Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Расчетно-пояснительная записка к курсовой работе

по курсу «Проектирование автоматизированных систем»

на тему:

**«**Система домашней автоматизации**»**

|  |  |
| --- | --- |
| Руководитель работы: | Выполнил: |
| Доцент кафедры ИТАС | Студент группы 820601 |
| Ломако А.В. | Шведов А.Р. |

Минск 2021

**РЕФЕРАТ**

МОДУЛЬ СТАТИСТИЧЕСКОГО УЧЕТА ЗАГРУЖЕННОСТИ АВТОСТОЯНОК: курсовой проект / А.М. Тихонович. – Минск: БГУИР, 2020, – п.з. –53 с., чертежей (плакатов) – 1 л. формата А1.

Курсовой проект на тему «Модуль статистического учета загруженности автостоянок» разработан с целью повышения степени удовлетворенности клиентов за счет за накопленной информации о клиентском поведении, правильной настройке инструментов маркетинга, оптимизации работы сотрудников компании.

Пояснительная записка к курсовому проекту состоит из введения, 4 разделов, включающих литературный обзор по теме курсового проекта, сравнительный анализ существующих аналогов системы, описание процессов проектирования и программной реализации, а также технико-экономического обоснования разработки и внедрения системы, заключение, список использованных источников и приложение, содержащее листинг кода отдельных классов.

Для разработки автоматизированной информационной системы был выбран объектно-ориентированный язык программирования *ABAP 7.40*. Система представляет собой веб-приложение доступ, к которому, пользователи (работники автостоянок) смогут получать посредствам веб-браузера. Она предназначена для использования в локальной сети либо сети Интернет.

В результате работы над курсовым проектом была спроектирована и разработана система управления бронирования места на автостоянках, дополнение базы данных автостоянками, подготовлено руководство пользователя и выполнено экономическое обоснование разработки и внедрения системы.

Результаты, полученные в ходе курсового проектирования, могут использоваться участниками компаний массового обслуживания. Данная модель имеет все сведения о структуре такой системы, как компоненты системы взаимосвязаны друг с другом. Созданная модель является также хорошим примером создания *UML* диаграмм и может служить примером написания моделей других проектов. Автоматизация позволяет повысить производительность и качество системы, оптимизировать процессы управления, снизить затраты.

Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет информационных технологий и управления

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––––––*

(подпись)

––––––– –––––––––––2021г.

ЗАДАНИЕ

по курсовому проектированию

Студенту    *Шведову Андрею Робертовичу–––––––––––––––––––*

1. Тема проекта   *Система учёта криптовалют.*

2. Срок сдачи студентом законченного проекта– –24.12.2021 г*.–*

3. Исходные данные к проекту Разработать систему домашней автоматизации, позволяющую пользователю использовать умные (IOT) устройства, иметь возможность получать последнюю информацию с установленных датчиков установленных в помещении, быстро оценивать общее состояние системы. Система представляет из себя объеденные в едином умные устройства и датчики, ПО для управления и центральную панель управления. Система не должна передавать информацию в глобальную сеть. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

ВВЕДЕНИЕ

1 СИСТЕМНО-АНАЛИТИЧЕСКАЯ ЧАСТЬ

2 ПРОЕКТНО-РАСЧЕТНАЯ ЧАСТЬ

3 РЕАЛИЗАЦИОННАЯ ЧАСТЬ

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Список графических материалов (с указанием точного назначения необходимых рисунков и графиков)

Структура приложения – 1 лист формата А1.

6. Консультант по проекту (с обозначением разделов проекта) *А.В. Ломако*

7. Дата выдачи задания 14.09 2021 г.–––

8. Календарный график работы над проектом на весь период проектирования

(с обозначением сроков выполнения и трудоемкости отдельных этапов):

17.10. 2021 – 30% 1-я опроцентовка

14.11. 2021 – 60% 2-я опроцентовка

13.12. 2021 – 90% 3-я опроцентовка

16.12.2021 – 100% защита работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель*– –––––––– А.В. Ломако*

(подпись)

Задание принял к исполнению *–––––––\_ А.Р. Шведов*

(подпись)

СОДЕРЖАНИЕ

[Введение 5](#_Toc90247013)

[1 Cистемно-аналитическая часть 6](#_Toc90247014)

[1.1 Описание и анализ объекта автоматизации 6](#_Toc90247015)

[1.2 Постановка задачи 7](#_Toc90247016)

[1.3 Обзор и анализ существующих аналогов 8](#_Toc90247017)

[1.4 Выбранные методологии и инструментальные средства проектирования 13](#_Toc90247018)

[2 Проектно-расчетная часть 16](#_Toc90247019)

[2.1 Концептуальная модель системы 16](#_Toc90247020)

[2.2 Информационное обеспечение 20](#_Toc90247021)

[2.3 Математическое и алгоритмическое обеспечение 26](#_Toc90247022)

[3 Реализационная часть 29](#_Toc90247023)

[3.1 Техническое обеспечение 29](#_Toc90247024)

[3.2 Программное обеспечение 32](#_Toc90247025)

[3.3 Организационное обеспечение 34](#_Toc90247026)

[Заключение 42](#_Toc90247027)

[Список использованных источников 43](#_Toc90247028)

Введение

Автоматизация в наше время охватывает практически все сферы человеческой деятельности, от поддержания микроклимата в помещении, до управления технологическими процессами в серьезных отраслях промышленности. Потребность в автоматизации возникла еще в глубокой древности и постоянно возрастала в процессе развития человеческих возможностей и средств производства. Изобретение любого нового средства передвижения или производства (автомобиль, самолет, паровоз, печатный станок и тому подобное) впоследствии вызывало необходимость усовершенствования, повышения комфорта, безопасности для окружающих и само собой, удобства в управлении.

Цель данного курсового проекта – создание системы домашней автоматизации или так называемой системы «Умный дом». Домашняя автоматизация в современных условиях — чрезвычайно гибкая система, которую пользователь конструирует и настраивает самостоятельно в зависимости от собственных потребностей. Это предполагает, что каждый владелец «умного дома» самостоятельно определяет, какие устройства куда установить и какие задачи они будут исполнять.

Наиболее распространенные примеры автоматических действий в «умном доме» — автоматическое включение и выключение света, автоматическая коррекция работы отопительной системы или кондиционера и автоматическое уведомление о вторжении, возгорании или протечке воды.

1. Cистемно-аналитическая часть
   1. Описание и анализ объекта автоматизации

Криптовалюты – это разновидность альтернативных валют. В отличие от множественных электронных денег и финансовых инструментов, криптовалюты децентрализованы, а это означает, что они не контролируются правительством какого-либо государства или организацией. Также криптовалюты являются одноранговым инструментом (*P2P*), что позволяет отдельным пользователям покупать и продавать товары друг другу напрямую, без посредничества сторонних организаций, таких как, например, крупные банки. Некоторые криптовалюты являются анонимными, но это не является их общей чертой.

Для того, чтобы криптовалюта функционировала независимо от любого централизованного посредника, всем участникам процесса необходимо иметь способ записи и хранения финансовых транзакций, чтобы исключить проблему двойного списания, которая позволяет дважды расплатиться одним и тем же криптографическим токеном, то есть, «купить» товаров на вдвое большую, чем имеющаяся, сумму. При этом проблема должна решаться без использования какого-то центрального сервера и базы, как это делается в банках.

В большинстве существующих криптовалют используется открытый криптографически защищенный распределенный реестр транзакций, который называется «*blockchain*» – «цепочка блоков». Блокчейн – это цепочка из блоков с записями транзакций, которые связаны между собой и защищаются с использованием криптографии. При этом каждый блок содержит свой собственный уникальный криптографический идентификатор, который указывает (связывает) его с предыдущим блоком цепи.

После добавления в блокчейн, блоки уже невозможно изменить без потери данных о всей последующей цепи, что незамедлительно дает другим пользователям знать, что было совершено стороннее вмешательство в обход правил. Это дает возможность просто отказать в использовании модифицированной версии цепочки (потому что без признания модифицированного блока со стороны большинства участников процесса он бесполезен) и продолжать работать с исходной ветвью.

Электронные криптовалютные кошельки могут быть привязаны к блокчейну, чтобы гарантировать, что их баланс соответствует действительности, а новые транзакции проверяются с помощью данных в цепочке блоков для гарантии того, что каждая из них – настоящая и была произведена криптовалютой, которая реально принадлежит плательщику (или его кошельку).

В отличие от привычных нам денег, криптовалюты не хранятся на привычных нам банковских счетах. Вместо этого пользователи криптовалют используют специальные программные или аппаратные кошельки.

Каждый кошелек, вне зависимости от среды функционирования, содержит приватный и публичный ключи. Публичный ключ – это цифровая строка, которую может увидеть каждый. Она состоит из набора букв и символов и используется при отправке средств на кошелек. Приватный ключ – это своего рода ключ безопасности, состоящий из серии случайных чисел, сгенерированных криптографически, которые невозможно взломать. По сути, это шестнадцатеричное число, состоящее из 256 бит или 32 байта, которое создано для подписи переводов.  
Подписанная транзакция отправляется в сеть и после одобрения добирается до получателя.  
 Криптокошельки делятся на кастодиальные и некастодиальные. Некастодиальный кошелек  – это вид децентрализованного кошелька, в котором клиенту принадлежат его личные ключи. Пользователь получает файл с личными ключами и должен написать мнемоническую фразу, с помощью которой он может восстановить доступ к своим средствам. Наличие личных ключей означает, что пользователь имеет полный контроль над средствами. Однако, следует иметь в виду, что полный контроль над деньгами также означает, что лишь потребитель несет полную ответственность за свои средства. Напротив, кастодиальный кошелек – это вид цифрового кошелька, в котором личные ключи и резервное копирование всех данных, то есть, защита, средств, находится на стороне разработчика.

С точки зрения пользователя, безусловно, некастодиальный кошелек является наиболее приемлемым вариантом, ведь в таком случае, злоумышленники не смогут похитить активы, однако, если уповать на человеческий фактор, и допускать вероятность совершения ошибок, то хранение средств в кастодиальных кошельках (не во всех случаях), дает надежду на восстановление утерянных средств.

* 1. Постановка задачи

Целью курсового проекта является создание системы учёта криптовалют. Система должна позволять пользователю создать криптокошелёк, зашифрованный секретной фразой, состоящей из двадцати четырёх слов. Криптокошелёк должен иметь функционал создания приватных ключей для кошельков некоторых криптовалют, а также обеспечить безопасное хранение созданных приватных ключей в зашифрованном с помощью ранее созданной пользователем секретной фразы виде на устройстве. Пользователь должен иметь возможность получить как конкретный приватный ключ от кошелька каждой криптовалюты, так и зашифрованный файл, содержащий все созданные приватные ключи, для входа в данный криптокошелек на другом устройстве.

Система должна предоставлять функционал для создания транзакций, просмотра текущего баланса и просмотра истории транзакций для каждой криптовалюты.  
 Система должна представлять из себя мобильное приложение, доступное для мобильных операционных систем *iOS* версии 13.0 и выше, а также *Android* версии 8.0 и выше.

* 1. Обзор и анализ существующих аналогов

Программный кошелек – один из самых распространенных методов хранения, которые подразумевает под собой мультиплатформенность, быстрый доступ к своим средствам, и относительную безопасность. Следует понимать, что каждый программный кошелек, так или иначе, привязан к своему физическому носителю, и его утрата, в большинстве случаев, приведет к безвозвратной потере средств. Рассмотрим наиболее популярные программные криптокошельки на данный момент.

* + 1. *Blockchain Wallet*.Blockchain Wallet – это один из самых популярных цифровой кошелек, который позволяет пользователям хранить свои криптовалюты и управлять ими в полной мере. Кошелек поддерживается компанией Blockchain, занимающейся разработкой программного обеспечения, основанной Питером Смитом и Николасом Кэри. Кошелек на блокчейне позволяет осуществлять переводы в криптовалюте и конвертировать их в валюту. Криптокошелёк представлен на рисунке 1.

