между mainnet и testnet;
* функция сканирования QR-кодов, содержащих адреса крипто-кошельков, для дальнейшего создания транзакции;
* функция генерации QR-кода адреса криптокошелька пользователя;
* возможность выбора времени автоблокировки приложения;
* функция защиты значимых операций пин-кодом.

Вышеперечисленные требования смогу обеспечить должный пользовательский опыт для конечного потребителя.

* 1. **Анализ и сравнительный обзор аналогов**

Для создания мобильного приложения с уникальным дизайном, удобным и простым для понимания интерфейсом, всеми необходимыми техническими средствами, нужно тщательно изучить уже представленные на рынке аналоги. Это даст возможность выделить успешные дизайнерские решения, организовать структуру страниц, определить необходимые модули, выделить все достоинства и недостатки, что в результате поможет создать лучший вариант представления для разрабатываемого клиента.

Одними из важнейших аспектов при оценке аналогов были функциональность в связке с простой восприятия интерфейса, уровень безопасности и заботы о данных пользователя так как это определяет дальнейшее отношение пользователей к системе

С появлением криптовалют стало необходимо найти место, где эти новые цифровые активы можно хранить. На данный момент у пользователей есть три варианта:

1. Хранить криптовалюту непосредственно на бирже, где она была куплена. Однако, известно, что криптовалютные платформы закрываются, замораживают средства пользователей или подвергаются взлому.
2. Купить аппаратный кошелек.
3. Использовать онлайн-кошелек, чтобы активы всегда были доступны и защищены от хакеров. Для инвесторов самый удобный способ хранения цифровых активов — это онлайн-кошельки, поскольку они позволяют получать доступ к своим активам в любое время и из любого места. Сегодня есть множество сервисов, предлагающих инвесторам возможность хранить свои криптовалюты. Далее будут рассмотрены самые популярные из них.
   * 1. **Приложение «Coins»**

COINS — это мобильное приложение, которое позволяет пользователям отправлять, получать, хранить, инвестировать и изучать криптовалюты.

Одна из самых популярных функций, которая выделяет его среди конкурентов, — возможность покупки криптовалют напрямую через Apple Pay.

COINS очень серьезно относится к безопасности – он защищен биометрическими технологиями и компанией по кибербезопасности Grey Wizard. В кошельке используется кроссплатформенная библиотека trust-wallet-core, которая реализует низкоуровневую функциональность криптографического кошелька для всех поддерживаемых блокчейнов.

Другая особенность COINS в том, что с пользователей берется очень низкая комиссия за проведение транзакций. Независимо от того, совершает ли пользователь транзакции с Bitcoin, Ethereum или Litecoin.

Логически приложение разделено на пять разделов, где каждый выполняет свою отдельную функцию для пользователя. Раздел «Сегодня» отображает данные об общем состоянии криптовалютного рынка, значимые новости и публикации, а также лидеров роста стоимости за последние 24 часа (рисунок 1.2.1.1).



Рисунок 1.2.1.1 – Раздел приложения «Сегодня»

Раздел «Монеты» выводит рыночную информацию о каждой монете. Данные могут быть отсортированы по множеству различных параметров, что дает пользователю самому контролировать и анализировать информацию, выводимую на экран. Скриншот экрана раздела приведен на рисунке 1.2.1.2.



Рисунок 1.2.1.2 – Раздел приложения «Монеты»

Раздел «Мои монеты» позволяет взаимодействовать пользователю с его личными активами – управлять текущими счетами, создавать новые, отправлять и принимать транзакции. Скриншот экрана раздела приведен на рисунке 1.2.1.3.



Рисунок 1.2.1.3 – Раздел приложения «Монеты»

Из плюсов приложения можно выделить приятный дизайн в стиле нативных iOS приложений, высокий уровень безопасности, который выражается в максимальном сокрытии приватных данных от глаз, невозможность производить операции со счетом до сохранения мнемоника в надежном месте, возможность использования биометрических функций устройства для защиты входа и основных операций.

Явным минусом приложения является отсутствие возможности добавить несколько адресов одной валюты, что накладывает ограничения на пользователя, так как ему необходимо искать метод сделать это при необходимости, что, скорее, всего, повлечет за собой переход пользователя на систему конкурентов и, как следствие, финансовые убытки.

