Реферат

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Марковский А.Г. |  |  | Реферат | Лит. | | | Лист | Листов |
| Пров. | | Годун А.В. |  |  |  |  |  | 1 | 1 |
| Консульт. | | Годун А.В. |  |  | 74417049, 2021 | | | | |
| Н. контр. | | Рыжанкова А.С. |  |  |
| Утв. | | Пацей Н.В. |  |  |

Пояснительная записка содержит 66 страниц, 50 рисунков, 8 таблиц, 27 источников, 6 приложений, 2 листинга.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КРИПТОВАЛЮТНЫЙ КОШЕЛЕК, ПЛАТФОРМА РАЗРАБОТКИ IOS, SWIFT, ПАТТЕРН ПРОЕКТИРОВАНИЯ VIPER, BLOCKCHAIN, XCODE

Объектом дипломного проекта является разработка мобильного приложения, выполняющего функции криптовалютного кошелька.

Цель дипломного проекта – спроектировать и разработать приложение для мобильной платформы iOS, представляющее из себя криптокошелек.

В разработке дипломного проекта был использован язык программирования Swift, протокол обмена данными HTTP.

Экономическая эффективность проекта заключаются в программной структуре проекта, благодаря которой, на ее основе, могут быть разработаны подобные приложения в более сжатые сроки.

Пояснительная записка состоит из введения, шести разделов и заключения.

Во введении представлена общая информация о текущем состоянии в сфере рассматриваемой темы.

В первом разделе представлены результаты аналитического обзора предметной области и технологий разработки.

Во втором разделе представлен процесс проектирования архитектуры мобильного приложения.

В третьем разделе продемонстрирован процесс разработки программного средства.

В четвертом разделе проведено тестирование программного средства и представлены его результаты.

В пятом разделе представлено руководство пользователя мобильного приложения.

В шестом разделе представлены результаты себестоимости и отпускной цены разработанного программного средства.

В заключении представлены итоги дипломного проекта и задачи, которые были решены в ходе разработки программного средства.

Abstract

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Марковский А.Г. |  |  | Реферат | Лит. | | | Лист | Листов |
| Пров. | | Годун А.В. |  |  |  |  |  | 1 | 1 |
| Консульт. | | Годун А.В. |  |  | 74417049, 2021 | | | | |
| Н. контр. | | Рыжанкова А.С. |  |  |
| Утв. | | Пацей Н.В. |  |  |

The explanatory note of the diploma project contains 100 pages of explanatory note, 11 illustrations, 12 tables, 58 sources used, 5 appendices.

MOBILE APPLICATION, CRYPTOCURRENCY WALLET, IOS DEVELOPMENT PLATFORM, SWIFT, VIPER PATTERN DESIGN, BLOCKCHAIN, XCODE

The object of the diploma project is to develop a mobile application that performs the functions of the cryptocurrency wallet.

The purpose of the graduation project is to design and develop an application for the IOS mobile platform, which represents the cryptococheries.

In the development of the graduation project, the Swift programming language was used, the HTTP data exchange protocol.

The economic efficiency of the project is to the project program structure, due to which, on its basis, such applications can be developed in more compressed time.

The explanatory note consists of an introduction, six sections, conclusion.

The introduction provides general information about the current state of affairs in the field of the topic under consideration.

The first section presents the results of an analytical review of the subject area and development technologies.

The second section introduces the process of designing a mobile application architecture.

In the third segment of the software development process.

The fourth section contains the results of software testing.

The fifth section provides a user guide for the mobile application.

The sixth section presents the results of the cost and selling price of the developed software tool.

In conclusion, the results of the diploma project and the tasks that were solved in the course of software development.

Содержание

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Марковский А.Г. |  |  | Содержание | Лит. | | | Лист | Листов |
| Пров. | | Годун А.В. |  |  |  |  |  | 1 | 1 |
| Консульт. | | Годун А.В. |  |  | 74417049, 2021 | | | | |
| Н. контр. | | Рыжанкова А.С. |  |  |
| Утв. | | Пацей Н.В. |  |  |

**Введение**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Марковский А.Г. |  |  | Введение | Лит. | | | Лист | Листов |
| Пров. | | Годун А.В. |  |  |  |  |  | 1 | 1 |
| Консульт. | | Годун А.В. |  |  | 74417049, 2021 | | | | |
| Н. контр. | | Рыжанкова А.С. |  |  |
| Утв. | | Пацей Н.В. |  |  |

С развитием электронных систем неоднократно возникали идеи создать электронный аналог наличных денег. Но проблемой становилась потенциальная возможность двойного расходования одних и тех же средств. При оплате наличными двойного расходования не возникает из-за того, что оплата сопровождается передачей денег, но электронным системам присуща возможность копирования состояния, что позволяет произвести несколько платежей в разных направлениях. Проблема решалась лишь с помощью доверенных посредников, которые ведут учёт платежей и гарантируют операции исключительно в рамках наличия средств.