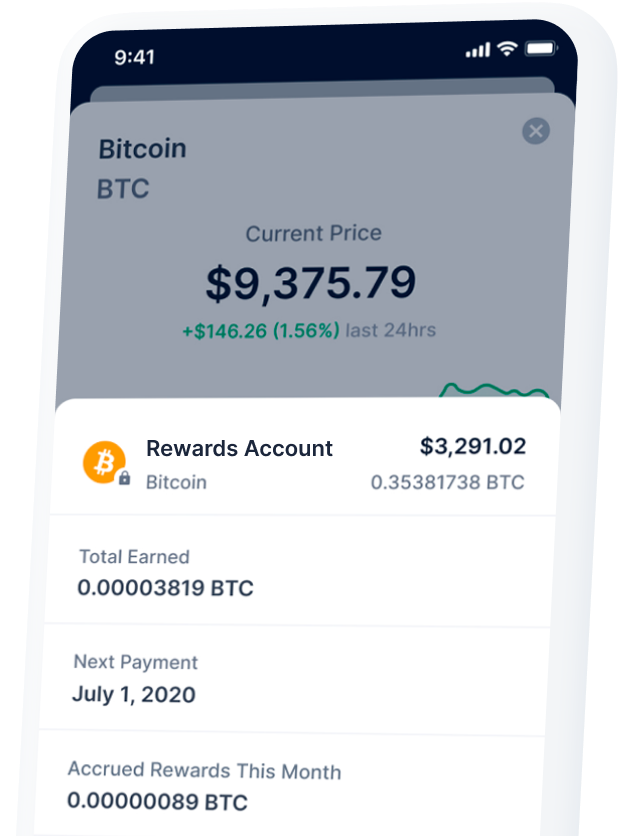


Рисунок 1 – Криптокошелёк

*Blockchain Wallet*

К плюсам данного кошелька относятся:

* поддержка мобильного приложения с хорошо проработанным пользовательским интерфейсом;
* поддержка веб-интерфейса;
* поддержка ведущих криптовалют;
* открытый исходный код.

К минусам данного кошелька относятся:

* наличие возможности вывода всех средств, если злоумышленники смогут получить доступ к почте пользователя;
* хранение приватных ключей на стороне разработчика.

*Blockchain Wall*et является самым популярным местом для хранения средств, однако, уровень безопасности оставляет желать лучшего. *Blockchain* позволяет пользователям хранить средства в относительной безопасности, и максимально нацелены на удобство. Этот вариант можно рассмотреть, если пользователь не собирается хранить больших сумм, и является новичком в крипто-пространстве, для которого важна простота функционала.

* + 1. *Coinbase Wallet. Coinbase Wallet* – это приложение, которое позволяет пользователям хранить свою собственную криптовалюту и взаимодействовать с блокчейном посредством децентрализованного приложения. Этот кошелек сочетает в себе универсальность и поддержку, предлагаемую компанией *Coinbase*. В этом кошельке пользователь может не только хранить *ETH* и токены, стандарта *ERC*, основанные на сети *Ethereum*, но также и взаимодействовать с криптовалютами *BTC*, *BCH* и *LTC*.  
       Данный криптокошелёк представлен на рисунке 2.

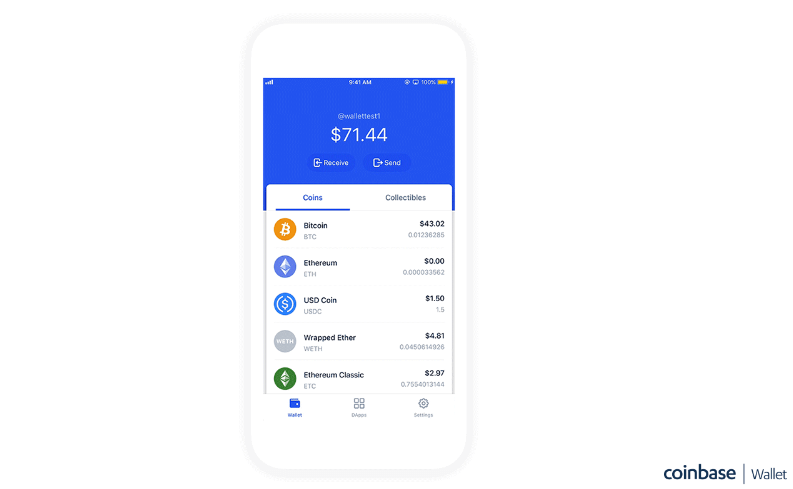


Рисунок 2 – Криптокошелёк *Coinbase Wallet*

Плюсам данного являются:

* поддержка мобильного приложения с хорошо проработанным пользовательским интерфейсом;
* поддержка веб-интерфейса;
* поддержка ведущих криптовалют.

Минусами данного кошелька являются:

* хранение приватных ключей на стороне разработчика;
* отсутствие открытого исходного кода.

Кошелек *Coinbase* представляет собой отличную альтернативу хранению своих средств. Во-первых, он генерирует сид для обеспечения безопасности, во-вторых, позволяет защитить приложение от использования считывателя отпечатков пальцев. В общем, это очень простое и удобное в использовании приложение, но с множеством инструментов. Кроме того, этот кошелек поддерживается компанией *Coinbase*. С другой стороны, он может доставлять определенные неудобства. При открытии некоторых децентрализованных приложений, браузер может зависнуть, что не всегда позволит произвести сделку в считанные секунды, как того будут требовать заданные пользователям стандарты качества.

* + 1. *Trust Wallet. Trust Wallet* – это приложение для мобильных устройств, которое позволяет отправлять, получать и хранить токены криптовалюты в блокчейне Ethereum. Портфолио с открытым исходным кодом, разработанное с упором на простоту, призвано предоставить платформу, которую легко настроить и использовать. *Trust Wallet* позволяет пользователям хранить свои приватные ключи на своих устройствах и предоставляет функцию резервного копирования для простого восстановления. Данный криптокошелёк представлен на рисунке 3.

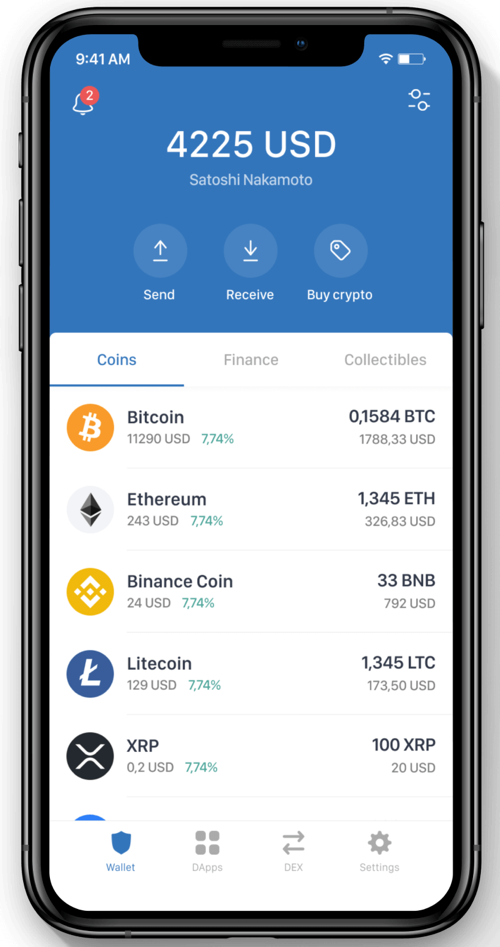


Рисунок 3 – Криптокошелёк *Trust Wallet*

Плюсам данного являются:

* возможность хранения любых *ERC* токенов;
* хранения приватных ключей на устройстве пользователя;
* возможность работы с децентрализованными приложениями.

К минусам данного кошелька можно отнести:

* отсутствие поддержки многих криптовалют;
* отсутствие двухфакторной аунтификации;
* плохой дизайн интерфейса, плохо ориентированный на пользователя.

*Trust Wallet* включает в себя встроенный браузер *Web3*, который позволяет легко исследовать децентрализованные интернет-приложения. Это приложение соответствует стандартам качества и безопасности и оптимизированы для работы на высоком уровне. *Trust Wallet* отлично подойдет для тех пользователей, которые не слишком придирчивы к функционалу, и не имеют надобности держать большой портфель различных токенов не *ERC*-стандарта.

* + 1. *Atomic Wallet. Atomic Wallet* – уникальный криптовалютный кошелек, который позволяет управлять более чем тремястами различными крипто активами, в дополнение к возможности добавления пользовательских токенов *ERC20*. Еще одна характеристика, которой может похвастаться этот кошелек, заключается в том, что в нем есть возможность проводить атомарные свопы, которые основаны на реализации обменов без посредников с использованием технологии смарт-контрактов. Данный криптокошелёк представлен на рисунке 4.

**

Рисунок 4 – Критокошелёк *Atomic Wallet*

Преимуществами данного криптокошелька являются:

* поддержка мобильного приложени;
* поддержка дестком-клиента;
* хранение приватных ключей на устройстве пользователя;
* широкий спектр поддерживаемых криптовалют;
* открытый исходный код.

К недостатком данного кошелька можно отнести:

* отсутствие веб-клиента.

*Atomic Wallet* полностью безопасен для хранения средств. Данный кошелек имеет открытый исходный код, что значительно снижает вероятность возникновения ошибок или внедрения скрытых уязвимостей в программу. Более того, важные данные, такие как пароли, зашифрованы и хранятся только локально на устройстве конечного пользователя.

* 1. Выбранные методологии и инструментальные средства проектирования

1.4.1 Методология *IDEF0*. Согласно методологии *IDEF0*, модель объекта управления строится в виде диаграмм, состоящих из блоков и стрелок. Блоки (*Activities*) обозначают работы (функции), выполняемые на объекте управления, а стрелки (*Arrows*) – материальные объекты и информацию, обрабатываемые в ходе выполнения работ или используемые для их выполнения. Строится набор диаграмм, последовательно детализирующих процессы функционирования объекта управления.

Общий вид блока диаграммы, построенной согласно методологии *IDEF0* (*IDEF0*-диаграммы, или *IDEF0*-модели), представлен на рисунке 5.

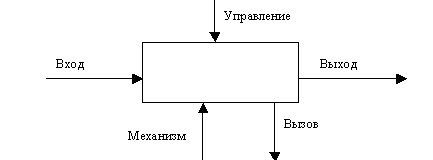


Рисунок 5 – Общий вид блока диаграммы *IDEF0*

Смысл стрелок, используемых на *IDEF0*-диаграмме, следующий.

Вход – материальные объекты (например, сырье) или информация, обрабатываемые в процессе выполнения работы для получения результата (выхода).

Управление – правила выполнения работы (методы, стандарты и т.д.).

Механизм – ресурсы для выполнения работы (персонал, станки, оборудование и т.д.).

Выход – результат выполнения работы (готовая продукция, результаты анализа информации и т.д.).

Вызов – указатель на другую модель.

1.4.2 Методология *DFD***.** *DFD* (англ. *Data flow diagram* – диаграмма потоков данных) – методология графического структурного анализа, описывающая внешние по отношению к системе источники, и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.

*DFD*–диаграммы включают следующие элементы:

* работы (*Activities*). Обычно они обозначают операции по обработке данных;
* стрелки (*Arrows*), обозначающие перемещение данных или объектов в процессе их обработки;
* внешние ссылки (*External* *references*) – входы и выходы модели. Используются для обозначения источников данных, обрабатываемых в анализируемой системе, или приемников результатов обработки данных;
* хранилища данных (*Data* *stores*). Используются для обозначения баз данных, массивов, картотек (при ручной обработке данных) и т.д.