* + 1. **Приложение «TrustWallet»**

TrustWallet, насчитывающий более 400 тыс. пользователей в месяц, — официальный криптокошелек Binance, одной из крупнейших в мире бирж криптовалют.

Он предлагает пользователям простое мобильное приложение для управления их токенами и криптовалютами и в то же время позволяет полностью контролировать личные ключи. Ключи хранятся локально и защищены от любой потенциальной атаки множеством уровней безопасности, гарантируя, что никто никогда и несанкционированно не получат к ним доступ.

Особенность кошелька в том, что он отлично работает с протоколами Binance DEX и Kyber Network, что позволяет совершать мгновенные сделки на децентрализованной бирже. Безопасность данных пользователей действительно внушительна для относительно простого криптокошелька, поэтому компания не получит доступ к личной информации.

К особенностям приложения можно отнести:

* закрытые ключи пользователей хранятся на их собственных устройствах;
* для быстрого восстановления доступа предусмотрено средство резервного копирования;
* встроенный браузер Web3 позволяет взаимодействовать с децентрализованными приложениями;
* возможность покупки и продажи монет (осуществляется через подключение к Binance DEX;
* кошелек анонимный – можно пользоваться им, не предоставляя личных данных;
* открытый исходный код и регулярные аудиты безопасности;
* возможность импортировать в Trust Wallet большинство других криптокошельков.

У Trust Wallet есть встроенный веб-браузер, который позволяет легко и безопасно пользоваться веб-приложениями. Кошелек совместим со многими криптовалютами и протоколами, в том числе ETH, BTC, XRP, TRX, XLM и BNB. Кроме того, он предлагает полную поддержку любых токенов ERC-20 в сети Ethereum и BEP2 в блокчейне Binance. Официальные версии кошелька доступны для Android и iOS в Google Play и App Store.

Логически приложение разделено на 2 основных раздела и 1 вспомогательный. Раздел «Wallet» отвечает за взаимодействие пользователя с его активами – просмотр баланса адресов каждой монеты, отображение истории транзакций, функции отправки транзакции и запроса средств путем генерирования QR-кода с закодированными в него данными об адресе пользователя и сумме транзакции. Скриншот экрана раздела приведен на рисунке 1.2.2.1.



Рисунок 1.2.2.1 – Раздел приложения «Wallet»

Следующим основным разделом разделом приложения является раздел «DEX», в котором находится функциональность по обмену одних монет на другие, что исключает затраты на перевод монет в базовые электронные валюты и обратно в криптовалюты, а также возможность покупки криптовалют через стандартные банковские системы через систему оплаты.

Trust Wallet подключается непосредственно к Binance DEX, чтобы дать пользователям возможность торговать. Внешне все выглядит как традиционная криптобиржа с книгой ордеров, где можно создавать свои ордеры на покупку и продажу. Следует помнить, что торговля на бирже требует сетевой комиссии, поскольку деньги переводятся напрямую с кошелька на кошелек.

Продажа осуществляется точно так же, необходимо переключиться на раздел «Продать» в поле создания ордера.

Основной проблемой в подобных разделах является сложность интерфейса для простых пользователей. В TrustWallet, в свою очередь, раздел выполнен лаконично – без лишней нагруженности, которая может отпугивать. Скриншоты экрана раздела «DEX» приведен на рисунке 1.2.2.2.