Технология криптовалют изначально нацелена на отсутствие доверенного узла за счёт искусственного усложнения внесения изменений в реестр истории операций. Отсутствие какого-либо администратора приводит к тому, что никто не может воздействовать на транзакции участников сети.

Вышеописанная особенность подтолкнула общество к изучению и использованию различных систем, работающих с криптовалютами: бирж, кошельков, хранилищ приватных данных и других. В связи с повышение спроса на такие системы, появилась необходимость создания удобных и быстрых интерфейсов доступа к повседневным процессам.

Исходя из этого, целью дипломного проекта является разработка мобильного приложения, представляющее из себя криптокошелек для платформы iOS со следующими функциональными возможностями:

* 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.

**1 Аналитический обзор литературы и постановка задачи**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 01.00.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Марковский А.Г. |  |  | 1 Аналитический обзор литературы и постановка задачи | Лит. | | | Лист | Листов |
| Пров. | | Годун А.В. |  |  |  |  |  | 1 | 1 |
| Консульт. | | Годун А.В. |  |  | 74417049, 2021 | | | | |
| Н. контр. | | Рыжанкова А.С. |  |  |
| Утв. | | Пацей Н.В. |  |  |

Для создания мобильного приложения с уникальным дизайном, удобным и простым для понимания интерфейсом, всем необходимым техническим функционалом, нужно тщательно изучить уже представленные на рынке аналоги. Это даст возможность выделить успешные дизайнерские решения, организовать грамотную структуру страниц, определить необходимые модули, отметить все достоинства и недостатки, что в результате поможет разработать лучший вариант продукта для конечного 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 наличие надежного места, где эти новые цифровые активы можно хранить. На данный момент у пользователей есть три варианта:

* хранить криптовалюту непосредственно на бирже, где она была куплена. Однако, известно, что криптовалютные платформы закрываются, замораживают средства пользователей или подвергаются взлому.
* купить аппаратный кошелек.
* 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.
  1. Исследование предметной области
     1. **Требования к хранению приватных данных пользователя**

Безопасность приложений – один из самых важных аспектов разработки программного обеспечения. Пользователи приложений надеются, что информация, которую они предоставляют, надежно защищена, поэтому нельзя так просто предоставлять кому-либо конфиденциальную информацию.

Исследования, проведенные над множеством iOS приложений, показывают, что во многих допускается одна и та же ошибка: конфиденциальная информация хранится там, где ее быть не должно.

В AppStore были найдены приложения, которые хранят данные в UserDefaults, такие как: токены, активные и возобновляемые подписки, число доступных денег и другое. Все эти данные можно получить и использовать со злым умыслом, начиная от управления платными подписками в приложении и заканчивая взломом на сетевом уровне.

Если разработчик решает хранить данные личного характера в UserDefaults, то он подвергаете ее риску. UserDefaults является хранилищем, где данные хранятся в файле со списком свойств, который находится внутри папки «Настройки» в вашем приложении. Данные сохраняются в приложении без шифрования. В UserDefaults следует хранить только небольшой объем информации, такой как настройки внутри приложения, то есть данные, которые не являются конфиденциальными для пользователя.

Реализация хэширования вручную может оказаться чересчур сложным, поэтому в разрабатываемом программном средстве можно воспользоваться библиотекой CryptoSwift. В ней собрано множество стандартных надежных алгоритмов шифрования, применяемых в Swift.

Никогда не стоить пренебрегать безопасностью разрабатываемого приложения. Понимание разницы и последствиий при хранении конфиденциальных данных в UserDefaults или Keychain является очень важным аспектом разработки под мобильную платформу iOS.

* + 1. **Способы защиты действий пользователя, требующих подтверждения**

В программных средствах, работающих с приватными данными, такими как, например, валютные активы пользователя, все значимые действия должны быть подтверждены непосредственно перед их программным выполнением для того, чтобы удостовериться в том, что именно владелец этих данных производит операции над ними и для того, чтобы дать пользователю понимание важности предстоящего действия, которое, возможно, в дальнейшем, нельзя будет обратить.

В данный момент есть много разнообразных способов решения задачи об подтверждении действий:

– пин-код или пин-пароль;

– одноразовые пароли на бумаге;

– отправка временного кода или ссылки на адрес электронной почты;

– отправка одноразового пароля по SMS;

– OTP-токены;

– биометрическое сканирование.

Одноразовые пароли на бумаге, как правило, используют онлайн-банки. Можно, например, получить в офисе кредитной организации карту со стираемым покрытием поверх полей с кодами или просто распечатать набор паролей в банкомате, а система дистанционного обслуживания при совершении транзакции на сайте будет просить ввести одноразовый набор символов под определённым номером.

Отправка пароля по электронной почте — это максимально упрощённый вариант защиты. Для аутентификации пользователя нужно знать всего лишь его адрес. Соответственно, уровень защиты таким методом — ниже, чем в других сценариях, особенно если используется один пароль для входа в почту и на целевой ресурс.

Ещё одним способом двухфакторной аутентификации является отправка одноразового пароля по SMS. Однако коды в SMS-сообщениях — тоже не очень надёжный вариант. Во-первых, такой пароль можно подсмотреть при помощи уведомлений на экране блокировки смартфона. Во-вторых, сообщение может быть перехвачено вредоносной программным обеспечением. Распространён также вид мошенничества, при котором путём обмана или сговора в салоне сотовой связи можно получить новую SIM-карту с нужным номером, и SMS-сообщения будут приходить на неё, а телефон жертвы даже не сможет подключиться к сети.

Более надёжным является аппаратный OTP-токен. Он представляет собой устройство в виде брелка с дисплеем и кнопкой. В памяти устройства заранее программируется определённое количество паролей. При нажатии на кнопку коды отображаются на дисплее.

Пин-код или пин-пароль устанавливается самим пользователем в основном при регистрации и используется в дальнейшем для подтверждения операций. Недостатки данного способа очевидны – код или пароль не является одноразовым, может совпадать с паролями других сервисов, а также может быть крайне простым, что не вызовет сложностей у злоумышленников при попытке взлома. Однако у пин-кода есть и преимущество – простота и скорость ввода, что может быть допустимо, в зависимости от разрабатываемого программного средства.

Биометрическое сканирование предполагает непосредственный физический контакт владельца с устройством, с которого производится операция, что, в большинстве случаев, является довольно надежным способом защиты.

* 1. Обзор аналогов разрабатываемого программного средства
     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 взымается 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 1.1.1.1).



Рисунок 1.1.1.1 – Раздел приложения «Сегодня»

Раздел «Монеты» выводит рыночную информацию о каждой монете. Данные могут быть отсортированы по множеству различных параметров, что дает пользователю возможность самому контролировать и анализировать информацию, выводимую на экран. Снимок экрана раздела приведен на рисунке 1.1.1.2.



Рисунок 1.1.1.2 – Раздел приложения «Монеты»

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



Рисунок 1.2.1.3 – Раздел приложения «Монеты»

Из достоинств 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 биометрического функционала 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 другой способ сделать это, если возникнет такая необходимость, что, скорее всего, повлечет за собой переход пользователя на платформу конкурентов и, как следствие, финансовые убытки.

* + 1. **Приложение «TrustWallet»**

TrustWallet, насчитывающий более 400 тыс. пользователей в месяц. TrustWallet — 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 и никогда несанкционированно не получит 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, что позволяет пользоваться им, не предоставляя личных данных;
* открытый исходный код и регулярные аудиты безопасности от сертифицированных специалистов;
* возможность импортировать в TrustWallet большинство других криптокошельков.

У TrustWallet 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 два основных раздела и один 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 предполагаемой транзакции. Снимок экрана раздела приведен на рисунке 1.2.2.1.



Рисунок 1.2.2.1 – Раздел приложения «Wallet»

Следующим основным 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.

TrustWallet подключается непосредственно к Binance DEX, чтобы дать пользователям возможность торговать активами. Внешне все выглядит как традиционная криптобиржа с книгой заказов, где можно создавать свои заказы 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 заказа.

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

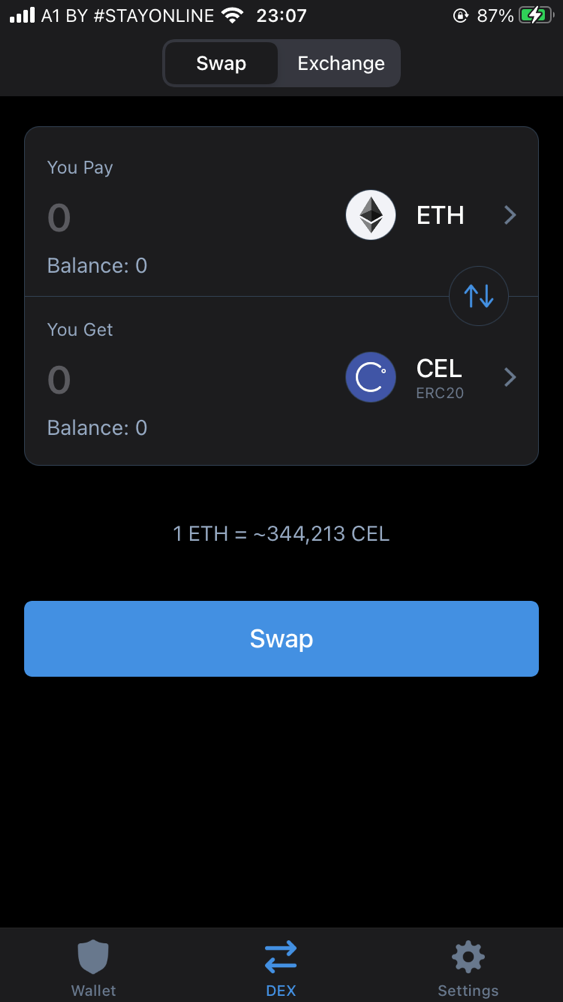


Рисунок 1.2.2.2 – Раздел приложения «DEX»

TrustWallet предоставляет простой способ безопасно хранить такие активы, где токены связаны с публичным адресом и защищены средствами блокчейна.

* 1. Постановка задачи

В рамках дипломного проекта необходимо разработать мобильное приложение для взаимодействия пользователя с его криптоактивами с применением следующего стека технологий:

* *iOS platform*;
* *Swift*;
* *Objective-C*;
* *Foundation;*
* *UIKit;*
* *Swinject*;
* *R.swift*;
* *SnapKit*;
* *Lottie*;
* *QRCodeReader;*
* *ReachabilitySwift;*
* *CryptoSwift*;
* *Keychain;*
* *GIT*.

Основные требования к проекту:

* приложение должно позволять создавать или импортировать криптокошельки Bitcoin, Ethereum, Litecoin криптовалют;
* возможность отправки транзакций поддерживаемых криптовалют между различными адресами;
* возможность просмотра истории транзакций каждого кошелька;
* возможность добавления 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 криптокошельков пользователя;
* возможность выбора времени автоблокировки приложения;
* функция защиты значимых операций пин-кодом.
  1. Выводы по разделу

Проанализировав существующие аналоги, можно сказать, что на сегодняшний день существует некоторое количество программных продуктов, предоставляющий функционал по доступу к криптовалютным активам в той или иной мере, но имеют значительные недостатки, серьезно влияющие на пользовательский опыт. В разрабатываемом программном средстве дипломного проекта они должны быть решены.

В разделе составлена постановка задачи, включающая обязательный набор функций и предполагаемый технологический стек разработки. По результатам исследования были выделены основные требования к хранению конфиденциальных данных пользователя и изучены основные способы защиты выполнения значимых действий. У создаваемого программного средства было обнаружено несколько аналогов, но анализ показывает то, что ни в одном из них нет полного комплекса всех предполагаемых возможностей.

Суммируя все вышеперечисленное можно сказать, что работа, проделанная по аналитическому обзору литературы, является крепким фундаментом для разрабатываемого программного средства. На его основе можно преступать к разработке дизайна пользовательского интерфейса приложения, проектированию архитектуры приложения и программной его реализации,.

Для реализации приложения будут использоваться следующие технологии:

* платформа iOS;
* в качестве основного языка программирования будет выступать язык Swift;
* дополнительный язык программирования, который в некоторых случаях может использоваться как вспомогательный, является Objective-C;
* в качестве архитектуры приложения будет использоваться VIPER паттерн, что позволит удобно организовать код приложения;
* форматом передаваемых данных станет *JSON*.

В дополнение к вышеперечисленному, для разработки приложения существует множество различных библиотек, технологий, утилит и подходов. Некоторые из них были рассмотрены выше.

**5 Руководство пользователя**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 05.00.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Марковский А.Г. |  |  | 5 Руководство пользователя | Лит. | | | Лист | Листов |
| Пров. | | Годун А.В. |  |  |  |  |  | 1 | 1 |
| Консульт. | | Годун А.В. |  |  | 74417049, 2021 | | | | |
| Н. контр. | | Рыжанкова А.С. |  |  |
| Утв. | | Пацей Н.В. |  |  |

**5.1 Первоначальная настройка PIN-кода приложения**

При первом входе в приложение перед пользователем становится задача установки и подтверждение шестизначного пин-кода приложения, который будет использоваться в дальнейшем для входа и ввод которого потребуется для подтверждения значимых действий в приложении. Снимок экрана с первоначальной настройкой приложения представлен на рисунке 5.1.

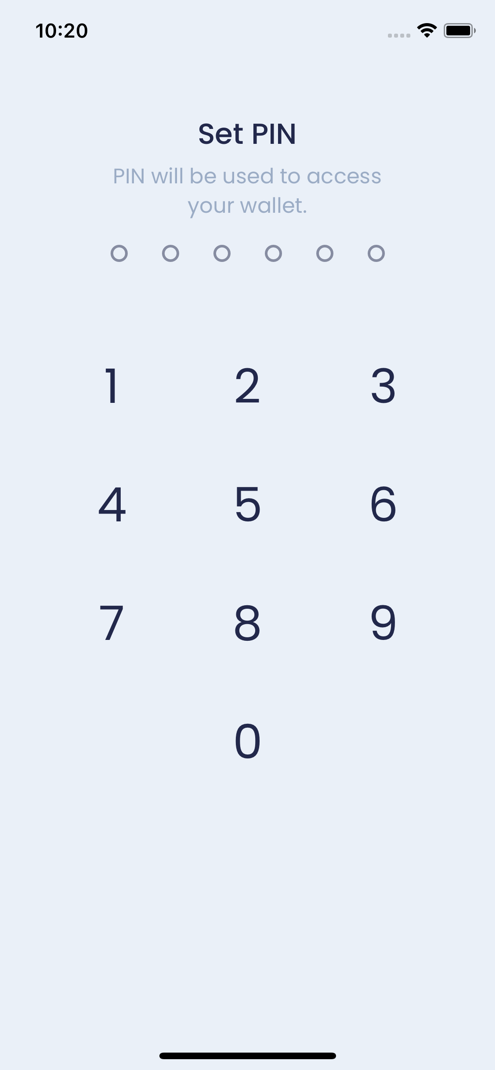


Рисунок 5.1.1 – Первоначальная настройка приложения

Далее, пользователь переходит на главный экран приложения, на котором отображены токены, кошельки которых могут быть созданы или импортированы.

В верхнем правом углу экрана располагается кнопка кнопка, ведущая в настройки приложения. Внешний вид главного экрана приложения изображен на рисунке 5.1.2.

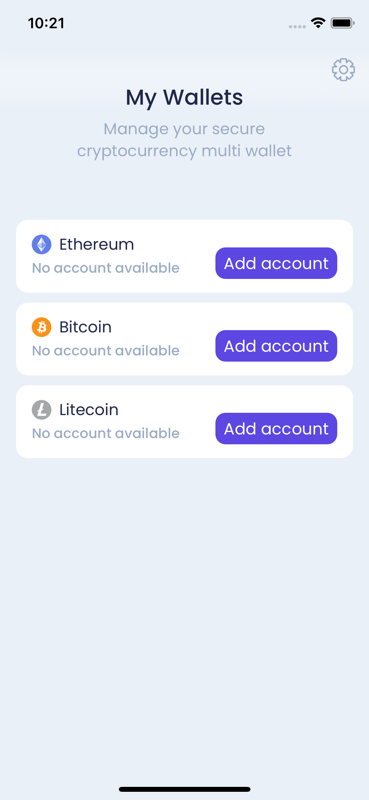


Рисунок 5.1.2 – Главный экран приложения

Нажимая на кнопку «*Add account*» пользователь перенаправляется на экран, изображенный на рисунке 5.2.

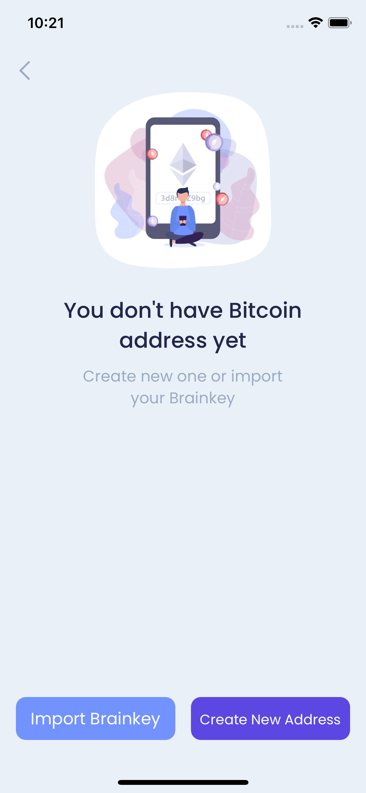


Рисунок 5.1.2 – Главный экран приложения

Данный экран предоставляет возможность выбора между функцией импорта

(отступ слева)и создания криптокошелька с нуля. Выбирая «Import Brainkey» пользователь переходит на экран ввода мнемоника – набора двенадцати слов, который идентифицирует конкретный кошелек. Снимок экрана импорта представлен на рисунке 5.1.3.



Рисунок 5.1.3 – Экран ввода мнемоника при импорте кошелька

В случает, если на экране выбора функции добавления кошелька пользователь выбирает «*Create New Address*», то появляется экран, на котором отображается сгенерированный случайным образом мнемоник. Пользователь имеет возможность скопировать данный набор слов в буфер обмена устройства, чтобы, в дальнейшем, сохранить его в надежном для себя месте и продолжить добавление (рисунок 5.1.4).



Рисунок 5.1.3 – Экран генерации мнемоника при создании кошелька

После завершения процесса добавления кошелька, данный кошелек будет

(отступ слева)отображен на главном экране приложения.

Выбирая определенный добавленный кошелек, производится переход в детали кошелька, где отображаются следующие данные и кнопки возможностей:

- текущий баланс и эквивалент в *USD* валюте;

- адрес текущего кошелька и кнопка копирования адреса в буфер обмена устройства;

- история транзакций в виде с писка с возможностью перехода к деталям каждой транзакции;

- кнопка настроек данного кошелька;

- кнопка, переводящая к сканированию QR-кода;

- кнопка «*Request*», нажатие на которую отображает на экране QR-код текущего кошелька, который может быть сканирован другими устройствами;

- кнопка «*Send*» перенаправляет пользователя на экран создания транзакции.

Снимок экрана деталей кошелька представлен на рисунке 5.1.4

**Заключение**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Марковский А.Г. |  |  | Заключение | Лит. | | | Лист | Листов |
| Пров. | | Годун А.В. |  |  |  |  |  | 1 | 1 |
| Консульт. | | Годун А.В. |  |  | 74417049, 2021 | | | | |
| Н. контр. | | Рыжанкова А.С. |  |  |
| Утв. | | Пацей Н.В. |  |  |

В ходе дипломного проектирования проанализированы существующие программные средства, применяемые для предоставления пользователям доступа к их криптоактивам с возможностью выполнять и отслеживать транзакции, выявлены их достоинства и недостатки. Произведенный анализ позволил определить проблемы анализируемых программных продуктой и найти пути их решения, чтобы применить их в рамках разработки собственного проекта.

Рассмотрены основные термины и определения, которые помогают прояснить необходимые технические аспекты и дают общее представление о работе технологии blockchain.

Рассмотрены различные библиотеки и фреймворки, которые могут быть использованы при разработке программного средства, описаны их достоинства и недостатки. На основе рассмотренных технологий был сделан и обоснован выбор тех библиотек и фреймворков, которые использовались в разработке приложения.

Спроектирована общая архитектура приложения. Описан принцип взаимодействия между компонентами приложения, а также архитектура каждого отдельного модуля. Были описаны принципы и подходы, применяемые для взаимодействия мобильного приложения с серверной API.

Разработано программное средство, представляющее собой мобильное приложение для платформы iOS, назначением которого является предоставление пользователю доступа к его криптоактивам и проведение операций над ними.

Рассмотрен процесс разработки программного средства: даны примеры различных элементов приложения, подробно описано их назначение, а также принцип взаимодействия. Представлены UML-диаграмма вариантов использования, диаграмма классов, блок-схема процесса создания транзакции, блок-схема процесса создания кошелька, диаграмма последовательности процесса создания криптокошелька.

**Список использованных источников**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *БГТУ 00.00.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум. | Подп. | Дата |
| Разраб. | | Марковский А.Г. |  |  | Список использованных источников | Лит. | | | Лист | Листов |
| Пров. | | Годун А.В. |  |  |  |  |  | 1 | 1 |
| Консульт. | | Годун А.В. |  |  | 74417049, 2021 | | | | |
| Н. контр. | | Рыжанкова А.С. |  |  |
| Утв. | | Пацей Н.В. |  |  |

**ПРИЛОЖЕНИЕ А**

**ПРИЛОЖЕНИЕ Б**

**ПРИЛОЖЕНИЕ В**

**ПРИЛОЖЕНИЕ Г**

* 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]

Objective-C

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.

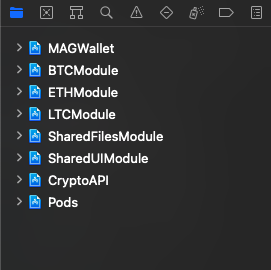


Рисунок 1.4.1 – Общая структура приложения

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.

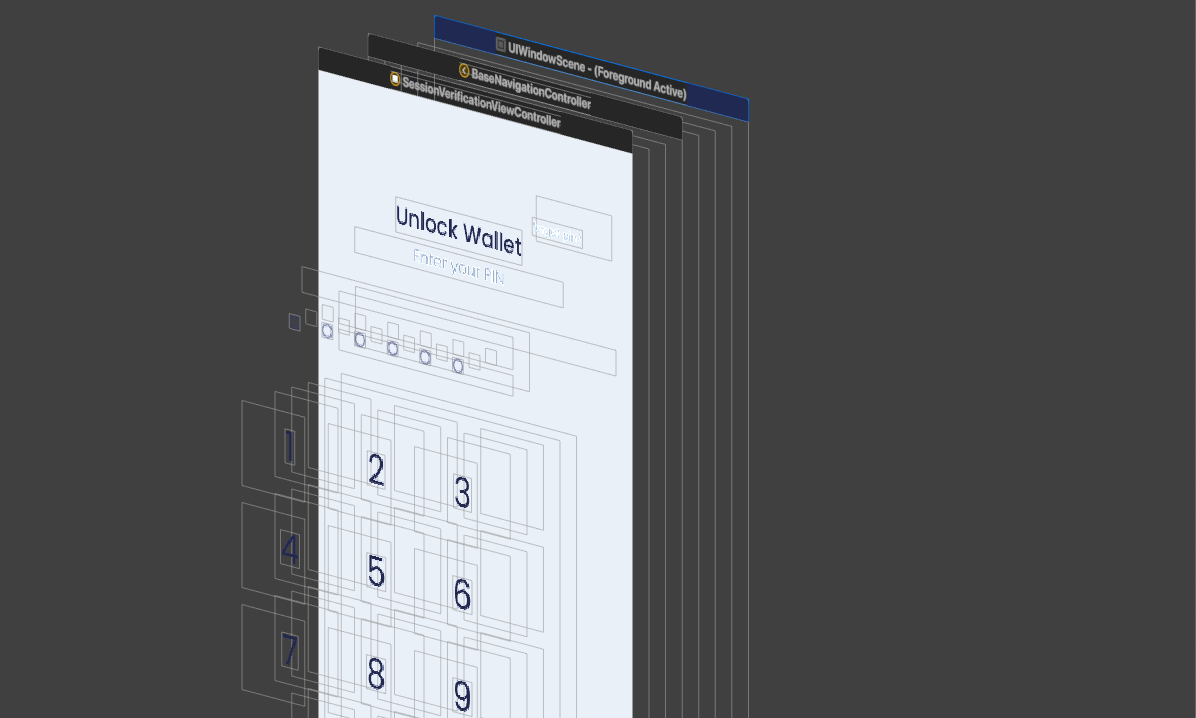


Рисунок 1.5.1 – Структура view ввода пин-кода

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.