Таким образом диаграммы потоков данных показывают, как каждый процесс преобразует свои входные данные в выходные, и выявляют отношения между этими процессами. *DFD* представляет моделируемую систему как сеть связанных работ. Стрелки в *DFD* показывают, как объекты (включая и данные) реально перемещаются от одного действия к другому. Диаграммы потоков данных моделируют систему как набор действий, соединенных друг с другом стрелками.

1.4.3 Методология *IDEF3* – способ описания процесса с использованием структурированного метода, позволяет представить положение вещей в предметной области как упорядоченную последовательность событий с одновременным описанием объектов, имеющих непосредственное отношение к процессу. *IDEF3* также может быть использован как метод проектирования бизнес-процессов. *IDEF3*-моделирование органично дополняет традиционное моделирование с использованием стандарта методологии *IDEF0*.

*IDEF3*-диаграммы включают следующие элементы:

* единицы работы (*UOW* – *Units of work*), аналогичные работам на *IDEF0*-диаграммах;
* стрелки (*Arrows*), обозначающие порядок выполнения работ;
* перекрестки (*Junctions*), отображающие логические связи между работами.

1.4.4 *UML* (англ. *Unified* *Modeling* *Language* – унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

*UML* является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой *UML*-моделью. *UML* был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. *UML* не является языком программирования, но на основании *UML*-моделей возможна генерация кода.

1. ПРОЕКТНО-РАСЧЕТНАЯ ЧАСТЬ
   1. Концептуальная модель системы

Проектируемая система разрабатывается для пользователей, которые хотят вести учёт криптовалют. Для этого разрабатывается мобильное приложение, в котором пользователь сможет создавать кошельки в блокчейнах различных криптовалют, просматривать информацию о балансе и истории транзакций на данных кошельках, а так же создавать новые транзакции.

* + 1. Построение модели начинается с описания функционирования системы в целом в виде контекстной диаграммы. [1] Исходя из перечня входных и выходных данных, а также основываясь на требованиях к системе, была разработана контекстная диаграмма по методологии *IDEF0*. Контекстная диаграмма представлена на рисунке 6.

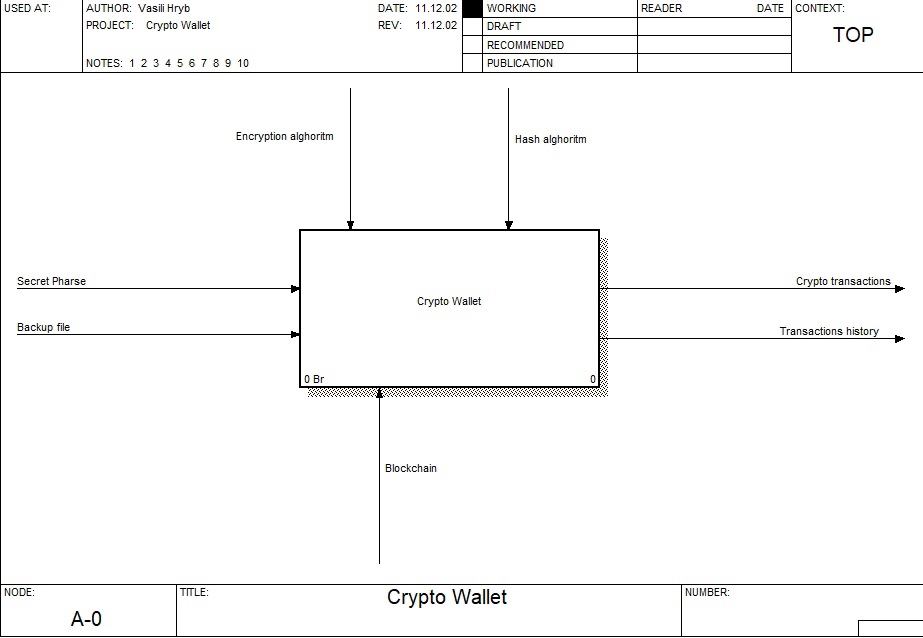


Рисунок 6 – Контекстная диаграмма системы учёта криптовалют

Контекстная диаграмма представляет собой общее описание работы системы учёта криптовалют и ее взаимодействия с внешними элементами или данными. На приведённой контекстной диаграмме представлены виды входной и выходной информации системы, а также механизмы и элементы управления. На вход поступает следующая информация:

* секретная фраза – фраза, состоящая из 24 слов. Данной фразой будет зашифрован резервный файл, содержащий приватные ключи от кошельков всех криптовалют, созданных пользователем в системе;
* резервный файл – файл, хранящий в зашифрованном виде приватные ключи от кошельков криптовалют. Данный файл необходим для восстановления доступа к созданным в системе ранее криптовалютным кошелькам при повторной авторизации.

В качестве выходной информации служат следующие данные:

* транзакции – созданные пользователем в системе транзакции в блокчейнах различных криптовалют;
* история транзакций – информация о транзакциях криптокошельков, созданных в системе, а также их баланса.

В качестве механизмов служат блокчейны криптовалют, поддерживаемых системой. В них хранится информация о всех созданных транзакциях и балансах кошельков.

Элементы управления представлены следующими составляющими:

* алгоритм шифрования – алгоритм, с помощью которого в системе будет зашифрован резервный файл, содержащий приватные ключи созданных криптокошельков;
* алгоритм хеширования – алгоритм, по которому будет сгенерирован приватный ключ из секретной фразы, используемый для шифрования резервного файла. Хеширование секретной фразы осуществляется для повышения криптографической стойкости системы.

2.1.2 Для более подробного описания и детализации рассматриваемого процесса используется диаграмма декомпозиции. Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой.

Контекстная диаграмма была разбита на три последовательных функциональных блока:

* «авторизация» – функциональный блок, отвечающий за предоставление доступа к системе, формирование персональных данных для незарегистрированных пользователях, проверка и верификация данных для входа в систему;
* «генерация кошелька» – функциональный блок, отвечающий за создание кошельков для поддерживаемых системой криптовалют;
* «управление транзакциями» – функциональный блок, отвечающий за просмотр истории транзакций, баланса кошельков и создание новых транзакций в блокчейнах поддерживаемых системой криптовалют.

Диаграмма декомпозиции контекстной диаграммы представлена на рисунке 7.

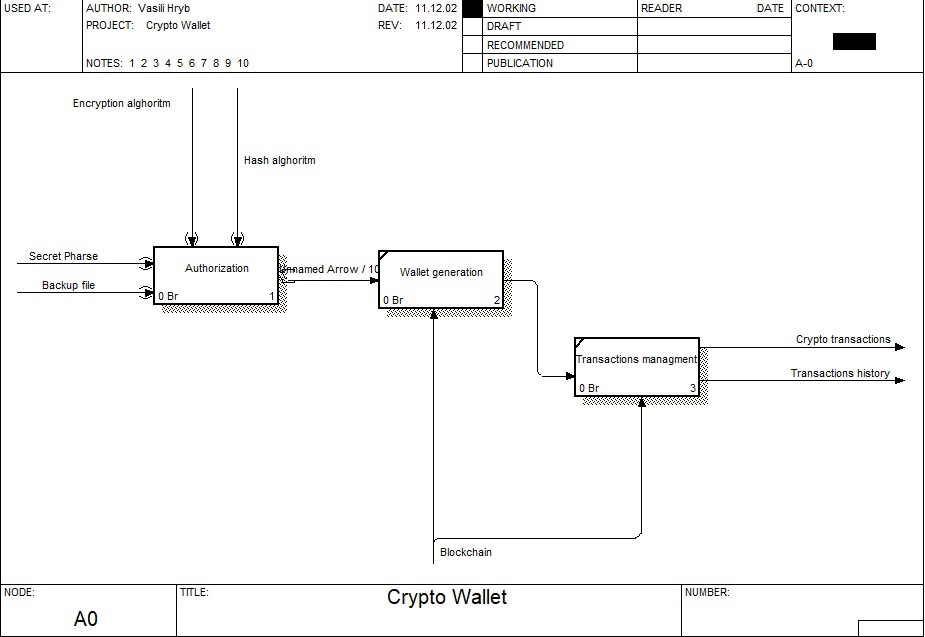


Рисунок 7 – Диаграмма декомпозиции

2.1.3 Диаграмма вариантов использования (*Use case diagram*, диаграмма прецедентов) – диаграмма, на которой отражены отношения, существующие между действующими лицами и вариантами использования. [2]

Основная задача диаграммы – представлять собой единое средство, дающее возможность заказчику, конечному пользователю и разработчику совместно обсуждать функциональность и поведение системы.

В рамках данного проекта рассматривается взаимодействие действующего лица «Пользователь» со системой. Разработанная диаграмма вариантов использования представлена на рисунке 8.

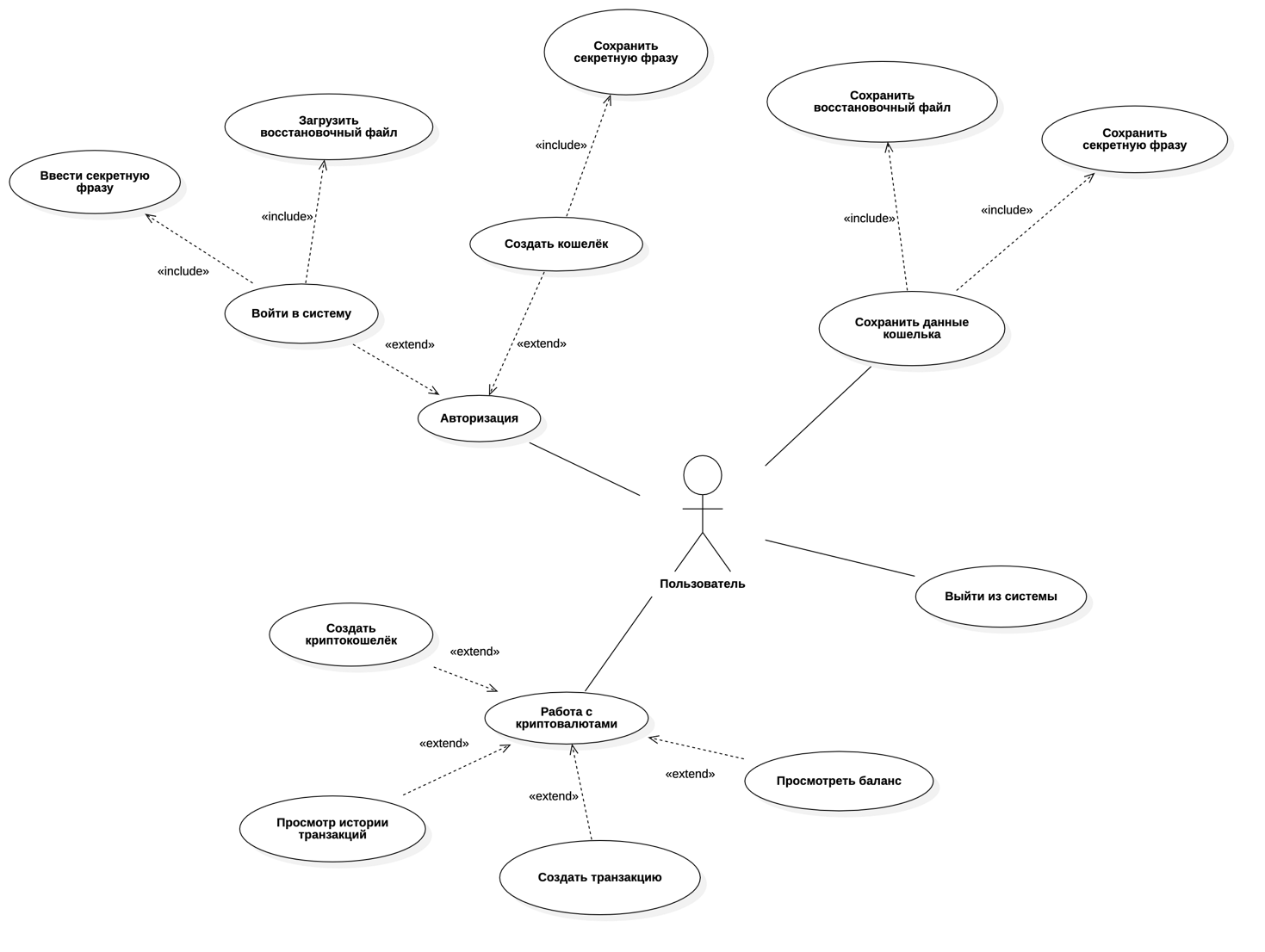


Рисунок 8 – Диаграмма вариантов использования

Система имеет следующие варианты использования:

* «Авторизация». Данный вариант использования отвечает за получения доступа к системе новым и существующим пользователям. Его расширяют прецеденты «Войти в систему» для существующих пользователей и «Создать кошелёк» для новых пользователей. Прецедент «Войти в систему» включает в себя ввод секретной фразы и загрузку резервного (восстановочного) файла. Прецедент «Создать кошелёк» включает в себя сохранения созданной секретной фразы, состоящей из 24 слов;
* «Сохранить данные кошелька». Данный вариант использования включает в себя сохранение секретной фразы и резервного файла;
* «Работа с криптовалютами». Данный вариант использования подразумевает учёт и управление кошельками всех поддерживаемых системой криптовалют. Он расширяется прецедентами «Создать криптокошелёк», «Просмотр истории транзакций», «Создать транзакцию» и «Просмотреть баланс».
  1. Информационное обеспечение
     1. Для подробного описания процесса «Авторизация» была использована методология *DFD*. Разработанная *DFD* диаграмма представлена на рисунке 9.