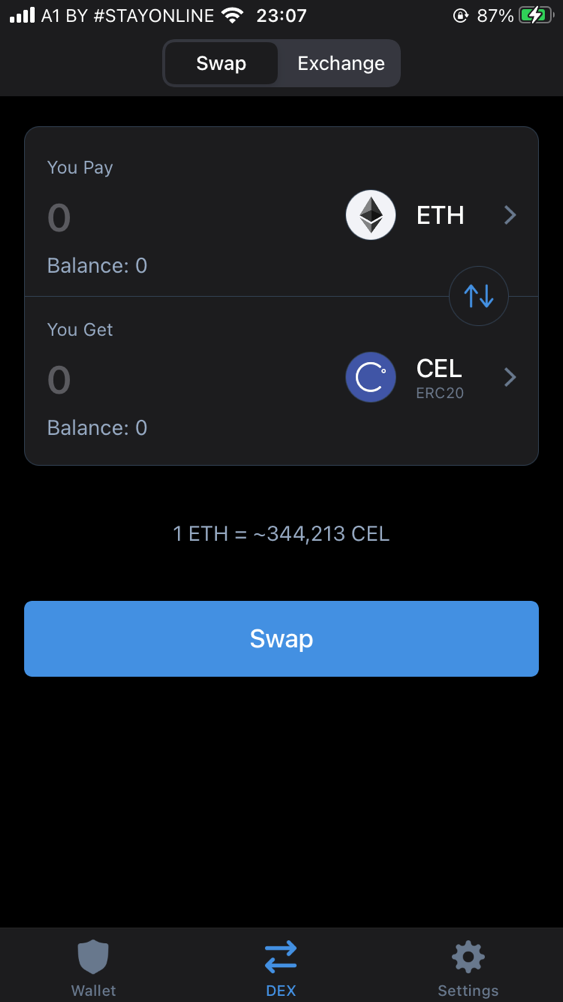


Рисунок 1.2.2.2 – Раздел приложения «DEX»

Криптоактивы такого рода обычно представляют реальные объекты – от произведений искусства до домашних животных. Есть ограничения на генерируемое количество токенов. Trust Wallet предоставляет простой способ безопасно хранить такие активы. Токены связаны с публичным адресом и защищены средствами блокчейна.

* 1. **Выбор инструментария и технологии проектирования**

iOS — популярная мобильная операционная система для смартфонов, электронных планшетов, носимых проигрывателей и некоторых других устройств, разрабатываемая и выпускаемая американской Apple, которая является лидеров в области разработки программного обеспечения и производства носимых устройств.

В iOS используется ядро XNU, основанное на микроядре Mach и содержащее программный код, разработанный компанией Apple. Ядро iOS почти идентично ядру настольной операционной системы Apple macOS. Начиная с самой первой версии, iOS работает только на планшетных компьютерах и смартфонах с процессорами архитектуры ARM. [1]

Swift — компилируемый язык программирования общего назначения. Создан компанией Apple в первую очередь для разработчиков iOS и macOS. Swift работает с фреймворками Cocoa и Cocoa Touch и совместим с основной кодовой базой Apple, написанной на Objective-C.[2] Swift задумывался как более лёгкий для чтения и устойчивый к ошибкам программиста язык, нежели предшествовавший ему Objective-C. Программы на Swift компилируются при помощи LLVM, входящей в интегрированную среду разработки Xcode. Swift может использовать runtime Objective-C, что делает возможным использование обоих языков (а также С) в рамках одной программы.

Xcode – это бесплатная IDE (интегрированная среда разработки) предназначенная преимущественно для разработки приложений для экосистемы Apple – iPad, iPhone и Mac. XCode обеспечивает инструменты для управления всем потоком операций разработки — от создания приложения, к тестированию, оптимизации и представлению его к App Store. [3]

На момент написания статьи Xcode предстал перед миром в 12 версиях. Данная IDE содержит в себе весь необходимый инструментарий для разработки приложений самого различного масштаба

Для удобства создания новых проектов присутствует большой выбор готовых шаблонов для разработки под все устройства Apple. Помимо шаблонов для разработки приложений есть выбор для создания различных библиотек.

Особенности IDE «Xcode»:

1. Интерфейс Xcode интегрирует редактирование кода, средство разработки пользовательского интерфейса, управление ассетами, тестирование и отладку в единственном окне рабочей области;
2. Пакет Xcode включает в себя изменённую версию свободного набора компиляторов GNU Compiler Collection и поддерживает языки C, C++, Objective-C, Objective-C++, Swift, Java, AppleScript, Python и Ruby с различными моделями программирования, включая Cocoa, Carbon и Java;
3. В состав входят большая часть документации разработчика от Apple и Interface Builder приложение, использующееся для создания графических интерфейсов.[4]

Выбор архитектуры приложения исходит из соображений разделения ответственности и слабосвязности между различными программными компонентами проекта, легкости восприятия логики написанной кодовой базы, а также удобства тестирования.

Под вышеописанные требования подходит архитектура «VIPER», которая позволяет разделить UI логику от бизнес-логики, выстроить прозрачную навигацию между экранами приложения, реализовывать в ней различные паттерны программирования. Архитектура «VIPER» будет рассмотрена подробнее в следующих разделах отчета.