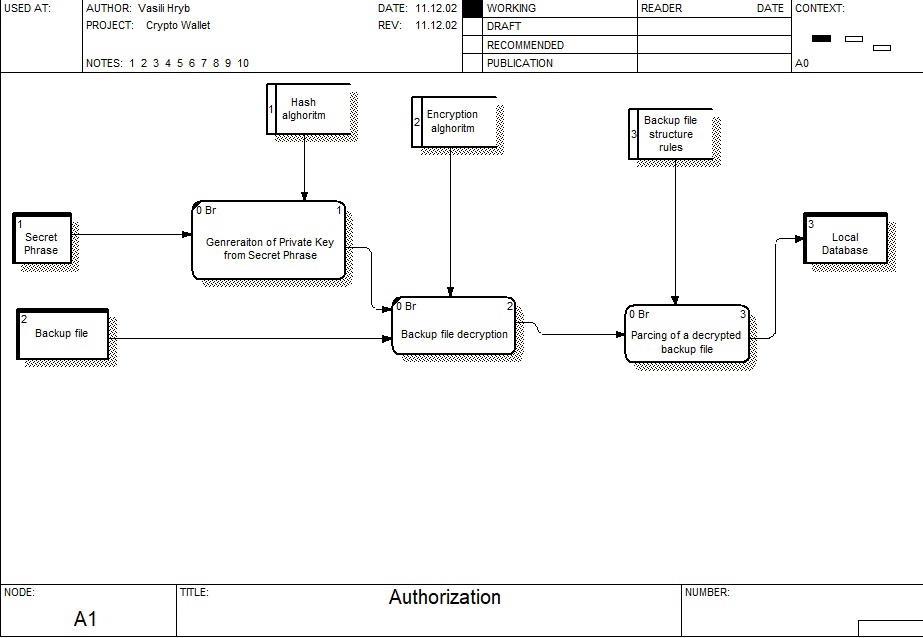


Рисунок 9 – *DFD* диаграмма для процесса «Авторизация»

На данной диаграмме представлены следующие работы:

* «Генерация приватного ключа из секретной фразы»;
* «Расшифровка резервного файла»;
* «Разбиение расшифрованного резервного файла».

Также на диаграмме представлены следующие внешние объекты.

* «Секретная фраза»;
* «Резервный файл»;
* «Локальная база данных».

Кроме этого, на диаграмме представлены следующие хранилища данных:

* «Алгоритм хеширования»;
* «Алгоритм шифрования».

Вначале процесса происходит генерация приватного ключа из секретной фразы путем хеширования по заданному алгоритму. Хеширование секретной фразы необходимо для повышение криптографической устойчивости системы.

После того, как приватный ключ получен, он используется для расшифровки резервного файла по заданному алгоритму шифрования. Резервный файл хранит приватные ключи от криптокошельков, созданных пользователем.

После расшифрованные данные разбиваются по заданной структуре резервного файла для получения приватных ключей от кошельков каждой из криптовалют, а также идентификатора соответствующих криптовалют. Полученный данные сохраняются в локальную базу данных.

2.2.2 Для описания работы «Разбиение расшифрованного резервного файла» с использованием структурированного метода, позволяющего представить положение вещей в предметной области как упорядоченную последовательность событий с одновременным описанием объектов, имеющих непосредственное отношение к процессу, была использована методология *IDEF*3.

Разработанная диаграмма представлена на рисунке 10. Диаграмма приводит описание процессов, происходящих в работе «Разбиение расшифрованного резервного файла». На данной диаграмме приведено семь процессов: «Открыть файл», «Извлечь текст из файла», «Извлечь идентификаторы криптовалют», «Извлечь приватные ключи», «Проверить результат», «Выдать ошибку», «Использовать извлечённые данные».

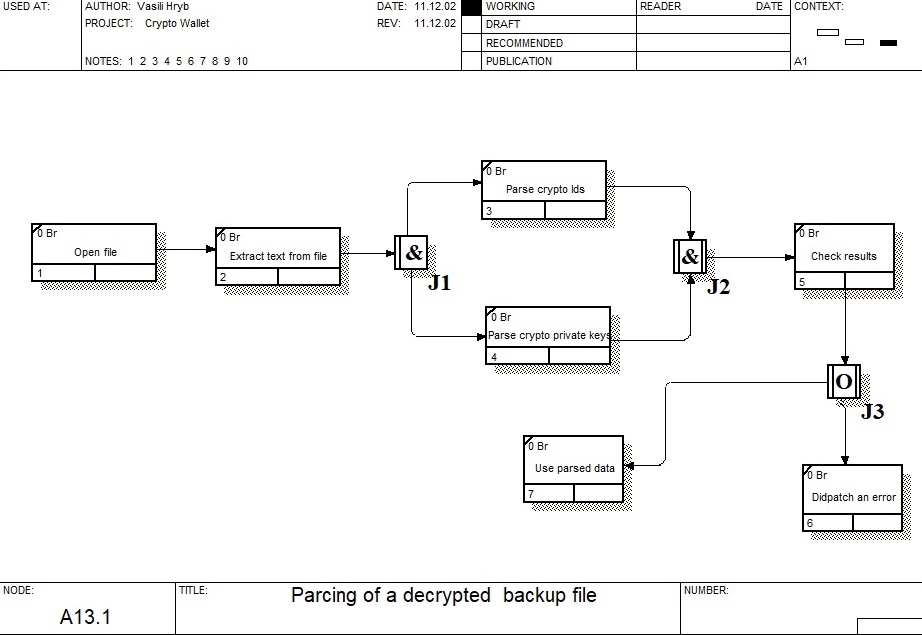


Рисунок 10 – Диаграмма *IDEF*3 для работы «Разбиение расшифрованного резервного файла»

2.2.3 Проектирование структуры локальной базы данных. Проектирование баз данных – процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основной этап процесс проектирования базы данных информационной системы – преобразование требований к данным в структуры данных. На выходе получаем СУБД-ориентированную структуру базы данных и спецификации прикладных программ. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.

Для реляционных БД (т.е. БД, где данные представлены в виде таблиц) логическая модель включает:

* описание таблиц;
* описание связей между таблицами;
* описание атрибутов.

При реализации автоматизированной системы необходимо хранение большого объема информации, в связи с чем возникает необходимость в использовании базы данных, которую изначально необходимо спроектировать. Основная цель проектирования базы данных – это сокращение избыточности хранимых данных, и впоследствии, экономия объема используемой памяти, устранение возможности возникновения противоречий из-за хранения в разных местах информации об одном и том же объекте.

Основные задачи проектирования баз данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности данных (правильности их содержания): исключение противоречий в содержании данных, исключение потерь и т.д.

Разработанная схема структуры локальной базы данных разрабатываемой системы представлена на рисунке 11.

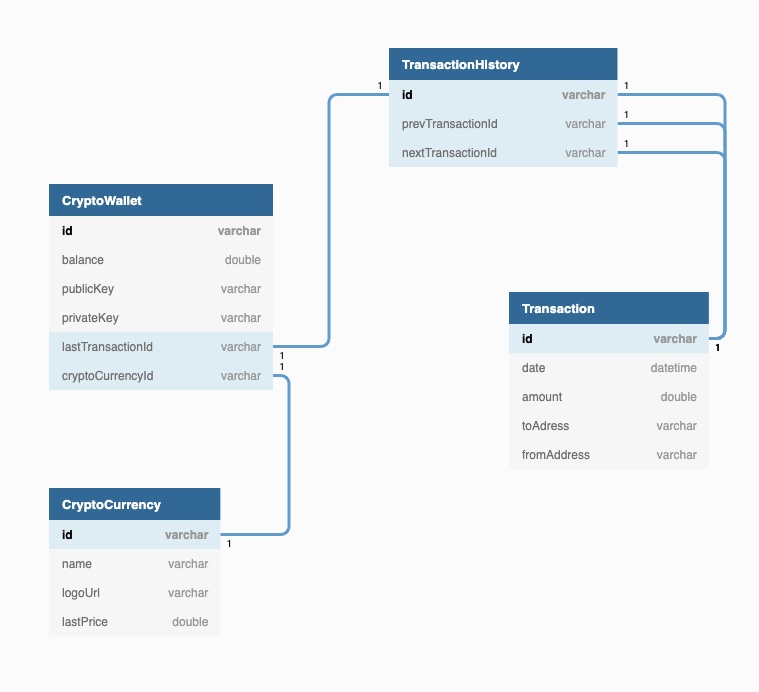


Рисунок 11 – Структура локальной базы данных

В структуре базы данных есть четыре различных сущностей. Структура сущности «*СryptoCurrency*» представлена в таблице 1.

Таблица 1 – Структура сущности «*СryptoCurrency*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *varchar* | Уникальный идентификатора криптовалюты, первичный ключ |
| *name* | *varchar* | Имя криптовалюты |
| *logoUrl* | *varchar* | Ссылка на загрузку логотипа криптовалюты |
| *lastPrice* | *double* | Цена криптовалюты |

Структура сущности «*Transaction*» представлена в таблице 2.

Таблица 2 – Структура сущности «*Transaction*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *varchar* | Уникальный идентификатора транзакции, первичный ключ |
| *date* | *datetime* | Дата транзакции |
| *amount* | *doulbe* | Количество криптовалюты, перечисленной в транзакции |
| *toAddress* | *varchar* | Публичный ключ кошелька-получателя |
| *fromAddress* | *varchar* | Публичный ключ кошелька-отправителя |

Структура сущности «*TransactionHistory*» представлена в таблице 3.

Таблица 3 – Структура сущности «*TransactionHistory*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *varchar* | Уникальный идентификатора транзакции, первичный ключ |
| *prevTransactionId* | *varchar* | Уникальный идентификатор предыдущей транзакции |
| *nextTransactionId* | *varchar* | Уникальный идентификатор следующей транзакции |

Структура сущности «*CryptoWallet*» представлена в таблице 4.

Таблица 4 – Структура сущности «*CryptoWallet*»

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| *id* | *varchar* | Уникальный идентификатора криптовалютного кошелька, первичный ключ |
| *balance* | *doulbe* | Баланс криптовалютного кошелька |
| *publicKey* | *varchar* | Публичный ключ криптовалютного кошелька |
| *privateKey* | *varchar* | Приватный ключ криптовалютного кошелька |
| *lastTransactionId* | *varchar* | Уникальный идентификатор последней транзакции, совершенной для данного криптовалютного кошелька |
| *cryptoCurrencyId* | *varchar* | Уникальный идентификатор криптовалюты, которую хранит данный криптовалютный кошелёк |

2.2.4 Для описания взаимодействия элементов системы была создана диаграмма классов. Диаграмма классов (*Class Diagram*) определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами.

В разрабатываемой системе можно выделить 5 классов:

* «*CryptoWallet*» – класс, отвечающий за управления криптовалютными кошельками и предоставление информации о них;
* «*Transaction*» – класс, содержащих информацию о транзакции и имеющий методы для предоставления данных о предыдущих транзакций и подсчитывания текущего баланса на основе историй всех транзакций;
* «*TransactionManager*» – класс, имеющий функционал для создания новых транзакций и получение информации о последней совершенно транзакции для заданного криптовалютного кошелька;
* «*CryptoManager*» – класс, предоставляющий функционал для хеширования и шифрования информации;
* «BlockchainManager» – класс, предоставляющий доступ к блокчейну заданной криптовалюты.

Созданная диаграмма представлена на рисунке 12.

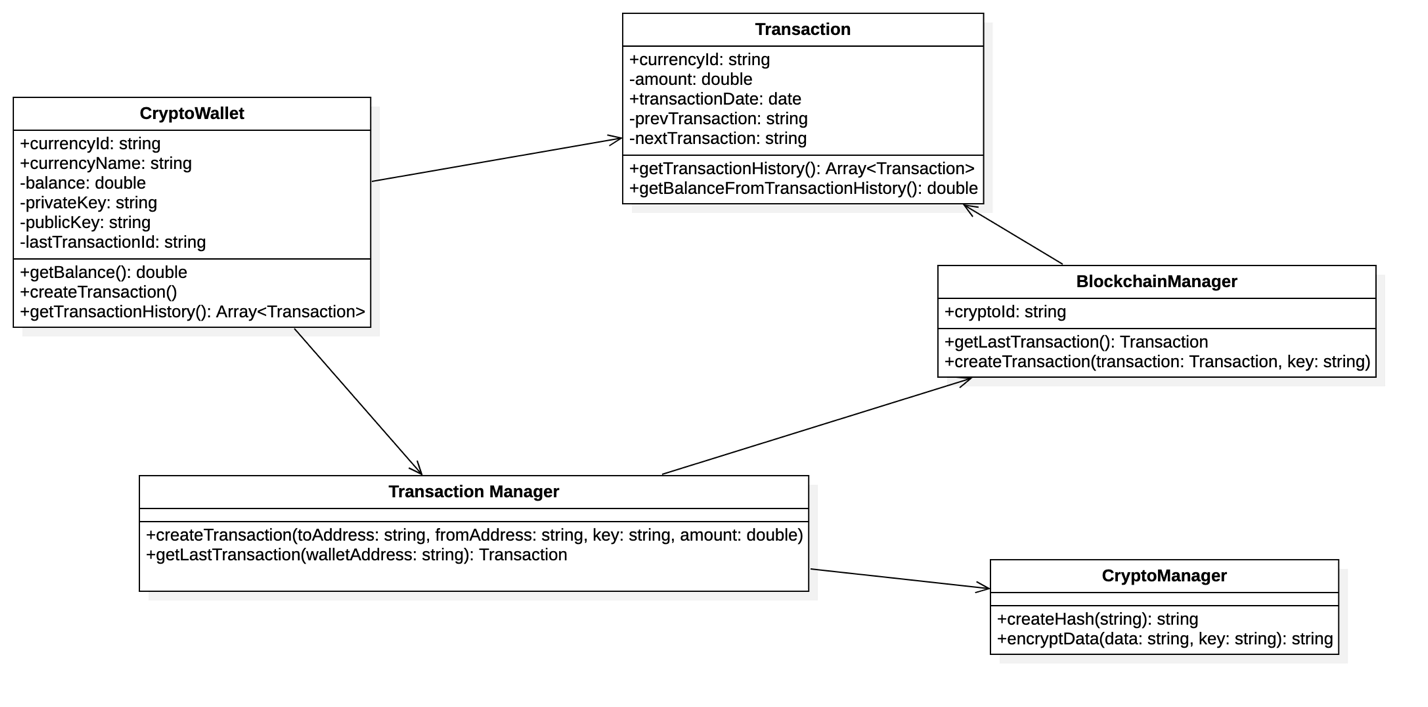


Рисунок 12 – Диаграмма классов системы учёта криптовалют

* 1. Математическое и алгоритмическое обеспечение
     1. При моделировании поведения проектируемой или анализируемой системы возникает необходимость не только представить процесс изменения ее состояний, но и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке *UML* используются так называемые диаграммы деятельности (*Activity Diagram*). Применяемая в них графическая нотация во многом похожа на нотацию диаграммы состояний, поскольку на диаграммах деятельности также присутствуют обозначения состояний и переходов.

Самым большим достоинством диаграмм деятельностей является поддержка параллелизма. Благодаря этому они являются мощным средством моделирования потоков работ и, по существу, параллельного программирования. Самый большой их недостаток заключается в том, что связи между действиями и объектами просматриваются не слишком четко.

Диаграмма деятельности разрабатываемой системы представлена на рисунке 13.

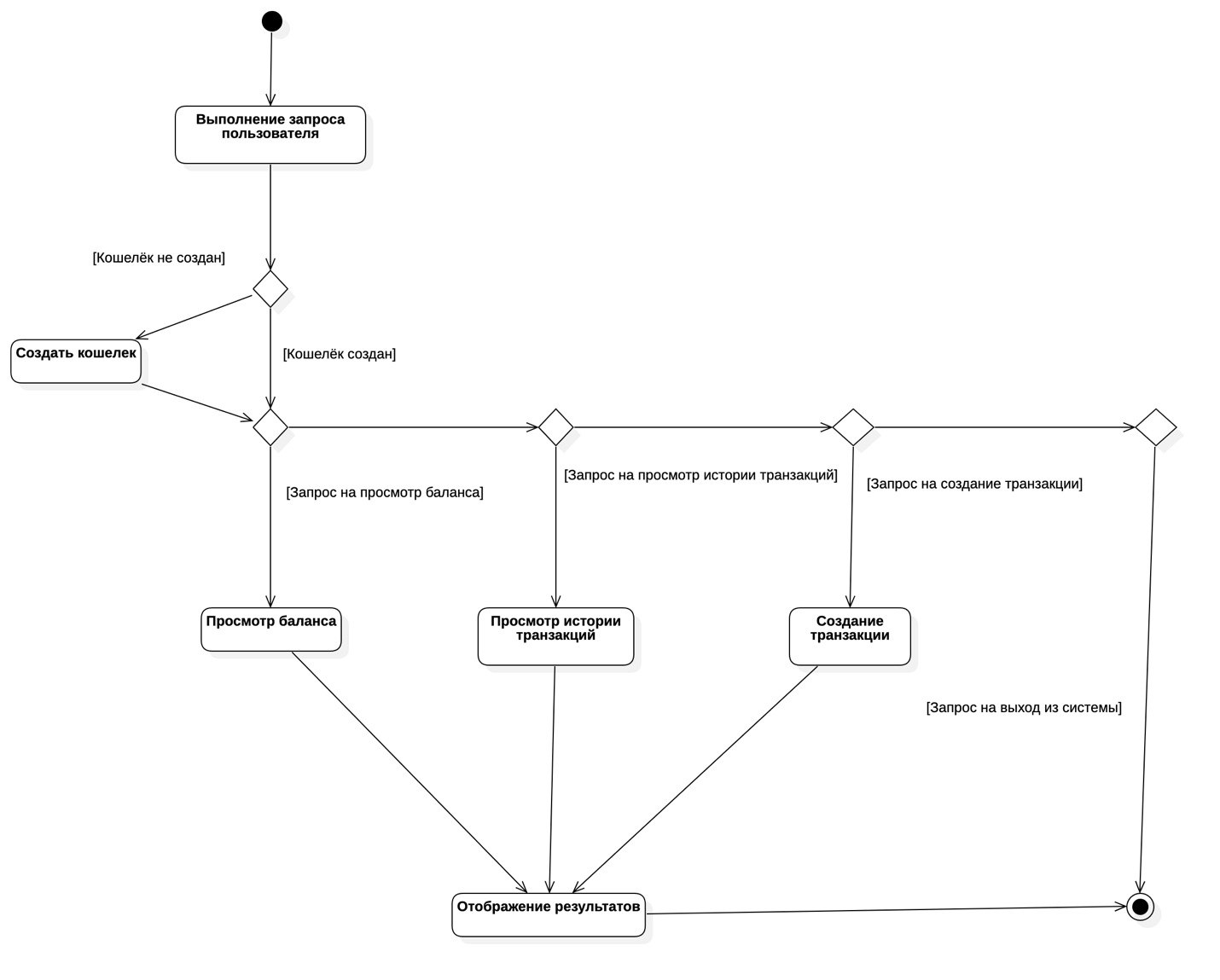


Рисунок 13 – Диаграмма деятельности системы учёта криптовалют

* + 1. Одной из ключевых функция разрабатываемой системы является управление кошельками криптовалют, в частности, просмотр текущего баланса криптокошелька. Для иллюстрации данного процесса была разработана схема алгоритма обновления баланса криптовалютного кошелька.

Блок-схема – это схематичное представление процесса, системы или компьютерного алгоритма. Блок-схемы часто применяются в разных сферах деятельности, чтобы документировать, изучать, планировать, совершенствовать и объяснять сложные процессы с помощью простых логичных диаграмм. Для построения блок-схем применяются прямоугольники, овалы, ромбы и некоторые другие фигуры (для обозначения конкретных операций), а также соединительные стрелки, которые указывают последовательность шагов или направление процесса. Блок-схемы варьируются от незамысловатых, нарисованных вручную до подробных, составленных на компьютере диаграмм со множеством шагов и процессов. Если учесть все возможные вариации, блок-схемы можно признать одним из самых распространенных видов схем во всем мире. Они широко используются в разных сферах как технической, так и нетехнической направленности. Иногда блок-схемы получают более узкоспециальные названия, например, схема процесса, схема рабочего процесса, функциональная блок-схема, моделирование бизнес-процессов, модель и нотация бизнес-процессов (*BPMN*) или схема технологического процесса (*PFD*). Они тесно связаны с другими распространенными видами схем, такими как диаграммы *DFD* и диаграммы активности на унифицированном языке моделирования (*UML*).

Разработанная схема представлена на рисунке 14.