Из дополнительных технологий можно выделить утилиту «Cocoapods», обеспечивающую подключение зависимостей к проекту. Все необходимые зависимости размещаются в файле Podfile, после чего, через выполнение команды через терминал операционной системы, в автоматическом режиме создается общее окружение проекта со всеми необходимыми интегрированными зависимостями.

Утилита Generamba – утилита автогенерации отдельные VIPER модули из шаблона. Generamba позволяет ограничить разработчика от однообразного создания файлов каждого отдельного файла модуля и написания его скелета.

SwiftLint — это утилита от разработчиков Realm для автоматической проверки Swift-кода, что позволяет придерживаться единого стиля в написании кода, делать его более читабельным и понятным для восприятия. Утилита содержит набор правил, основанных на GitHub's Swift Style Guide. Все правила находятся в файле swiftlint.yml, куда можно добавлять пользовательские правила анализатора. SwiftLint поддерживает интеграцию с Xcode, Appcode и Atom.

* 1. **Проектирование программного средства**

Проектирования программного средства является важным этапом жизненного цикла разработки ПО, так как на данном этапе закладывается архитектура и структура проекта, которая, в свою очередь, при правильном подходе, обеспечивает масштабируемость, тестируемость, и скорость разработки программного средства.

Архитектура разрабатываемого приложения представляет собой несколько проектов-модулей, которые образуют иерархию зависимостей. Общая структура приложения представлена на рисунке 1.4.1.

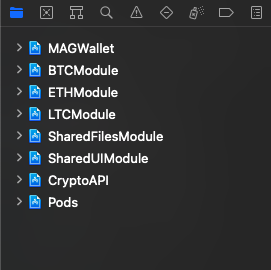


Рисунок 1.4.1 – Общая структура приложения

Модуль SharedUIModule является проектом, содержащим в себе классы, структуры, а также ассеты, связанные с пользовательским интерфейсом, которые являются общими для всех остальных модулей. Такими компонентами являются классы кастомизированных элементов интерфейса, цвета, изображения и другое.

Модуль SharedFilesModule – проект, который включает в себя классы, структуры, расширения и протоколы, которые используется в остальных модулях-проектах и могут быть выделены в отдельный модуль. Данное архитектурное решение позволит избежать дублирования объемных частей кода и создать единое место для общих элементов. Такими компонентами являются классы кор-компонентов, сервисов, хелперов, event-proxy и некоторых других.

BTCModule, ETHModule, LTCModule – модули Bitcoin, Ethereum и Litecoin криптовалют соответственно. В данных модулях находится UI компоненты и бизнес-логика соответствующей валюты.

Модуль CryptoAPI, который необходим для выполнения запросов, связанных с работой в blockchain сети, носит вспомогательную функцию и может использоваться в BTCModule, ETHModule, LTCModule и MAGWallet модулях.

Основой приложения, которая объединяет в себе все вышеперечисленные модули, является модуль MAGWallet. Модуль отвечает за жизненный цикл приложения, построение экранов и навигацию между ними, следит за основными возможностями устройства, такой как, например, доступ в Интернет.

Архитектура экранов в приложении основана на архитектуре «VIPER», которая упоминалась ранее. В ней каждый модуль состоит минимум из 7 необходимых частей: ViewController, Presenter, Interactor, Router, Configurator, Assembler и Protocols. Пример структуры модуля изображен на рисунке 1.4.2.



Рисунок 1.4.2 – Структура VIPER-модуля

Детальнее рассмотрим каждую составляющую модуля. ViewController – класс, реализующий протокол UIViewController библиотеки UIKit, и отвечающий за то, как отображать пришедшие из вне данные, обработку и отдачу во внешнюю среду UI ивентов, таких как, например, нажатия кнопок, расположенных на данном экране.

Interactor отвечает за логику, с которая выполняется на экран, держит зависимости на сервисы, core-компоненты и различные прокси.

Router необходим VIPER-модуля для сборку других модулей и обеспечения навигации в случаях, когда один экран порождает другой.

Presenter является связующим звеном между ViewController, Interactor и Router. ViewController сообщает Presenter об действиях пользователя, тот, в свою очередь, основываясь на выполненном действии определяет, какую бизнес-логику должен выполнить Interactor. Interactor отдает результат работы снова в Presenter, а Presenter принимает решение, какие изменения должны произойти на экране пользователя. В случаях, когда действия пользователя ведут к переходу на другой экран, Presenter обращается к Router, чтобы получить новый VIPER-модуль для отображения, и, в последствии, отдает его на отображение во ViewController.

Protocols содержит в себе протоколы, определяющие интерфейс взаимодействия между различными частями VIPER-модуля.

* 1. **Программная реализация проекта**

При разработке проекта использовались некоторые нестандартные подходы для решения поставленных задач. В данном разделе отчета будут выделены и описаны некоторые из них.

Одно из самых важных требований к приложению криптокошельку – его безопасность. Никакая из важных операций, доступных в приложении, не должна быть выполнена несанкционированно.

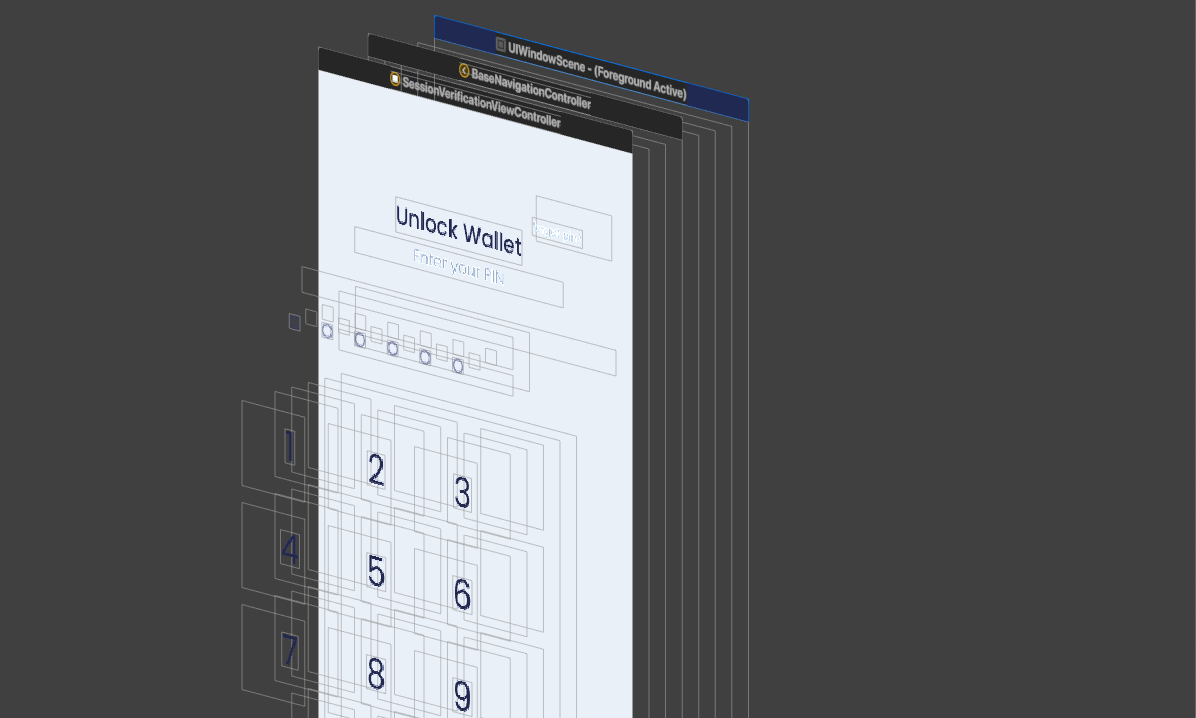


Рисунок 1.5.1 – Структура view ввода пин-кода

В приложении предусмотрена защита посредством ввода пин-кода перед каждой такой операцией., установленного пользователем. Для того, чтобы обеспечить неизменность ситуации, когда пользователю необходимо ввести пин-код перед совершением каких-либо действий, был разработан механизм, суть которого заключается в постоянном наличии прозрачного неактивного UI cлоя поверх всего остального интерфейса приложения. В момент времени, когда необходимо подтвердить какую-либо из операций, во view этого слоя генерируется форма ввода пин-кода. В таком случает, экран пин-кода может быть показан в любой момент приложения и неважно, какие действия на данный момент совершает пользователь – слой пин-кода располагается выше всего в дереве иерархии view приложения и перекроет любой другой интерфейс пользователя. Рассматриваемая концепция изображена на рисунке 1.5.1.