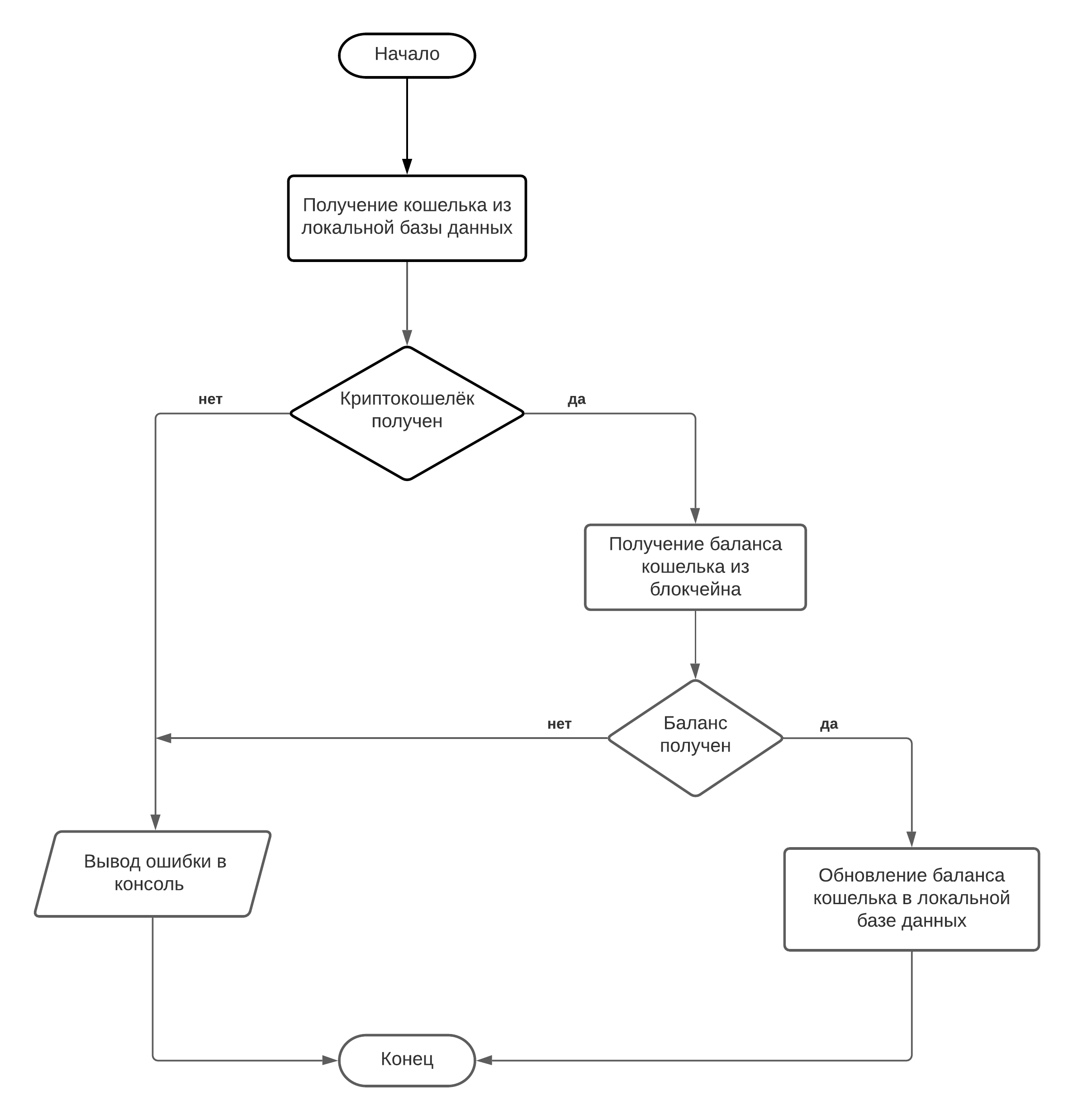


Рисунок 14 – Схема алгоритма обновления баланса

криптовалютного кошелька

1. РЕАЛИЗАЦИОННАЯ ЧАСТЬ
   1. Инструментальное обеспечение

Система учёта криптовалют представляет из себя мобильное приложение для операционных систем *iOS* и *Android.*

Для разработки приложения использованы следующие программные средства:

* язык программирования *Javascript*;
* фреймворк для разработки кроссплатформенных мобильных *React-Native*;
* криптографическая библиотека *crypto-js;*
* СУБД *Realm;*
* *Nomics API* для получения актуально стоимости криптовалют;
* библиотека *bitcore-lib* для доступа к блокчейну криптовалюты *Bitcoin*;
* среды разработки *Visual Studio Code* и *Xcode.*

3.1.1 Язык программирования *Javascript*. Изначально *JavaScript* был создан, чтобы «сделать веб-страницы живыми». Программы на этом языке называются скриптами. Они могут встраиваться в *HTML* и выполняться автоматически при загрузке веб-страницы.

Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

Когда *JavaScript* создавался, у него было другое имя – «*LiveScript*». Однако, язык *Java* был очень популярен в то время, и было решено, что позиционирование *JavaScript* как «младшего брата» *Java* будет полезно.

Со временем *JavaScript* стал полностью независимым языком со своей собственной спецификацией, называющейся *ECMAScript*, и сейчас не имеет никакого отношения к *Java*. [3]

Сегодня *JavaScript* может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся «движком» *JavaScript*.

У браузера есть собственный движок, который иногда называют «виртуальная машина *JavaScript*».

Разные движки имеют разные «кодовые имена». Например:

* *V8* – в *Chrome* и *Opera*;
* *SpiderMonkey* – в *Firefox*;
* «*Trident*» и «*Chakra*» для разных версий *IE*, «*ChakraCore*» для *Microsoft* *Edge*, «*Nitro*» и «*SquirrelFish*» для *Safari* и т.д.

Современный *JavaScript* – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.

Возможности *JavaScript* сильно зависят от окружения, в котором он работает. Например, *Node.JS* поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д. [4]

В браузере для *JavaScript* доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

* добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили;
* реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш;
* отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии *AJAX* и *COMET*);
* получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения;
* запоминать данные на стороне клиента. [5]

3.1.2 *React-Native* – это кроссплатформенный фреймворк с открытым исходным кодом для разработки нативных мобильных и настольных приложений на языках программирования *Javascirpt* и *Typescript*.

Основные принципы работы *React Native* практически идентичны принципам работы *React*, за исключением того, что *React-Native* управляет не браузерной *DOM*, а платформенными интерфейсными компонентами. *JavaScript*-код, написанный разработчиком, выполняется в фоновом потоке, и взаимодействует с платформенными *API* через асинхронную систему обмена данными, называемую *Bridge*. [6]

Хотя система стилей (способ конфигурации визуальных свойств элементов интерфейса) *React-Native* имеет синтаксис, похожий на *CSS*, фреймворк не использует технологии *HTML* или *CSS* как таковые. Вместо этого для каждой из поддерживаемых фреймворком операционных систем реализованы программные адаптеры, применяющие заданный разработчиком стиль к платформенному интерфейсному элементу.

*ReactNative* также позволяет разработчикам использовать уже существующий код, написанный на других языках программирования – например, *Java* или *Kotlin* для *Android* и *Objective-C* или *Swift* для *iOS*. Также *React Native* поддерживает интеграцию в уже существующие приложения – например, часть интерфейса мобильного приложения может быть реализована на *React Native*, а часть – при помощи чисто платформенных средств.

3.1.3 *Realm* – это нативная *NoSQL* база данных для *Android*, *iOS*, *Xamarin* и *React-Native*. Изначально спроектированная как настоящая объектно-ориентированная база данных, *Realm* отличается от других аналогичных библиотек, тем что рассматривает объекты данных как живые объекты – это значит, что объекты обновляются синхронно. Они мгновенно реагируют на изменения и легко сохраняются. Мобильная база данных *Realm* распространяется с открытым исходным кодом с 2016 года и бесплатна для разработчиков.

Первое время *Realm* был представлен только базами данных для разных платформ, сейчас появился сервер для синхронизации между всеми устройствами.

Благодаря использованию собственного движка, обеспечивающего высокую скорость работы и простоту в применении, мобильная база данных *Realm* выполняет запросы и синхронизирует объекты значительно быстрее, чем *Core Data* и *SQLite*, и осуществляет параллельный доступ к данным без проблем. Это значит, что несколько источников могут получить доступ к одному и тому же объекту без необходимости управлять блокировкой или каких-либо проблем с несогласованностью данных. Данная особенность является серьезным конкурентным преимуществом. Практика показывает, что в большинстве случаев *Realm* значительно превосходит в скорости не только *SQLite*, но и другие альтернативные *ORM* для *Android*, такие как *ORMLite* и *Greendao*.

Мобильная база данных *Realm* предлагает службы шифрования для защиты базы на диске с помощью *AES-*256 + *SHA-*2 64-разрядного шифрования. Это позволяет все данные, хранящиеся на диске зашифровывать и расшифровывать с помощью алгоритма *AES-*256 и проверять с помощью технологии *HMAC SHA-2*.

* 1. Программное обеспечение

3.2.1 В структуре проекта имеются следующие основные директории:

* *api.* В данной директории хранится программный код, отвечающий за взаимодействие с основными модулями системы;
* *components.* В данной директории хранятся компоненты, отвечающие за отображение информации на дисплее;
* *containers.* Данная директория предназначена для хранения компонентов-контейнеров для компонентов, отвечающих за отображение информации. Контейнеры предоставляют компонентам необходимую для отображения информацию, а также обрабатывают действия пользователя;
* *constants.* В данной директории хранятся константы, используемые в системе;
* *data*. Данная директория предназначена для хранения внутренней информации, например словаря для генерации секретной фразы;
* *database*. В данной директории хранятся схемы, описывающие структуру локальной базы данных;
* *assets.* Данная директория хранит подключаемые стили, шрифты и иконки, используемые в приложении.

Полная структура приложения приведена на рисунке 15.

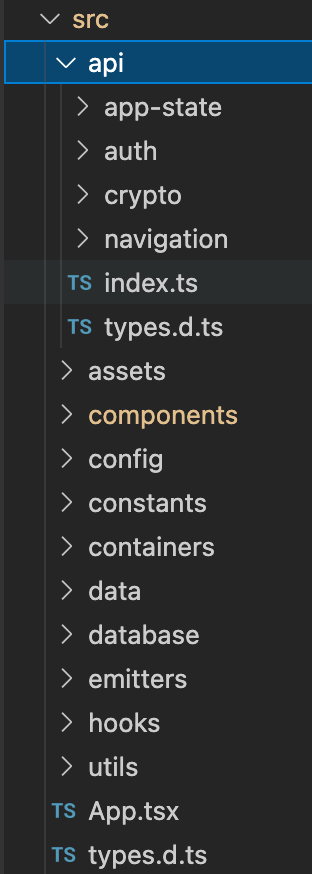


Рисунок 15 – Структура приложения

3.2.2 По функциональности система была разбита на следующие модули:

* *app-state.* Данный модуль отвечает за изменение глобального состояния приложения, такое как статус подключения к интернету, статус работы в фоновом режиме, язык системы и так делее;
* *auth.* Данный модуль отвечает за работу с авторизаций пользователя, выход из системы, а также сохранение секретной фразы и резервного файла;
* *crypto.* Данный модуль отвечает за работу с криптовалютами, создание криптокошельков, транзакций, получение баланса и так далее*;*
* *navigation.* Данный модуль отвечает за навигацию между экранами приложения.

Ниже приведён программный код реализации генерации секретной фразы из модуля *auth*:

*import dictionary from 'appData/dictionary.json';*

*import { SECRETPHRASE\_LENGTH } from 'appConstants';*

*import { Random } from '../../random';*

*const getRandomWord = async (): Promise<string> =>*

*dictionary[await Random.getRandomNumberUpTo(dictionary.length)];*

*const generatePassphrase = async (): Promise<string> => {*

*const phrase: Array<string> = [];*

*for (let i = 0; i < SECRETPHRASE\_LENGTH; i++) {*

*const word = await getRandomWord();*

*phrase.push(word.toUpperCase());*

*}*

*const finalPassphrase = phrase.join(' ');*

*console.log('---------- PASSPHRASE -----------');*

*console.log(finalPassphrase);*

*console.log('---------------------------------');*

*return finalPassphrase;*

*};*

* 1. Организационное обеспечение

Для доступа к системе пользователю необходимо установить на свой смартфон под управлением операционной системы *Android* или *iOS* мобильный клиент.

После запуска приложения пользователь попадёт на начальный экран с кнопками «*Sign in*» и «*Create Wallet*». Начальный экран представлен на рисунке 16.

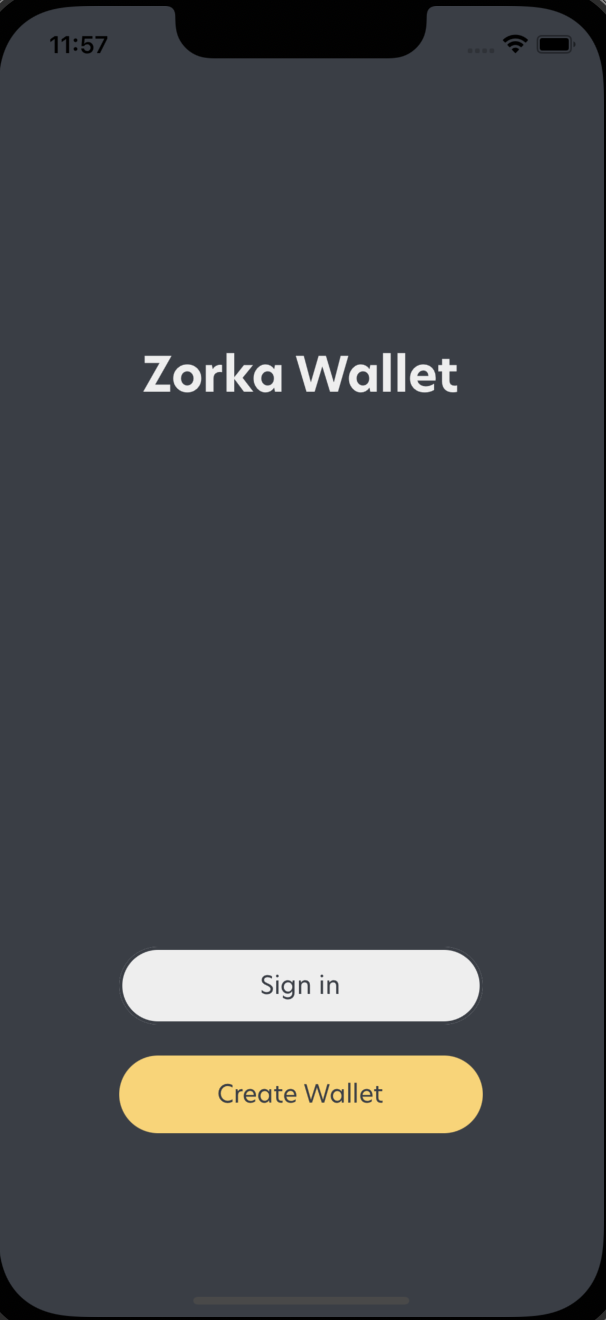


Рисунок 16 – Начальный экран

При нажатии на кнопку «*Create Wallet*» пользователь перейдёт на экран создания секретной фразы. Экран создания секретной фразы содержит кнопку «*Create a Secret Phrase*». Данный экран представлен на рисунке 17.

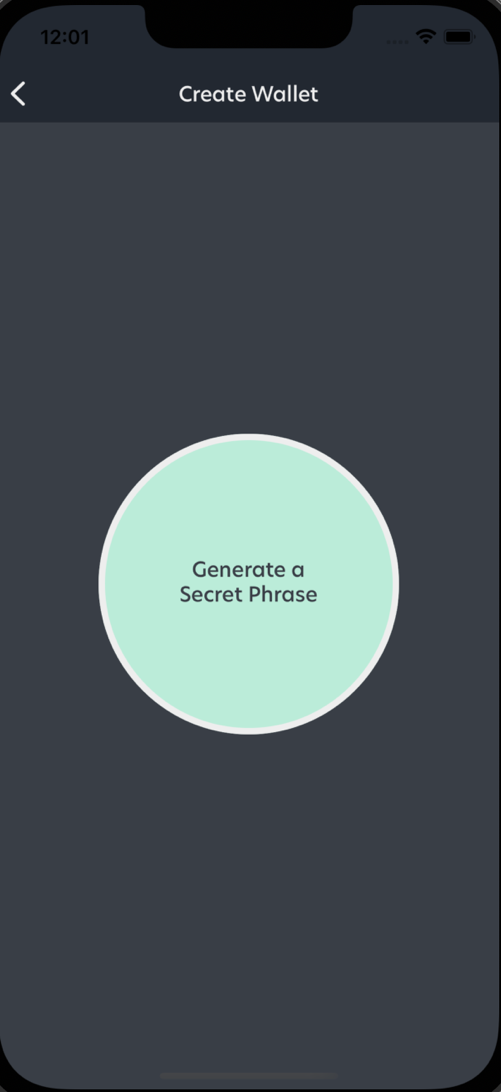


Рисунок 17 – Экран создания секретной фразы

При нажатии на кнопку «*Create a Secret Phrase*» система сгенерирует случайным образом секретную фразу, состоящую из 24 слов, после чего пользователь перейдет на экран секретной фразы. Экран секретной фразы содержит секретную фразу, кнопку «*Save*», по нажатия на которую пользователю будет предложено сохранить секретную фразу, а также кнопку «*Done*», по нажатия на которую пользователь перейдёт к списку поддерживаемых криптовалют.

Данный экран, процесс сохранения секретной фразы, а также экран списка поддерживаемых криптовалют представлены на рисунке 18, рисунке 19 и рисунке 20 соответственно.



Рисунок 18 – Экран секретной фразы

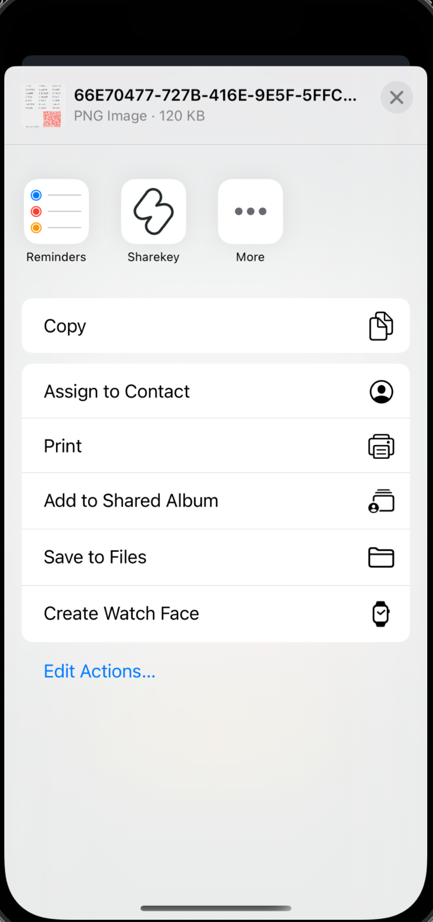


Рисунок 19 – Процесс сохранения секретной фразы

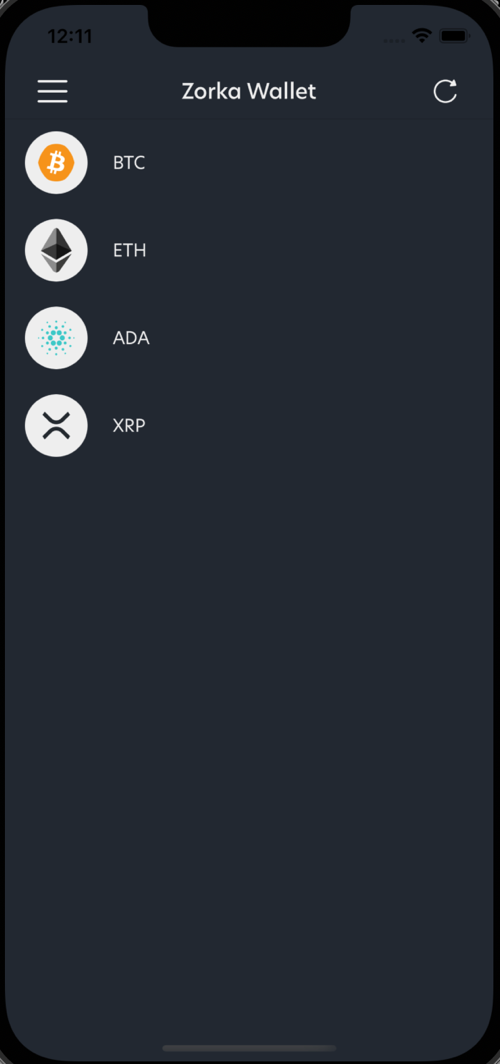


Рисунок 20 – Список поддерживаемых криптовалют

При нажатии на кнопку «меню» в левом верхнем углу экрана откроется боковое меню, где пользователю будут доступны опции перехода в экран просмотра секретная фразы и выхода из системы. На экране просмотра секретной фразы имеются кнопки «*Save Backup*», нажатие на которую откроет меню сохранения резервного файла и кнопка «*Save Secret Phrase*», нажатие на которую откроет меню сохранения секретной фразы.

Боковое меню представлено на рисунке 21, экран просмотра секретной фразы представлен на рисунке 22. Меню сохранения резервного файла и меню сохранения секретной фразы представлены на рисунке 23 и рисунке 24 соответственно.