Далее, борясь за безопасность пользовательских данных, будет рассмотрен сервис бизнес-слоя, отвечающий за сохранение конфиденциальных данных на устройстве в зашифрованном виде. Такое шифрование может понадобится для того, чтобы безопасно хранить мнемоник пользователя, например. Часть функциональных возможностей класса изображено на рисунке 1.5.2. Полный код класса расположен в приложении А.



Рисунок 1.5.2 – Часть класса SensitiveDataService

Здесь, все данные, которые необходимо хранить в зашифрованном виде, хранятся в так называемой map, представляющей из себя словарь значений типа string. Само шифрование значений происходит при помощи core-компонента CryptoCore, задача которого заключается в функциях шифрования и дешифрования данных.

# **Заключение**

За время прохождения преддипломной практики было выполнено ознакомление со структурой и с комплексом технических и программных средств предприятия ООО «ПикселПлекс Лабс», выполнен аналитический обзор аналогов по тематике дипломного проектирования, выполнено проектирование, разработан дизайн, выполнена программная реализация приложения.

Выполненная работа основана на официальной документации используемых технологий, выбранные методы реализации практичны и современны, могут использоваться в текущих реалиях рынка разработки мобильных приложений. Основные цели и задачи были выполнены.

В ходе прохождения практики регулярно поступала информация от руководителя, контролирующего этапы разработки.

# **Список используемых источников**

1. Википедия: Операционная система iOS [Электронный ресурс] — Режим https://ru.wikipedia.org/wiki/IOS — Дата доступа: 15. 04. 2021.
2. Википедия: Swift – язык пограммирования [Электронный ресурс] — Режим https://ru.wikipedia.org/wiki/Swift — Дата доступа: 15. 04. 2021.
3. Xcode. Что это такое? [Электронный ресурс]. —Режим доступа: https://bizzapps.ru/p/xcode/ — Дата доступа: 16. 04. 2021.
4. Xcode. Профессиональна IDE для разработки [Электронный ресурс] —Режим доступа: <http://nikulux.ru/ios-macos/xcode/> — Дата доступа: 16. 04. 2021.

**ПРИЛОЖЕНИЕ А**

final class SensitiveDataServiceImp: SensitiveDataService {

var protectedStorage: StorageCore!

var cryptoCore: CryptoCoreComponent!

func redecryptSensitiveData(newPass: String, oldPass: String) throws {

let map = try obtainStorageMap()

var newMap = [String: String]()

for (name, encryptedPass) in map {

let decryptedPass = try cryptoCore.decrypt(hash: encryptedPass, salt: oldPass)

let newEncryptedPass = try cryptoCore.encrypt(string: decryptedPass, salt: newPass)

newMap[name] = newEncryptedPass

}

try setStorageMap(map: newMap)

}

func obtainSensitiveData(pass: String, key: String) throws -> String {

let map = try obtainStorageMap()

if let encryptedValue = map[key] {

let decryptedValue = try cryptoCore.decrypt(hash: encryptedValue, salt: pass)

return decryptedValue

} else {

throw StorageCoreError.resultNotFound

}

}

func setSensitiveData(pass: String, key: String, data: String) throws {

var map = try obtainStorageMap()

let decryptedValue = try cryptoCore.encrypt(string: data, salt: pass)

map[key] = decryptedValue

try setStorageMap(map: map)

}

func removeSensitiveData(key: String) throws {

if var map = protectedStorage.get(key: Constants.Keys.walletsDataMap, type: [String: String].self) {

map[key] = nil

try setStorageMap(map: map)

} else {

throw StorageCoreError.resultNotFound

}

}

private func obtainStorageMap() throws -> [String: String] {

if let map = protectedStorage.get(key: Constants.Keys.walletsDataMap, type: [String: String].self) {

return map

} else {

let map = [String: String]()

try setStorageMap(map: map)

return map

}

}

private func setStorageMap(map: [String: String]) throws {

try protectedStorage.set(key: Constants.Keys.walletsDataMap, value: map)

}

func clear() {

try? setStorageMap(map: [:])

}

}