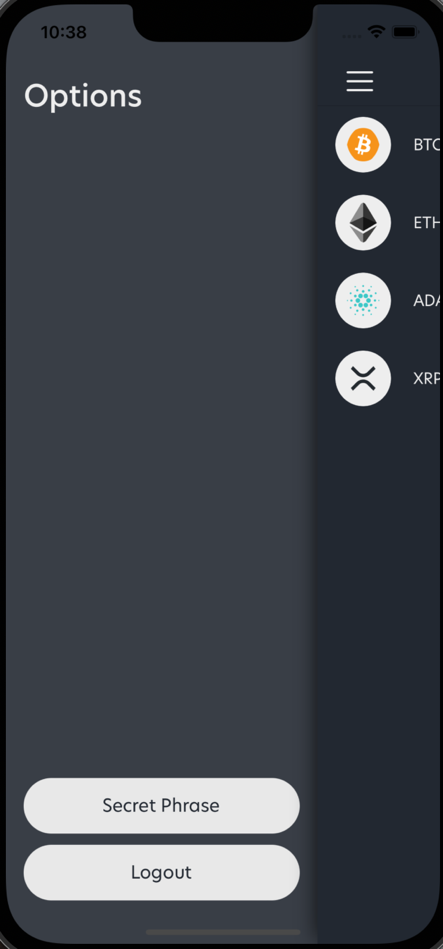


Рисунок 21 – Боковое меню

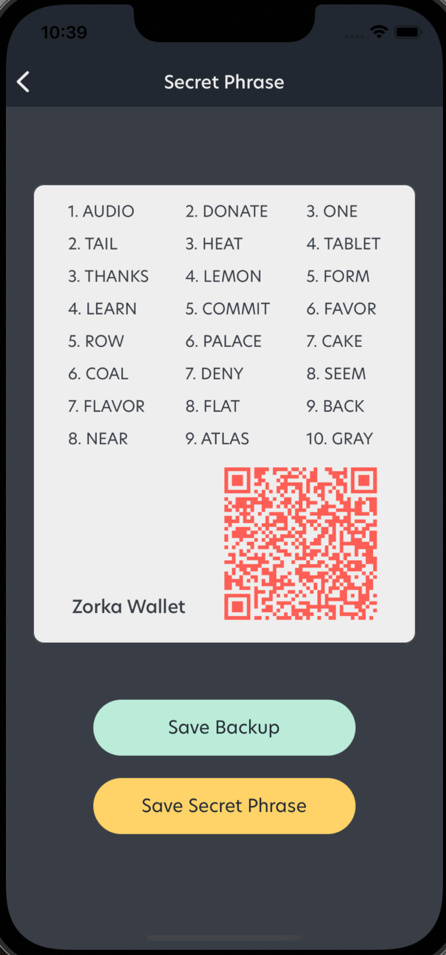


Рисунок 22 – Экран просмотра секретной фразы

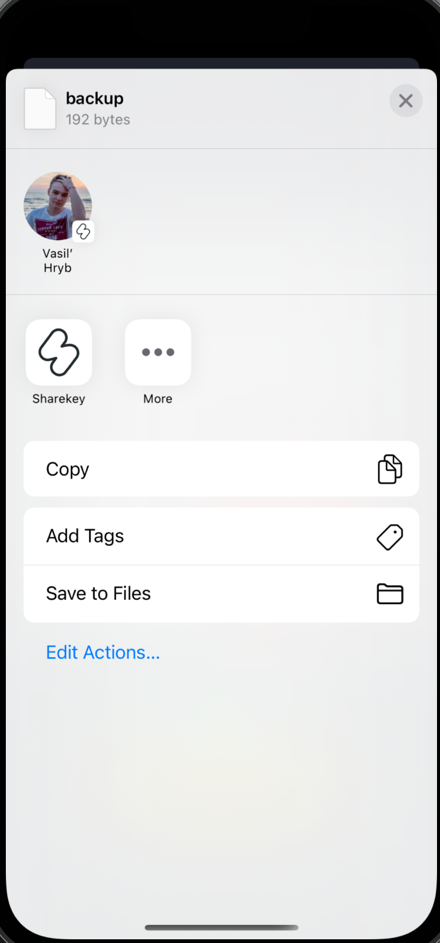


Рисунок 23 – Меню сохранения резервного файла

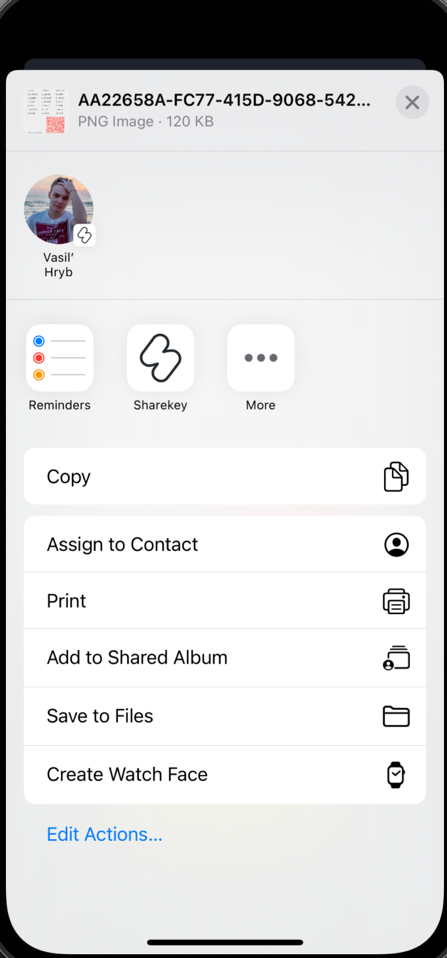


Рисунок 24 – Меню сохранения секретной фразы

При нажатии на криптовалюту на главном экране, откроется экран кошелька выбранной криптовалюты. На данном экране отображается баланс и публичный адрес криптокошелька, а также кнопки «*View History*», «*Send Crypto*» и «*Copy Public Address*». При нажатии на кнопку «*View History*», пользователю отображается история всех транзакций данного криптокошелька, при нажатии на кнопку «*Send Crypto*» пользователю отображается меню создания транзакции, а при нажатии на кнопку «*Copy Public Address*» публичный адрес кошелка копируется в буфер обмена устройства.

Экран криптокошелька представлен на рисунке 25.

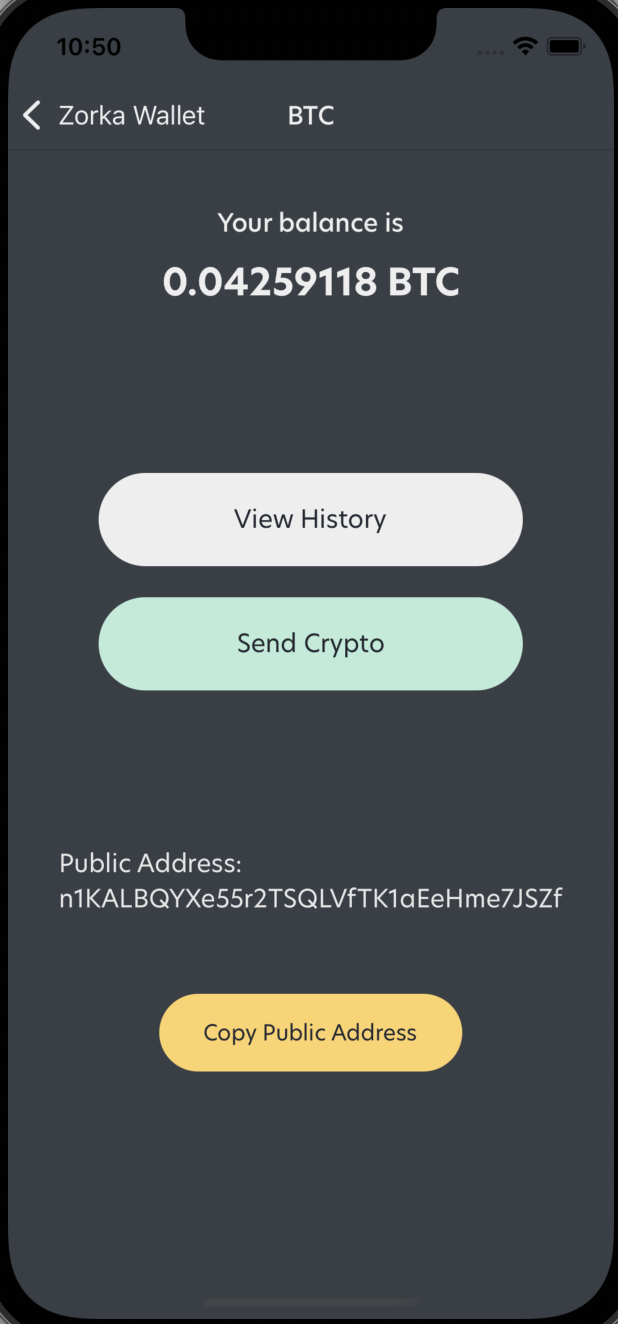


Рисунок 25 – Экран криптокошелька

В истории транзакций для каждой транзакции отображается тип транзакций зелёным цветом для входящих транзакций и красным для исходящих, размер транзакции, а также дата транзакции. История транзакций представлена на рисунке 26.

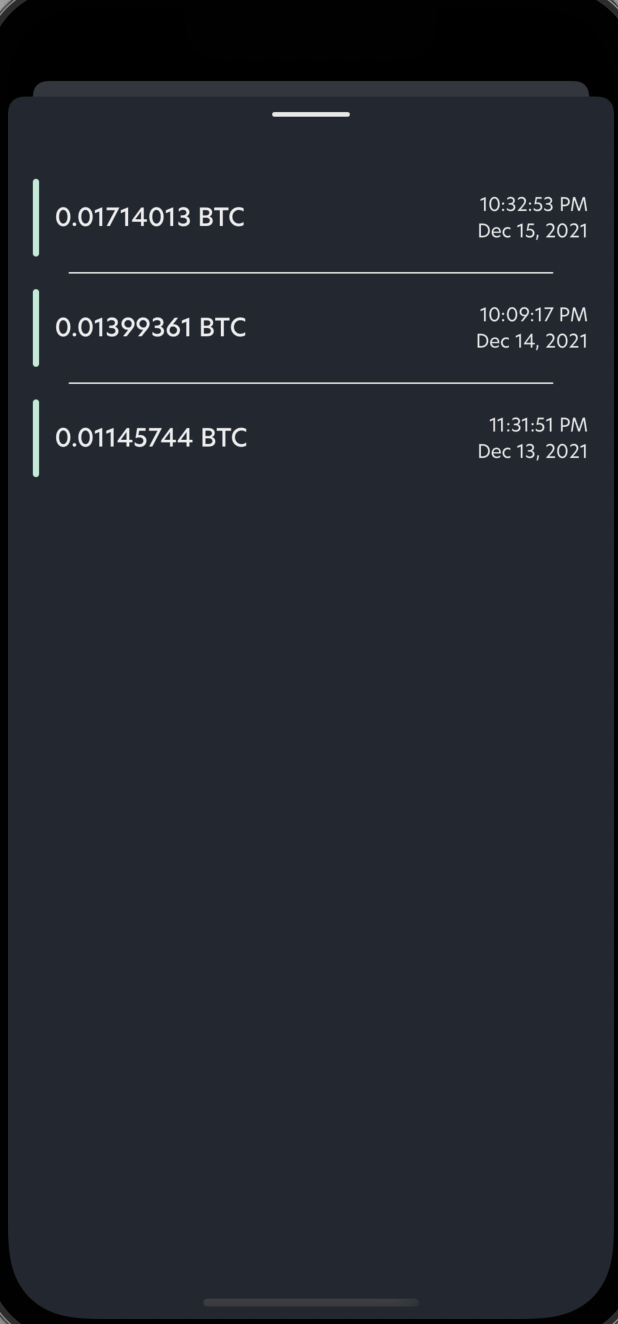


Рисунок 26 – История транзакций

Заключение

В курсовой работе использованы методы проектирования автоматизированных систем. Показан метод графического представления автоматизированной системы с помощью диаграмм.

Для «системы учёта криптовалют» разработаны диаграммы *IDEF0*, *IDEF3*, *DFD* и *UML*-диаграммы. Изучены понятия класс, атрибут, операция, связь, сущность и т.д.

Определен набор технологий для реализации программного обеспечения системы. Спроектирован и разработан пользовательский интерфейс, который позволяет максимально просто и удобно пользоваться всеми функциями системы.

Система позволит упростить управление пользователем криптовалютами, а также обеспечит безопасный и удобный способ хранения приватных ключей различных криптовалют.

Таким образом система удовлетворяет всем техническим и функциональным требованиям, имеет простой и понятный интерфейс, позволяющий снизить затраты и время на освоение программного средства.

В дальнейшем систему можно улучшить добавлением поддержки новых криптовалют, созданием серверной части для хранения приватных ключей пользователей в зашифрованном виде, что позволит исключить из процесса авторизации необходимость резервного файла, а также добавлением функциональности, связанной с авторизацией пользователя с использованием сканера отпечатка пальца.

Функциональность системы можно расширить добавлением курса криптовалют и подсчета общего баланса криптокошелька в различных мировых фиатных валютах.

Кроме этого, круг потенциальных пользователей системы может быть расширен добавлением поддержки новых языков.

В ходе проведения курсовой работы были углублены, закреплены и конкретизированы теоретические знания в области автоматизации процессов, приобретены навыки по проектированию.

Список использованных источников

[1] Гайдамакин, Н. А. Автоматизированные информационные системы, базы и банки данных. Вводный курс: Учебное пособие / Н. А. Гайдамакин. – Москва. : Гелиос АРВ, 2002. – 68 с.

[2] Бабич, А. В. *UML*: Первое знакомство / А.В. Бабич. – Москва.: БИНОМ. Лаборатория знаний, 2008. – 342 с.

[3]  Флэнаган, Дэвид *JavaScript*. Подробное руководство / Дэвид Флэнаган. – М.: Питер, 2021. – 720 c.

[4]  Крокфорд, Дуглас Как устроен *JavaScript* / Дуглас Крокфорд –   
СПб. : Питер, 2019. – 304 с.

[5] Николас, З. *JavaScript*. Оптимизация производительности / З. Николас. – Москва. : Символ-плюс, 2016. – 482 c.

[6] React Native [Электронный ресурс]. – Режим доступа: https://reactnative.dev/docs/getting-started.

**Ведомость курсового проекта**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–53 01 02 01 08 ПЗ | | | | Пояснительная записка | | | | 44 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР.000.001.ПД | | | | Диаграмма вариантов использования | | | | Формат А1 | | | |
|  | | | | б | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-53 01 02 01 08 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | Л. | № докум. | Подп. | Дата | Система учёта криптовалют |  | | | | Лист | Листов |
| Разраб. | | Гриб |  |  |  | Т | |  | 44 | 44 |
| Пров. | | Ломако |  |  | Кафедра ИТАС  гр. 820602 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |