Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет  информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем Дисциплина «Технологии проектирования сложных информационных  систем»

«К ЗАЩИТЕ ДОПУСТИТЬ»

Руководитель курсового проекта,

ассистент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.Н. Тонкович

\_\_\_.\_\_\_\_.2023

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА КОНВЕРТАЦИИ И**

**АНАЛИЗА ТРЕНДА КРИПТОВАЛЮТ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ»**

БГУИР КП 1-40 05 01-10 005 ПЗ

Выполнил студент группы 114302

Верховодко Никита Вячеславович

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

(подпись студента)

Курсовой проект представлен на

проверку \_\_\_.\_\_\_\_.2023

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

(подпись студента)

Минск 2023

**РЕФЕРАТ**

БГУИР КП 1-40 05 01 005 ПЗ

**Верховодко, Н.В.** пояснительная записка к курсовому проекту / Н.В. Верховодко.  – Минск : БГУИР, 2023. – 777 с., чертежей (плакатов) – 4 л. формат А3.

Пояснительная записка 777 страниц, 777 рисунков, 20 источников, 3  приложения

**ПРОГРАММНОГО СРЕДСТВА КОНВЕРТАЦИИ И АНАЛИЗА ТРЕНДА КРИПТОВАЛЮТ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ**

*Цель проектирования:* разработка программного средства для конвертации и анализа криптовалюты.

*Методология проведения работы*: в процессе решения поставленных задач ипользованы методики проектирования графического пользовательского итерфейса и алгоритмы, описывающие логику работы приложения в рамках клиент-серверной архитектуры.

*Результаты работы:* выполнен анализ предметной области и существющих аналогов разрабатываемого программного средства; построена функциональная модель предметной области с использованием графической нотации BPMN; выполнен анализ пользовательских требований к разрабатываемому программному средству; составлена спецификация функциональных требований к программному средству; представлены UML диаграммы, отражающие поведение и аспекты взаимодействия программного средства; представлены макеты пользовательского интерфейса; спроектирована архитектура программного средства при помощи стандарта UML; разработано и протестировано программное средство; составлено руководство по развертыванию и использованию программного средства.

*Область применения результатов:* могут быть применены как для личного, так и корпоративного использования.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc134222423)

[1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИССЛЕДОВАНИЙ 6](#_Toc134222424)

[И ПРОГРАММНЫХ РЕШЕНИЙ 6](#_Toc134222425)

[1.1 Описание и анализ предметной области 6](#_Toc134222426)

[1.2 Сравнительный обзор аналогов программного средства 8](#_Toc134222427)

[2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 11](#_Toc134222428)

[И РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ 11](#_Toc134222429)

[СРЕДСТВУ 11](#_Toc134222430)

[2.1 Анализ и формализация бизнес-процессов предметной области 11](#_Toc134222431)

[2.2 Анализ требований к разрабатываемому программному 14](#_Toc134222432)

[средству. Спецификация функциональных 14](#_Toc134222433)

[и нефункциональных требований 14](#_Toc134222434)

[3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 27](#_Toc134222435)

[3.1 Проектирование пользовательского интерфейса 27](#_Toc134222436)

[3.2 Обоснование выбора компонентов и технологий для 27](#_Toc134222437)

[реализации программного средства 27](#_Toc134222438)

[3.3 Проектирование схемы информационной модели предметной 28](#_Toc134222439)

[области 28](#_Toc134222440)

[3.4 Архитектурные решения 36](#_Toc134222441)

[3.5 Описание динамических аспектов поведения объектов системы 37](#_Toc134222442)

[3.6 Описание и разработка алгоритмов, реализующих бизнес-логику разрабатываемого программного средства 40](#_Toc134222443)

[ЗАКЛЮЧЕНИЕ 43](#_Toc134222444)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 44](#_Toc134222445)

# **ВВЕДЕНИЕ**

С момента появления первой криптовалюты биткоина в 2009 году, рынок криптовалют значительно вырос и развился. Сегодня криптовалюты являются одним из наиболее важных активов в мире цифровых финансов. Стабильность и рост криптовалютных рынков привлекают внимание многих инвесторов, торговцев и аналитиков.

Для успешной работы на криптовалютных рынках необходимо иметь доступ к актуальной информации и быть в курсе изменений на рынке в режиме реального времени. Именно для этой цели было разработано программное средство для конвертации и анализа тренда криптовалют в режиме реального времени.

В данной курсовой работе будет рассмотрено программное средство для конвертации и анализа тренда криптовалют в режиме реального времени. В работе будет проведен обзор литературы, анализ возможностей и функциональности программного средства, а также приведены результаты его тестирования. Криптовалюты являются децентрализованными цифровыми валютами, которые используют криптографию для защиты своей безопасности и контроля за созданием новых единиц. С каждым годом рынок криптовалют становится все более значимым и перспективным. В настоящее время многие инвесторы, трейдеры и аналитики пользуются криптовалютами как альтернативным инструментом для инвестирования и заработка денег.

Однако работа на криптовалютных рынках требует быстрого реагирования на изменения и актуальной информации о текущих трендах. В связи с этим, программное средство для конвертации и анализа тренда криптовалют в режиме реального времени является незаменимым инструментом для работы на криптовалютных рынках.

Цель данной курсовой работы состоит в исследовании программного средства для конвертации и анализа тренда криптовалют в режиме реального времени. Для достижения этой цели были поставлены следующие задачи:

* провести обзор литературы по криптовалютам и их рынку;
* рассмотреть возможности и функциональность программного средства для конвертации и анализа тренда криптовалют в режиме реального времени;
* протестировать программное средство на практике и оценить его эффективность.

В данной работе будет использоваться методология, основанная на анализе литературных источников, экспериментальных данных и математическом моделировании. Результаты работы могут быть полезными для трейдеров, инвесторов и аналитиков, которые работают с криптовалютами и заинтересованы в получении актуальной информации и анализе трендов.

Курсовой проект состоит из введения, пяти разделов с краткими выводами по каждому разделу, заключения, списка использованных источников, приложений и ведомости.

В первом разделе приведен обзор современного состояния проблемы управления криптовалютными сбережениями *IT*-компании, проанализированы аналоги программного средства.

Во втором разделе представлены анализ и моделирование предметной области с формализацией ее основных бизнес-процессов. На основании выполненного анализа предметной области, методов и средств, применяемых для существующих аналогов разработаны предложения по совершенствованию рассмотренных бизнес-процессов. Приведена спецификация функциональных и нефункциональных требований к программному средству. В третьем разделе предложены архитектурные решения, выполнено проектирование пользовательского интерфейса, разработаны алгоритмические и программные реализации модулей программного средства. В четвертом разделе представлены доказательства работоспособности программного средства на основе тестовых примеров.

В пятом разделе разработано руководство по развертыванию и использованию программного средства.

В приложениях представлены листинги кода алгоритмов, реализующих бизнес-логику, и скрипта генерации базы данных, диаграмма классов, схема алгоритма управления онлайн-курсом корпоративного обучения сотрудников. Курсовой проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности составляет 99,9%. Цитирования обозначены ссылками на публикации, указанными в «Списке использованных источников».

Скриншот приведен в приложении А на рисунке А.1.

# **1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИССЛЕДОВАНИЙ**

# **И ПРОГРАММНЫХ РЕШЕНИЙ**

## **Описание и анализ предметной области**

Криптовалюты стали объектом интереса для многих людей по всему миру в последние годы. Это электронные деньги, которые используются для обмена в интернете без привязки к банковской системе или правительству. Они основаны на технологии блокчейн, которая позволяет создавать децентрализованные и защищенные базы данных.

Анализ тренда криптовалют является важной задачей для инвесторов, трейдеров, аналитиков и всех, кто интересуется рынком криптовалют. Существует множество инструментов, которые помогают в анализе рынка криптовалют, однако не все из них обладают функциональностью конвертации и анализа тренда в режиме реального времени.

Литературный обзор показывает, что в настоящее время существует много работ, посвященных анализу рынка криптовалют. В частности, авторы работы (*Zhang*, *Xu*, & *Xu*, 2021) рассматривают методы и инструменты анализа данных, которые используются в сфере криптовалют и показывают применение аналитических инструментов для прогнозирования цен на криптовалюты. В другой работе (*Mukherjee* & *Bose*, 2019) авторы рассматривают различные методы анализа временных рядов, которые могут быть использованы для анализа рынка криптовалют.

Существует также множество программных решений, которые помогают в анализе рынка криптовалют. Некоторые из них, например, *CoinTracking*, *CryptoCompare*, *Coinigy*, *Bitsgap* и *TradingView*, обладают функциональностью конвертации и анализа тренда в режиме реального времени. Однако, многие из них имеют ограничения по функциональности и ограничения по использованию.

Некоторые из программных решений имеют ограничения по поддерживаемым криптовалютам и биржам. Например, *CoinTracking* поддерживает только 77 бирж, в то время как *CryptoCompare* поддерживает более 100 бирж, но имеет ограничения по количеству данных, которые можно получить в бесплатной версии. *TradingView* позволяет использовать множество индикаторов и стратегий, но не имеет функциональности конвертации валют.

Из анализа литературных исследований и программных решений следует, что существует потребность в программном средстве, которое будет обладать функциональностью конвертации и анализа тренда криптовалют в режиме реального времени, поддерживать большое количество криптовалют и бирж, а также обладать возможностью использования различных индикаторов и стратегий для анализа рынка криптовалют.

Таким образом, разработка программного средства, способного анализировать тренд криптовалют в режиме реального времени, может быть важным инструментом для инвесторов, трейдеров и аналитиков, работающих на рынке криптовалют. Однако, для создания такого программного средства необходимо провести дополнительное исследование и разработку, которая может быть сложной задачей, учитывая высокую сложность и динамичность рынка криптовалют.

Программное средство конвертации, покупки, продажи и анализа тренда криптовалют в режиме реального времени является инструментом для работы с криптовалютами. Оно позволяет пользователям конвертировать одну криптовалюту в другую, покупать и продавать криптовалюты на различных биржах, а также анализировать тренды цен на криптовалюты в режиме реального времени.

Существует множество программных средств для работы с криптовалютами, но многие из них имеют недостатки. Некоторые из них имеют ограниченный функционал и не могут обрабатывать большие объемы данных, другие могут иметь проблемы с безопасностью или быть неудобными в использовании.

Одним из основных преимуществ программного средства конвертации, покупки, продажи и анализа тренда криптовалют в режиме реального времени является его способность работать с большими объемами данных в режиме реального времени. Это позволяет пользователям получать актуальную информацию о ценах на криптовалюты и быстро реагировать на изменения на рынке.

Кроме того, программное средство конвертации, покупки, продажи и анализа тренда криптовалют в режиме реального времени может иметь функции анализа данных, такие как графики и диаграммы, которые помогают пользователям понимать тренды цен на криптовалюты и принимать более обоснованные решения.

Однако, использование программного средства конвертации, покупки, продажи и анализа тренда криптовалют в режиме реального времени также имеет свои недостатки. Один из них – это высокий уровень риска при работ с криптовалютами, который связан с изменчивостью цен на рынке и возможностью кибератак и кражи криптовалюты.

Кроме того, не все страны законодательно регулируют криптовалюты, что может создавать юридические проблемы для пользователей и разработчиков программного средства.

Для решения этих проблем могут использоваться различные методы, такие как внедрение системы безопасности и шифрования, юридическое регулирование использования криптовалют, а также разработка функций для управления рисками при работе с криптовалютами.

Таким образом, программное средство конвертации, покупки, продажи и анализа тренда криптовалют в режиме реального времени имеет большой потенциал для работы с криптовалютами, но также сопряжено с рядом рисков и проблем, которые требуют решения. Разработчики программного средства должны учитывать все эти факторы и создавать безопасные, удобные и функциональные инструменты для работы с криптовалютами.

## **1.2 Сравнительный обзор аналогов программного средства**

Существует множество программных продуктов на рынке, которые предоставляют функционал конвертации и анализа тренда криптовалют в режиме реального времени.

Таблица 1 – Аналоговые средства

|  |  |  |  |
| --- | --- | --- | --- |
| Название аналога | Компания-разработчик | *URL* официального  сайта компании-  разработчика | *URL* источников  для обзора |
| Binance | Binance | https://www.binance.com/en | https://ttrcoin.com/markets/binance/ |
| *Crypto.com* | *Monaco Technologies GmbH* | https://crypto.com/ | https://cryptonisation.com/kriptovalyutnaya-birzha-crypto-com-egistratsiya-torgovlya-otzyvy/ |

*Crypto.com* – это приложение для покупки, продажи и хранения криптовалют. Оно предоставляет пользователю простой и интуитивно понятный интерфейс, а также широкий функционал, включая конвертацию криптовалют и аналитические инструменты. *Crypto.com* также предоставляет собственную криптовалютную карту, которая позволяет пользователям использовать свои криптовалютные средства для покупок в реальном мире. Однако, есть ограничения по доступности сервиса для жителей определенных стран.

*Binance* – это платформа для торговли криптовалютами, которая предоставляет широкий функционал, включая конвертацию криптовалют и аналитические инструменты. *Binance* также имеет свою криптовалютную биржу, на которой пользователи могут торговать различными криптовалютами. Однако, некоторые пользователи могут столкнуться с ограничениями доступности в некоторых странах.

*Binance* является одной из крупнейших и самых популярных криптобирж в мире. Она была создана в 2017 году и имеет штаб-квартиру в Шанхае, Китай. *Binance* предлагает широкий выбор криптовалют, включая *Bitcoin*, *Ethereum*, *Litecoin* и другие. Биржа также предоставляет множество инструментов для трейдинга, таких как маржинальная торговля и фьючерсы.

Рассмотрим некоторые критерии, на которых можно оценить *Binance* как платформу для торговли криптовалютами:

1. Комиссии: *Binance* имеет одни из самых низких комиссий на рынке криптобирж. Комиссия за транзакцию составляет 0,1%, но при использовании токена *Binance* (*BNB*) эта комиссия может быть снижена до 0,075%.
2. Ликвидность: Биржа имеет высокую ликвидность на большинстве пар торгов. Это означает, что вы можете быстро и легко купить или продать криптовалюту на *Binance*, что делает ее привлекательной для торговли.
3. Безопасность: *Binance* сильно инвестировала в свою безопасность, используя различные меры, такие как двухфакторная аутентификация, системы оповещения о подозрительной активности и защита от *DDoS*-атак. Кроме того, *Binance* хранит большинство своих средств в холодных кошельках, что значительно снижает риск взлома.
4. Репутация: Она имеет хорошую репутацию на рынке криптобирж и находится на вершине списка самых популярных и надежных криптобирж. Она также активно работает над улучшением своей платформы и предоставлении лучшего опыта для своих пользователей.
5. Диапазон торгуемых активов: *Binance* предлагает широкий диапазон торгуемых криптовалют, включая наиболее популярные, такие как *Bitcoin*, *Ethereum*, *Litecoin* и другие. Кроме того, биржа также предлагает торговлю фьючерсами и опционами.

Плюсы:

1. Большой выбор криптовалют: *Binance* поддерживает более 100 криптовалют, что делает ее одной из крупнейших и наиболее разнообразных платформ для торговли криптовалютой.
2. Низкие комиссии: *Binance* имеет одни из самых низких комиссий на рынке, особенно при использовании их собственной криптовалюты *Binance* *Coin* (*BNB*).
3. Высокий уровень безопасности: *Binance* имеет многофакторную аутентификацию, криптографическое хранение средств и систему защиты от взлома. Безопасность является одним из приоритетов *Binance*.
4. *Binance Launchpad: Binance Launchpad* позволяет пользователям участвовать в публичных продажах токенов на начальном этапе развития проекта. Это может предоставить пользователям доступ к перспективным криптовалютным проектам.

Минусы:

1. Ограничения регистрации: *Binance* может быть недоступен для пользователей из некоторых стран, и процесс регистрации может быть более сложным, чем на других платформах.
2. Недостаток регулирования: как и большинство платформ для торговли криптовалютой, *Binance* не регулируется во всех странах, что может создавать некоторые риски для пользователей.
3. Недостаток поддержки: некоторые пользователи жаловались на недостаток поддержки со стороны *Binance*, что может привести к задержкам в разрешении проблем и вопросов пользователей.

В целом, *Binance* является одной из наиболее популярных и широко используемых платформ для торговли криптовалютой. Однако, как и любая платформа, у нее есть свои преимущества и недостатки, которые следует учитывать при выборе платформы для торгов

Все перечисленные программные продукты предоставляют широкий функционал и удобный интерфейс для работы с криптовалютами, включая конвертацию и аналитические инструменты. Однако, каждый из них имеет свои ограничения и особенности, и выбор конкретного продукта зависит от индивидуальных потребностей и предпочтений пользователя.

# **2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

# **И РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ**

# **СРЕДСТВУ**

## **2.1 Анализ и формализация бизнес-процессов предметной области**

Необходимо создать такое программное изделие, которое позволило бы получить доступ к данным о прохождении слушателями обучающего курса. В процессе проектирования программного средства сначала строится функциональная модель существующей организации работ, называемая моделью *AS-IS* в нотации *BPMN*. *BPMN* – это стандартная методология для отображения бизнес-процессов с использованием метода блок-схемы.

Анализ модели *AS-IS* позволяет выявить наиболее слабы места и наметить пути улучшения ситуации за счет структурных перестроек или организации бизнес-процессов. Таким образом, на рисунке 2.1 представлена модель *AS-IS* деятельности конвертации и анализа тренда криптовалют.

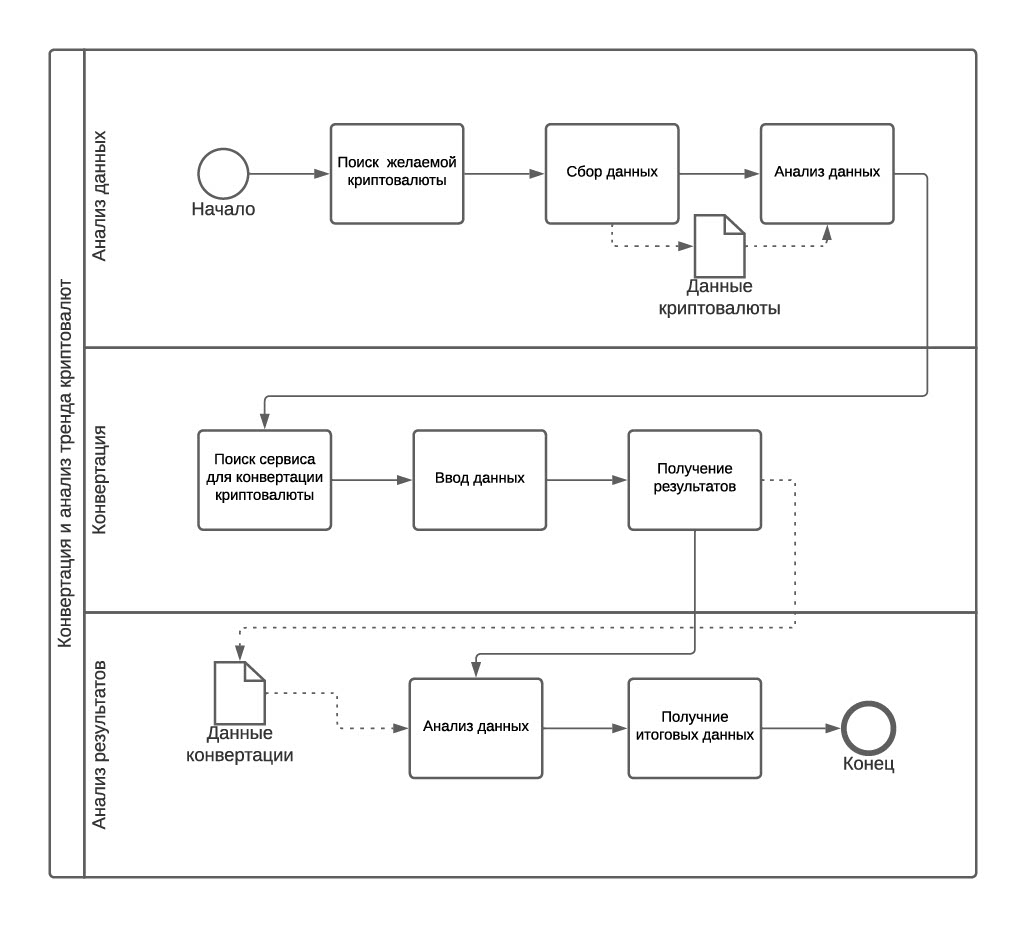


Рисунок 2.1 – Модель *AS-IS* деятельности управления обучающим

курсом в нотации *BPMN*

Как видно на модели *AS-IS*, процесс конвертации криптовалют занимает очень много действий и времени.

Найденные в модели *AS-IS* недостатки можно исправить в модели *TO-BE*. Именно на основе модели *TO-BE* производиться реорганизация бизнес- процессов и проектирование программного средства. Модель *ТО-ВЕ* нужна для оценки последствий внедрения информационной системы и анализа альтернативных путей выполнения работы и документирования того, как программное средство будет функционировать в будущем. Модель *TO-BE* изображена на рисунке 2.2.

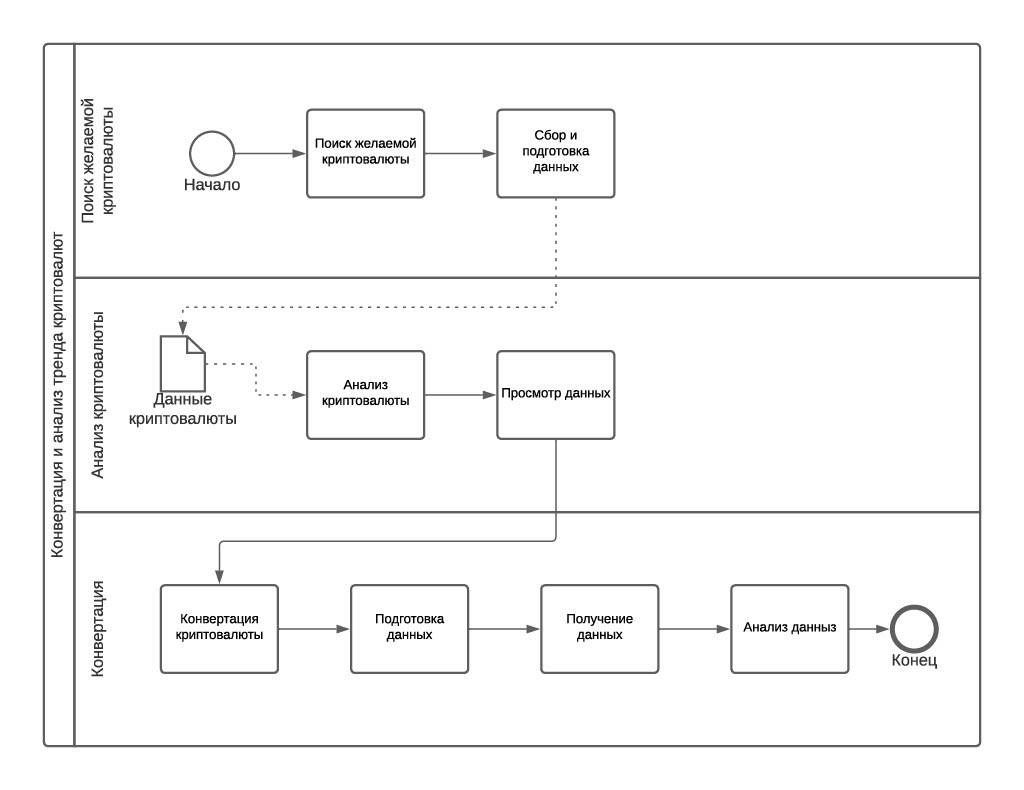


Рисунок 2.2 – Модель *TO-BE* деятельности управления обучающим

курсом в нотации *BPMN*

Таким образом, был проведен анализ бизнес-процессов предметной области, а также составлены модели *AS-IS* и *TO-BE* в нотации *BPMN*. Следует отметить, что модель *TO-BE* отображает те полезные функции, которые позволят успешно внедрить и использовать данное программное обеспечение для конвертации и анализа тренда криптовалют.

Контекстная диаграмма прежде всего позволяет быстро, кратко и ёмко описать назначение и границы криптокошелька, выявить и устранить коллективные расхождения в их понимании, показать и договориться о её масштабе. Функции криптокошелька разделяются в зависимости от роли пользователя. Со стороны пользователя, криптокошелек должен позволять создавать и управлять мультивалютными кошельками, проводить транзакции с различными криптовалютами, хранить и защищать личные данные и финансовые средства, включая возможность двухфакторной аутентификации и шифрования. Также иметь возможность отслеживать историю транзакций, получать уведомления об изменении курса криптовалют и других важных событий.

Со стороны разработчика, криптокошелек должен иметь функциональность для блокировки пользователей, нарушающих правила пользования, а так же просмотр истории действий пользователей.

Кроме того, криптокошелек должен иметь удобный интерфейс для управления и проведения транзакций. Криптокошелек должен быть также совместим с различными операционными системами и браузерами.

В целом, криптокошелек должен обеспечивать безопасность и удобство при хранении и проведении транзакций с криптовалютами, а также быть гибким и легко настраиваемым для удовлетворения потребностей различных пользователей.

Контекстная диаграмма для выпуска программного средства представлена на рисунке 2.3.

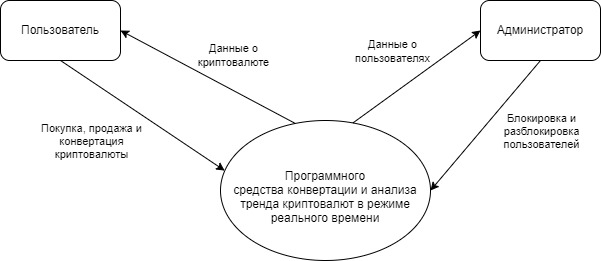


Рисунок 2.3 – Контекстная диаграмма для выпуска программного

средства

В данном разделе описывается процесс проектирования программного средства для получения доступа к данным о прохождении обучающего курса. Для этого была построена функциональная модель существующей организации работ в нотации *BPMN*, называемая моделью *AS-IS*. Анализ этой модели помог выявить слабые места и наметить пути улучшения ситуации. Затем была составлена модель *TO-BE*, на основе которой производится реорганизация бизнес-процессов и проектирование программного средства.

Кроме того, в разделе описывается контекстная диаграмма для криптокошелька, которая позволяет кратко и ёмко описать его функции и границы, выявить расхождения в понимании и договориться о масштабе. Описываются функции криптокошелька как со стороны пользователя, так и со стороны разработчика, а также его основные требования и интерфейс.

## **2.2 Анализ требований к разрабатываемому программному**

## **средству. Спецификация функциональных**

## **и нефункциональных требований**

Разрабатываемое программное средство должно обеспечивать процесс анализа тренда и конвертации криптовалют. В данный процесс входит создание программного средства для конвертации и анализа тренда криптовалют в режиме реального времени. Это включает в себя конвертацию и анализ тренда на криптовалютном рынке. Данное программное средство моделируется на основе уже существующих аналогов.

Программные средства такого типа появились ввиду необходимости повышения уровня качества анализа тренда криптовалют и удобства работы с криптовалютными данными. Необходимо рассмотреть исходные данные, которые суммируют обоснование и содержание нового продукта. Таким образом, проанализировав предметную область, а также исследовав аналоги разрабатываемого продукта, необходимо заметить, что все существующие программные средства имеют ряд таких недостатков, как отсутствие возможности анализировать данные в режиме реального времени и предоставлять подробную информацию о криптовалютах.

В программе будет два типа пользователей: пользователь, желающий пользоваться криптовалютой и модератор, который имеет возможность управлять данными пользователей. У разных пользователей будут разные функциональные возможности, они могут пересекаться, однако некоторые из них доступны только модераторам, для корректной работы программного средства. Данные функции являются также в будущем возможными *use cases*.

В программе будет 2 типа пользователей: пользователь и администратор.

Основными функциональными требованиями к программному средству являются:

* конвертация, покупка, продажа и обмен криптовалют в режиме реального времени;
* отображение актуальных цен на рынке криптовалют;
* предоставление статистических данных о ценах, объемах торгов и других показателях для каждой криптовалюты;
* реализация функционала управления балансом пользователей, возможность внесения и вывода средств;
* защита пользовательских данных и средств с помощью высоких стандартов безопасности.

Анализ требований к разрабатываемому программному средству является ключевым этапом в разработке любого программного продукта. Он позволяет установить основные цели и задачи, которые должно решать разрабатываемое ПО, а также определить его функциональные и нефункциональные требования.

Спецификация функциональных требований описывает, какие функции должно выполнять программное средство и как они должны быть реализованы. Кроме того, она определяет, какие данные должны обрабатываться и каким образом должны быть представлены пользователю. Спецификация функциональных требований помогает определить объем работы и ожидания пользователей, что в свою очередь влияет на планирование разработки и создание качественного продукта.

Нефункциональные требования определяют характеристики программного средства, которые не связаны с его функциональностью, но которые важны для успешной работы продукта. Это, например, производительность, надежность, безопасность и удобство использования. Спецификация нефункциональных требований помогает разработчикам оценить реализуемость проекта, выбрать соответствующие технологии и определить риски разработки.

Таким образом, анализ требований к разрабатываемому программному средству и спецификация функциональных и нефункциональных требований являются важнейшими компонентами процесса разработки ПО. Они позволяют определить основные характеристики продукта, обеспечить его соответствие потребностям пользователей и обеспечить качественный результат в конечном итоге.

Диаграмма вариантов использования (*Use Case Diagram*) является одним из наиболее популярных инструментов в разработке программного обеспечения для моделирования функциональных требований и поведения системы из точки зрения конечных пользователей.

Она представляет собой графическое представление функциональности системы и описывает, как акторы (пользователи, внешние системы, устройства и т.д.) взаимодействуют с системой, используя ее функциональные возможности для достижения определенных целей.

Для создания диаграммы вариантов использования необходимо определить список акторов и список вариантов использования, а затем связать их между собой, используя отношения. Кроме того, можно добавить описание каждого варианта использования и дополнительные атрибуты, такие как действия, предусловия и постусловия. На диаграммах вариантов использования отображается взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему. Из диаграмм вариантов использования можно получить довольно много информации о системе. Этот тип диаграмм описывает общую функциональность системы. Для описания работы системы на концептуальном уровне, отражения отношений между действующими лицами и функциями ниже представлена диаграмма вариантов использования (рисунок 2.3).



Рисунок 2.3 – *Use case* диаграмма

В таблице 7 приведем подробное описание вариантов использования. В

данной таблице описываются ход действий, пред- и пост-условия, альтернативные потоки действий, триггеры и исключения.

Таблица 1 – Описание вариантов использования

|  |  |
| --- | --- |
| Категории варианта использования | Описание |
| **Имя варианта использования** | UC-1 **Купить криптовалюту** |
| Основное действующее лицо | Пользователь |
| Описание | Пользователь может выбрать криптовалюту, которую он хочет купить, указать сумму, которую он хочет потратить, и получить конвертированную криптовалюту на свой счет. После этого пользователь может использовать программное средство для анализа тренда криптовалют в режиме реального времени. |
| Триггер | Пользователь хочет купить криптовалюту на платформе. |
| Предварительное условия | Пользователь должен иметь зарегистрированную учетную запись на платформе и быть авторизованным в системе. |
| Выходные условия | Пользователь получает криптовалюту на свой счет. |
| Нормальное направление развития варианта использования | Вход в учетную запись: пользователь должен войти в свою учетную запись на платформе, используя логин и пароль.  Выбор криптовалюты: после входа в учетную запись пользователь должен выбрать криптовалюту, которую он хочет купить, из списка поддерживаемых на платформе.  Ввод суммы: после выбора криптовалюты пользователь должен ввести сумму, которую он хочет потратить на покупку криптовалюты.  Установка цены покупки: платформа автоматически рассчитывает цену покупки на основе текущей рыночной цены выбранной криптовалюты. Пользователь может либо подтвердить цену, либо установить свою собственную цену покупки.  Подтверждение сделки: после ввода суммы и установки цены покупки пользователь должен подтвердить сделку и выполнить необходимые  Оплата: после подтверждения сделки пользователь должен выбрать способ оплаты и выполнить оплату с помощью доступных на платформе методов. |
| Исключения при выполнении  варианта использования |  |
| Альтернативный поток действий при  исполнении варианта использования |  |
| Постусловия после выполнения  варианта использования |  |
| Связь с другими вариантами  использования | Отсутствует |
| **Имя варианта использования** | UC-2 **Просмотр цены** |
| Основное действующее лицо | Пользователь |
| Описание | позволяет пользователю программного средства просмотреть текущую цену выбранной криптовалюты в режиме реального времени. |
| Триггер | Пользователь запустил программное средство и выбрал опцию Просмотр цены. |
| Предварительное условия | Пользователь должен иметь доступ к интернету и выбрать криптовалюту, цену которой он хочет узнать. |
| Выходные условия | Пользователь видит текущую цену выбранной криптовалюты. |
| Нормальное направление развития варианта использования | Пользователь может продолжать работу с программным средством, используя полученную информацию для анализа тренда криптовалюты и принятия решений по ее покупке/продаже. |
| **Имя варианта использования** | UC-3 **Продажа криптовалюты** |
| Основное действующее лицо | Пользователь |
| Описание | позволяет пользователю программного средства продать выбранную криптовалюту. |
| Триггер | Пользователь запустил программное средство и выбрал опцию «Продать криптовалюту». |
| Предварительное условия | Пользователь должен иметь доступ к интернету, иметь на своем счете достаточное количество выбранной криптовалюты и выбрать способ продажи. |
| Выходные условия | Пользователь успешно продал выбранную криптовалюту и получил соответствующую сумму в выбранной валюте. |
| Нормальное направление развития варианта использования | Пользователь может использовать полученные средства для дальнейших инвестиций или вывода на свой банковский счет. |
| **Имя варианта использования** | UC-4 **Покупка криптовалюты** |
| Основное действующее лицо | Пользователь |
| Описание | Бизнес-процесс позволяет пользователю программного средства купить выбранную криптовалюту. |
| Триггер | Пользователь запустил программное средство и выбрал опцию «Купить криптовалюту». |
| Предварительное условия | Пользователь должен иметь доступ к интернету, иметь на своем счете достаточное количество выбранной валюты и выбрать способ покупки. |
| Выходные условия | Пользователь успешно купил выбранную криптовалюту и соответствующая сумма была списана с его счета в выбранной валюте. |
| Нормальное направление развития варианта использования | Пользователь может использовать купленную криптовалюту для дальнейших инвестиций или обмена на другие валюты. |
| **Имя варианта использования** | UC-5 **Перевод криптовалюты** |
| Основное действующее лицо | Пользователь |
| Описание | Позволяет пользователю программного средства сделать перевод выбранной криптовалюты на другой счет. |
| Триггер | Пользователь запустил программное средство и выбрал опцию «Перевести криптовалюту». |
| Предварительное условия | Пользователь должен иметь доступ к интернету, иметь на своем счете достаточное количество выбранной криптовалюты, указать адрес кошелька получателя и ввести необходимую сумму для перевода. |
| Выходные условия | Пользователь успешно перевел выбранную криптовалюту на указанный адрес кошелька получателя. |
| Нормальное направление развития варианта использования | Пользователь может использовать переведенную криптовалюту для дальнейших инвестиций или обмена на другие валюты, а также совершать более быстрые и безопасные транзакции без участия посредников. |
| **Имя варианта использования** | UC-6 **Посмотреть объем на продажу** |
| Основное действующее лицо | Пользователь |
| Описание | Позволяет пользователю программного средства посмотреть объем криптовалюты, которая доступна на продажу на конкретной бирже. |
| Триггер | Пользователь запустил программное средство и выбрал опцию «Посмотреть объем на продажу». |
| Предварительное условия | Пользователь должен иметь доступ к интернету и иметь учетную запись на бирже, на которой он хочет посмотреть объем на продажу выбранной криптовалюты. |
| Выходные условия | Пользователь получил информацию о доступном объеме криптовалюты на продажу на выбранной бирже. |
| Нормальное направление развития варианта использования | Пользователь может использовать эту информацию для определения лучшей цены для покупки или продажи криптовалюты на выбранной бирже. Это может помочь пользователю сократить риски и получить наибольшую выгоду при совершении сделок на бирже. |
| **Имя варианта использования** | UC-7 **Посмотреть изменение цен за сутки** |
| Основное действующее лицо | Пользователь |
| Описание | Данный бизнес-процесс позволяет пользователю программного средства просмотреть изменение цен за сутки на выбранные криптовалютные пары. |
| Триггер | Пользователь открывает программное средство и выбирает необходимую криптовалютную пару. |
| Предварительное условия | Программное средство запущено, имеется доступ к интернету, на бирже имеется информация о ценах за сутки. |
| Выходные условия | Пользователь получает информацию о изменении цен на выбранную криптовалютную пару за сутки |
| Нормальное направление развития варианта использования | Расширение функционала программного средства для анализа изменения цен на более длительный период времени (например, неделю, месяц), а также для прогнозирования изменения цен на основе анализа тренда и факторов, влияющих на криптовалютный рынок. |
| **Имя варианта использования** | UC-8 **Посмотреть изменение цен за сутки** |
| Основное действующее лицо | Пользователь |
| Описание | Данный бизнес-процесс позволяет пользователю программного средства просмотреть изменение цен за сутки на выбранные криптовалютные пары. |
| Триггер | Пользователь открывает программное средство и выбирает необходимую криптовалютную пару. |
| Предварительное условия | Программное средство запущено, имеется доступ к интернету, на бирже имеется информация о ценах за сутки. |
| Выходные условия | Пользователь получает информацию о изменении цен на выбранную криптовалютную пару за сутки |
| Нормальное направление развития варианта использования | Расширение функционала программного средства для анализа изменения цен на более длительный период времени (например, неделю, месяц), а также для прогнозирования изменения цен на основе анализа тренда и факторов, влияющих на криптовалютный рынок. |
| **Имя варианта использования** | UC-9 **Посмотреть курс** |
| Основное действующее лицо | Пользователь |
| Описание | Данный бизнес-процесс позволяет пользователю посмотреть курс выбранной криптовалюты на текущий момент времени. |
| Триггер | Желание пользователя посмотреть текущий курс криптовалюты. |
| Предварительное условия | Пользователь должен открыть программное средство для конвертации и анализа тренда криптовалют в режиме реального времени и иметь доступ к интернету. |
| Выходные условия | Пользователь получает информацию о текущем курсе выбранной криптовалюты. |
| Нормальное направление развития варианта использования | Развитие данного бизнес-процесса может быть направлено на предоставление пользователю дополнительной информации о курсе, например, графическое отображение изменения курса за определенный период времени. Также возможно предоставление пользователю информации о курсе криптовалют на других биржах и учет текущих курсов обмена для конвертации криптовалюты. |

Таким образом, был проведен анализ требований к разрабатываемому

программному средству, была составлена спецификация функциональных

требований на разработку программы

Нефункциональные требования – это требования, которые не связаны с

конкретными функциями приложения, но являются важными для его работ. Примеры нефункциональных требований для веб-приложения:

Требования операционной среды:

– *ОЕ*-1 – ПС должно разрабатываться на ОС *Windows*;

– *ОЕ*-2 – язык для написания ПО – *C*#, *JavaScript*, фреймворк .*net* *MVC*, предназначен для написания веб-приложений

– *ОЕ*-3 – для корректной работы веб-приложения, необходимо будет иметь стабильное подключение к сети Интернет

Требования по удобству использования:

− *USE*-1 – приложение должно быть простым и интуитивно понятным для пользователей. Все функции и элементы управления должны быть легко доступны и понятны;

− *USE*-2 – навигация в приложении должна быть легкой и понятной. Пользователи должны иметь возможность легко находить нужную информацию и переходить между различными страницами приложения;

− *USE*-3 – элементы управления, макет и дизайн должны быть консистентными по всему приложению. Это поможет пользователям быстро ориентироваться и понимать, как использовать функции приложения;

− *USE*-4 – приложение должно быстро реагировать на действия пользователя и быстро загружаться;

− *USE*-5 – система должна давать обратную связь пользователю (индикатор загрузки сайта, окна ошибок, окно 404, подсказки);

Требования к производительности:

− *PER*-1 – приложение должно работать быстро и отзывчиво. Пользователи не должны ждать дольше 3-х секунд загрузки страниц и выполнения действий;

− *PER*-2 – код приложения должен быть оптимизирован, чтобы работать быстро и эффективно;

− *PER*-3 – база данных приложения должна быть оптимизирована для быстрой загрузки и запросов;

Требования к безопасности:

− *SEC*-1 – приложение должно иметь механизм аутентификации и авторизации пользователей.

− *SEC*-2 – доступ к получению и изменению данных должны иметь исключительно авторизованные пользователи.

**2.3 Постановка задачи на разработку программного средства**

Задача данного курсового проекта – спроектировать программную поддержку создания сценариев автоматизированного обучения клиент-серверной архитектуры.

Особенности такой модели заключаются в том, что пользователь отправляет определенный запрос на сервер, где тот системно обрабатывается и конечный результат отсылается клиенту. В возможности сервера входит одновременное обслуживание сразу нескольких клиентов.

Если одновременно поступает более одного запроса, то такие запросы устанавливаются в определенную очередь и сервером выполняются по

Очереди. Порой запросы могут иметь свои собственные приоритеты. Часть

запросов с более высокими приоритетами будут постоянно выполняться в

первоочередном порядке.

Функции, которые могут реализоваться на стороне сервера:

* хранение, защита и доступ к данным;
* работа с поступающими клиентскими запросами;
* процесс отправки ответа клиенту.

На стороне клиента:

* площадка по предоставлению пользовательского графического интерфейса;
* формулировка запроса к серверу и его последующая отправка;
* получение итогов запроса и отправка дополнительной группы команд (запросы на добавление, обновление информации, удаление группы данных).

Архитектура системы клиент-сервер формулирует принципы виртуального общения между локальными компьютерами, а все правила и принципы взаимодействия находятся внутри протокола – это особый набор правил, на основании которого выполняется точное взаимодействие между компьютерами внутри виртуальной сети.

В нашем случае взаимодействие клиента и сервера осуществляется по совокупности протоколов *TCP*/*IP*.

Приложение разделено на три уровня и спроектировано с применением паттерна *MVC* (сокращение от *Model*-*View*-*Controller*) – это архитектурный паттерн, который делит модули на три группы:

* модель (*model*);
* представление (*view*);
* контроллер (*controller*).

В разрабатываемом приложении модель будет содержать данные, за которыми приходит пользователь, запускает сервер, который принимает запросы от клиента и вызывает соответствующие метода класса, непосредственно взаимодействующего с базой данных и выполняющего запросы на выборку, обновление, добавление и удаление записей.

Представление – графический пользовательский интерфейс, упрощающий работу пользователя, скрывая особенности реализации его запросов в других модулях.

Контроллеры – «тонкая прослойка» между моделью и представлением, которая считывает события в пользовательском интерфейсе и отправляет соответствующие запросы серверу.

Программный код должен написан объектно-ориентированном языке программирования высокого уровня *С*#. Его преимуществами являются удобство, безопасность и тот факт, что написанные на нём приложения запускаются на любой поддерживающей его платформе.

Для проектирования серверной части программного средства был использован Rider. Rider - это интегрированная среда разработки (IDE) от JetBrains, которая поддерживает множество языков программирования, в том числе C#. Rider предоставляет множество полезных функций, таких как автоматическая проверка кода, анализ производительности и поддержка отладки. Rider также имеет интеграцию с различными системами контроля версий, включая Git, Subversion и Mercurial.

Выбор Rider для разработки серверной части обусловлен тем, что это мощный инструмент для разработки на C# с множеством полезных функций и интеграцией с системами контроля версий.

PostgreSQL:

PostgreSQL - это открытая реляционная система управления базами данных (СУБД), которая предоставляет широкий спектр возможностей, включая хранение, поиск и обработку данных. PostgreSQL является надежной и производительной СУБД с многолетней историей и активным сообществом разработчиков.

Выбор PostgreSQL для бэкэнд-разработки обусловлен тем, что это надежная, производительная и масштабируемая СУБД, которая может поддерживать большие объемы данных.

DataGrip:

DataGrip - это интегрированная среда разработки для работы с базами данных, разработанная JetBrains. DataGrip поддерживает множество СУБД, включая PostgreSQL, MySQL и Oracle. DataGrip предоставляет множество полезных функций, таких как визуальный редактор SQL-запросов, инструменты для автоматизации задач и удобный пользовательский интерфейс.

Таблица 11 – Инфраструктура разработки программного средства

|  |  |
| --- | --- |
| Наименование  утилиты | Направление использования |
| Rider | Среда для разработки исходного кода бэкэнда |
| Datagrip | Реляционная система управления базами  данных |
| VSCode | Среда для разработки исходного кода фронтенда |
| GitHub | Веб-сервис для хостинга |
| Git | Система управления версиями |

Разработка программного средства «Конвертация и анализ тренда криптовалют в режиме реального времени» на основе технологии .NET и языка программирования C# позволит создать масштабируемое, быстрое и надежное приложение, способное обрабатывать большие объемы данных.

# **3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

## **3.1 Проектирование пользовательского интерфейса**

Загрузка…

* 1. **Обоснование выбора компонентов и технологий для**

**реализации программного средства**

В данном разделе будет рассмотрено обоснование выбора компонентов и технологий для реализации программного средства для Конвертации и анализа тренда криптовалют в режиме реального времени.

В качестве основной платформы для разработки был выбран *.Net* *Framework*, так как это современный и широко используемый фреймворк для создания приложений под *Windows*. Благодаря большой популярности, в *.Net* *Framework* имеется огромное количество библиотек и инструментов, которые позволяют разрабатывать высококачественное программное обеспечение быстро и эффективно. Также *.Net Framework* обладает высокой стабильностью и безопасностью, что особенно важно для работы с криптовалютами.

В качестве основного языка программирования для реализации программного средства был выбран язык *C#.* Этот язык программирования является одним из самых популярных и широко используемых языков в *.Net* Framework. Он обладает высокой производительностью, обширной библиотекой классов и инструментов, что позволяет быстро и эффективно разрабатывать приложения.

Для создания пользовательского интерфейса были выбраны веб-технологии, так как они позволяют создавать кроссплатформенные приложения с высокой степенью доступности. Веб-технологии также позволяют быстро разрабатывать интерфейс с помощью *HTML*, *CSS* и *JavaScript*, что упрощает процесс разработки.

*PostgreSQL* – это мощная реляционная система управления базами данных (СУБД), которая предлагает множество функций и возможностей, позволяющих удовлетворить большинство потребностей в области хранения, обработки и доступа к данным. Вот некоторые причины, по которым *PostgreSQL* может быть отличным выбором для вашего проекта:

Бесплатная и открытая платформа: *PostgreSQL* распространяется на условиях лицензии *BSD*, что означает, что вы можете использовать, изменять и распространять ее бесплатно.

Высокая надежность и производительность: *PostgreSQL* обладает высокой степенью надежности и производительности благодаря своей способности обрабатывать большие объемы данных, поддерживать многопоточность и обеспечивать устойчивость к отказам.

Расширяемость: *PostgreSQL* является расширяемой платформой и позволяет пользователям разрабатывать свои собственные модули расширения, которые могут быть интегрированы в систему для решения специфических задач.

Большое сообщество пользователей и разработчиков: *PostgreSQL* имеет активное сообщество пользователей и разработчиков, которое предоставляет множество бесплатных инструментов, библиотек и решений для упрощения разработки и управления базами данных.

Поддержка стандартов: *PostgreSQL* полностью совместима со стандартами *ANSI* *SQL* и *SQL* 2016, что облегчает перенос приложений с других платформ.

Поддержка репликации и кластеризации: *PostgreSQL* поддерживает функции репликации и кластеризации, что обеспечивает высокую доступность и масштабируемость базы данных.

Безопасность: *PostgreSQL* обладает мощными механизмами безопасности, включая авторизацию пользователей, контроль доступа и поддержку SSL. В целом, *PostgreSQL* является мощной, надежной и расширяемой системой управления базами данных, которая предлагает множество функций и возможностей для удовлетворения различных потребностей в области хранения и обработки данных.

Таким образом, выбор компонентов и технологий для реализации программного средства для Конвертации и анализа тренда криптовалют в режиме реального времени был основан на высокой производительности, широкой доступности, безопасности и удобстве использования.

## **3.3 Проектирование схемы информационной модели предметной**

## **области**

Конвертер и анализатор тренда криптовалюты в режиме реального времени - это программное средство, которое предназначено для сбора, обработки и анализа данных о криптовалютах. Программное средство получает данные с различных бирж, агрегирует их и предоставляет пользователю готовую информацию в удобном для восприятия виде.

Для хранения данных программное средство использует базу данных, которая проектируется с учетом требований предметной области. База данных состоит из нескольких таблиц, каждая из которых отображает определенную сущность в предметной области.

Таблицы в базе данных могут содержать следующие данные:

1. Криптовалюта - таблица содержит информацию о каждой криптовалюте, включая ее *ID*, название, символ, текущую цену, изменение цены за последние 24 часа, объем торгов за последние 24 часа и другие характеристики.
2. Транзакция - таблица содержит информацию о каждой совершенной транзакции, включая ее *ID*, *ID* пользователя, который совершил транзакцию, ID криптовалюты, сумму, цену, по которой произошла транзакция, и другие характеристики.
3. Пользователь - таблица содержит информацию о каждом пользователе, включая его *ID*, имя, фамилию, логин, пароль, адрес электронной почты и другие характеристики.
4. Портфель - таблица содержит информацию о портфеле каждого пользователя, включая его *ID*, *ID* пользователя, *ID* криптовалюты, количество криптовалюты и другие характеристики.
5. История транзакций - таблица содержит информацию о каждой транзакции, которая была выполнена пользователем, включая ее *ID*, *ID* пользователя, ID криптовалюты, сумму, цену, по которой произошла транзакция, и другие характеристики.

В целом, база данных является центральным элементом программного средства конвертации и анализа тренда криптовалюты в режиме реального времени, поскольку она хранит и обрабатывает все данные, которые используются в программном средстве. Как только данные получены, они сохраняются в базу данных и используются для создания отчетов и графиков, которые отображают текущий тренд криптовалюты.

Проектирование любой информационной системы начинается с представления предметной области в виде набора взаимосвязанных сущ-ностей, имеющих свои атрибуты. В дальнейшем представление информации о проектируемой системе пытаются упростить, убрать или исключить номалии модификации, удаления и добавления, предотвратить избыточное дублирование данных, сделать более ее читабельной, а само ее представление более понятным и удобным. Таким образом, процесс проектирования представляет собой процесс нормализации схем отношений, при этом каждая следующая нормальная форма обладает свойствами, которая имела предыдущая нормальная форма и некоторыми новыми, осуществляющими улучшение структуры модели. В теории реляционных баз данных чаще всего используются первые три нормальные формы. Каждой из них соответствует определенный набор ограничений.

Концептуальное проектирование – анализ предметной области и ее описание. На рисунке 3.22 представлена ER-модель предметной области, в которой отражены основные сущности и их связи между ними.

Общий вид логической схемы базы данных приведен на рисунке 3.

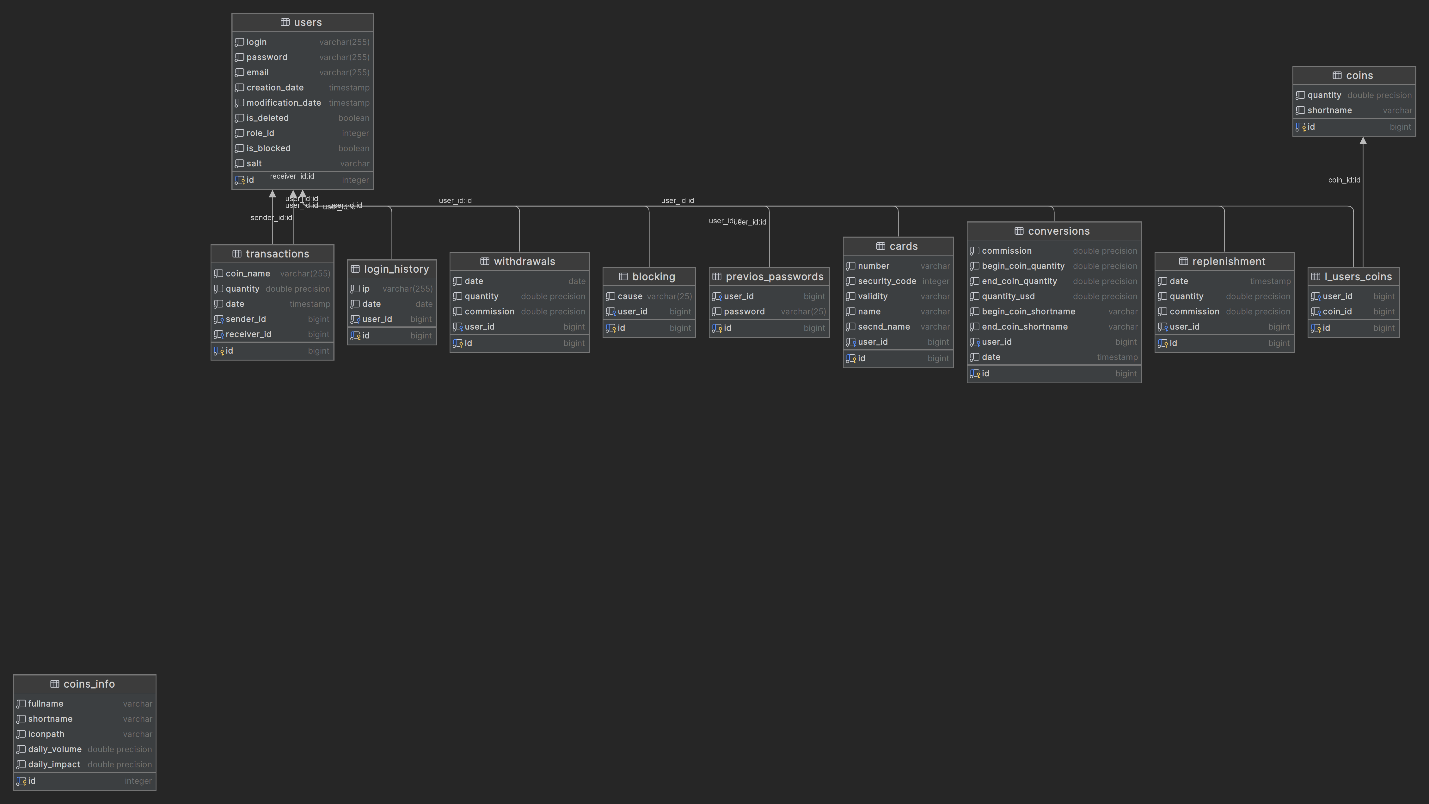


Рисунок 3. – «Схема базы данных»

В сущности «Перевод» первичным ключом является атрибут *ID* артиста. Описание сущности «Перевод», предназначенное для хранения данных об Переводах криптовалюты, представлено в таблице 1.

Таблица 1 – Описание отношения «Перевод»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID перевода | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор |
| Дата | Обязательное | Дата выполнения вывода средств |
| Количество | Обязательное | Количество средств, выводимых пользователем |
| Комиссия | Обязательное | Комиссия, удерживаемая за операцию вывода средств |
| ID пользователя | Обязательное | Идентификатор пользователя, производящего вывод |

В сущности «Пользователь» первичным ключом является атрибут *ID* пользователя. Описание сущности «Пользователь», предназначенное для хранения данных об пользователях, представлено в таблице 2.

Таблица 2 – Описание отношения «Пользователь»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID пользователя | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор пользователя, автоматически генерируемый при создании записи |
| Имя пользователя | Обязательное | Логин пользователя, строка длиной до 255 символов, обязательное поле. |
| Пароль | Обязательное | Пароль пользователя, строка длиной до 255 символов, обязательное поле |
| Адрес электронной почты | Обязательное | Электронная почта пользователя, строка длиной до 255 символов, обязательное поле |
| Дата создания | Обязательное | Дата создания записи о пользователе, обязательное поле |
| Дата редактирования | Обязательное | Дата последнего изменения записи о пользователе, обязательное поле |
| Переменная хранящая статус удаления аккаунта | Обязательное | Флаг удаления пользователя (true - удален, false - не удален), обязательное поле |
| ID роли пользователя | Обязательное | Идентификатор роли пользователя, обязательное поле |
| Переменная хранящая статус блокировки аккаунта | Обязательное | Флаг блокировки пользователя (true - заблокирован, false - не заблокирован), обязательное поле |
| Соль | Обязательное | Случайная строка, добавляемая к паролю для увеличения его безопасности, обязательное поле |

В сущности «Транзакция» первичным ключом является атрибут *ID* транзакции. Описание сущности «Транзакция», предназначенное для хранения данных об транзакциях, представлено в таблице 3.

Таблица 3 – Описание отношения «Транзакция»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID транзакции | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор транзакции |
| Название монеты | Обязательное | Название криптовалюты, которая была передана |
| Количество монеты | Обязательное | Количество криптовалюты, которая была передана |
| Дата | Обязательное | Дата и время проведения транзакции |
| ID отправителя | Обязательное | Идентификатор пользователя-отправителя |
| ID получателя | Обязательное | Идентификатор пользователя-получателя |

В сущности «Предыдущие пароли» первичным ключом является атрибут *ID* транзакции. Описание сущности «Предыдущие пароли», предназначенное для хранения данных об предыдущих паролях, представлено в таблице 4.

Таблица 4 – Описание отношения «Предыдущие пароли»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID записи | Обязательное автоматическое (ключевое поле) | Идентификатор предыдущего пароля (первичный ключ) |
| ID пользователя | Обязательное | Идентификатор пользователя, которому принадлежит предыдущий пароль (внешний ключ, связанный с таблицей «Пользователи») |
| Пароль | Обязательное | Зашифрованный предыдущий пароль пользователя |

В сущности «Пополнение» первичным ключом является атрибут *ID* пополнения. Описание сущности «Пополнение», предназначенное для хранения данных об Пополнениях баланса, представлено в таблице 5.

Таблица 5 – Описание отношения «Пополнение»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID пополнения | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор пополнения |
| Дата | Обязательное | Дата и время пополнения |
| Количество монет | Обязательное | Количество средств, пополненных пользователем |
| Комиссия | Обязательное | Комиссия, взимаемая при пополнении |
| ID пользователя | Обязательное | Идентификатор пользователя, который пополнил свой баланс, внешний ключ, ссылается на таблицу пользователей |

В сущности «История входов» первичным ключом является атрибут *ID* история входов. Описание сущности «История входов», предназначенное для хранения данных об истории входов в аккаунт, представлено в таблице 6.

Таблица 6 – Описание отношения «История входов»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID входа | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор записи в таблице |
| IP-адресс устройства | Обязательное | IP-адрес пользователя, который выполнил вход |
| Дата | Обязательное | Дата и время входа пользователя |
| ID пользователя | Обязательное | Уникальный идентификатор пользователя, который выполнил вход, связан с таблицей «Пользователи» через внешний ключ. |

В сущности «Связующая таблица пользователей и монет» первичным ключом является атрибут *ID* записи. Описание сущности «Связующая таблица пользователей и монет», предназначенное связи пользователей и монет, представлено в таблице 7.

Таблица 7 – Описание отношения «Связующая таблица пользователей и монет»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID записи | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор связки пользователя с монетой |
| ID пользователя | Обязательное | Идентификатор пользователя, которому принадлежит монета |
| ID монеты | Обязательное | Идентификатор монеты, которая принадлежит пользователю. |

В сущности «Кредитная карта» первичным ключом является атрибут *ID* кредитной карты. Описание сущности «Кредитная карта», предназначенное связи пользователей и монет, представлено в таблице 8.

Таблица 8 – Описание отношения «Кредитная карта»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID карты | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор карты |
| Номер карты | Обязательное | Номер карты |
| Код безопасности | Обязательное | Код безопасности карты |
| Срок действия | Обязательное | Срок действия карты |
| Имя владельца | Обязательное | Имя владельца карты |
| Фамилия владельца | Обязательное | Фамилия владельца карты |
| ID пользователя | Обязательное | Идентификатор пользователя, которому принадлежит карта, внешний ключ к таблице «Пользователи» |

В сущности «Монета» первичным ключом является атрибут *ID* монеты. Описание сущности «Монета», предназначенное для хранения данных о монетах, представлено в таблице 9.

Таблица 9 – Описание отношения «Монета»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID монеты | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор монеты |
| Количество | Обязательное | Количество монет в наличии |
| Название монеты | Обязательное | Короткое название монеты (например, BTC, ETH) |

В сущности «Блокировка» первичным ключом является атрибут *ID* блокировки. Описание сущности «Блокировка», предназначенное для хранения данных о блокировках пользователей, представлено в таблице 10.

Таблица 10 – Описание отношения «Блокировка»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID блокировки | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор блокировки. Автоматически генерируется каждый раз при создании новой записи. |
| Причина | Обязательное | Причина блокировки. |
| Пользователь | Обязательное | Идентификатор пользователя, который был заблокирован. Связан с таблицей «Пользователи». |

В сущности «Вывод» первичным ключом является атрибут *ID* вывода. Описание сущности «Вывод», предназначенное для хранения данных о выводах криптовалюты пользователей, представлено в таблице 11.

Таблица 11 – Описание отношения «Вывод»

|  |  |  |
| --- | --- | --- |
| **Название атрибута** | **Заполнение** | **Описание** |
| ID вывода | Обязательное автоматическое (ключевое поле) | Уникальный идентификатор вывода средств |
| Дата | обязательное | Дата вывода средств |
| Количество | обязательное | Количество выводимых средств |
| Комиссия | обязательное | Комиссия за вывод средств |
| ID пользователя | обязательное | Идентификатор пользователя, который инициировал вывод средств |

**3.4 Архитектурные решения**

Для разрабатываемого программного средства будут использованы паттерн Клиент-сервер, т. к. программное средство разрабатывается с клиент-серверной архитектурой, а также паттерн Модель-представление-контроллер.

Паттерн применяется, когда есть общие ресурсы и сервисы, к которым нужно обеспечить доступ большого количества распределенных клиентов, и при этом необходимо контролировать доступ или качество обслуживания.

В подходе «клиент-сервер» компоненты и соединительные элементы обладают определенным поведением.

Компоненты, называемые «клиентами», отправляют запросы компоненту, называемому «сервер», и ждут ответа.

Компонент «сервер» получает запрос от клиента и отправляет ему ответ.

Подход «клиент-сервер» может применяться в моделировании части системы, имеющей много компонентов, отправляющих запросы (это «клиенты») другому компоненту (это «сервер»), который обеспечивает работу сервисов, – например, онлайн-приложения (электронная почта, обмен документами и банковское дело).

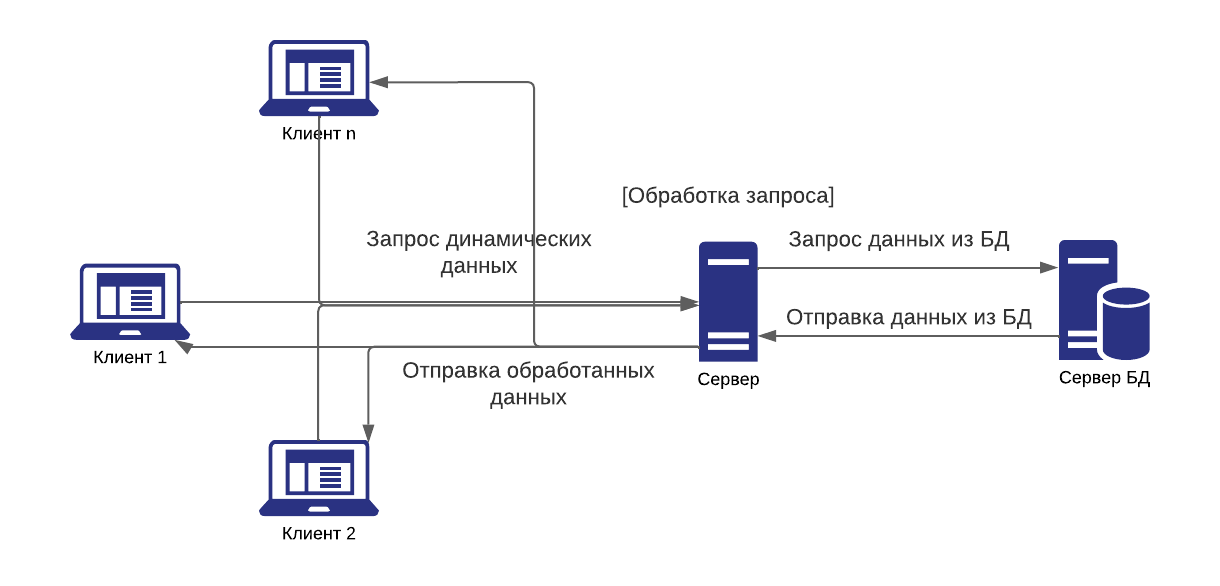
Для реализации архитектуры «клиент-сервер» для данного программного средства необходимо создать трехуровневую систему, состоящую из приложения клиента, приложения сервера, а также сервера с базой данных. Визуализация структуры трехуровневой системы представлена на рисунке 3.

Рисунок 3 – Архитектура трехуровневой системы «Клиент-Сервер»

Такой подход с к разработка данного программного средства позволит делегировать только необходимый минимум задач на каждый уровень системы, что минимизирует общую нагрузку и поспособствует быстродействию системы.

## **3.5 Описание динамических аспектов поведения объектов системы**

Для описания моделей представления программного средства используются диаграммы *UML*.

*Unified* *Modeling* *Language* (*UML*) — унифицированный язык моделирования. Расшифруем: «modeling» подразумевает создание модели, описывающей объект. «*Unified*» (универсальный, единый) — подходит для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов. *UML* описывает объект в едином заданном синтаксисе, поэтому, где бы вы не нарисовали диаграмму, ее правила будут понятны для всех, кто знаком с этим графическим языком — даже в другой стране [17].

Диаграмма в языке моделирования *UML* – это наглядное представление некоей совокупности элементов модели системы в виде графа, на котором дуги (отношения) связывают вершины (сущности). В своём графическом виде различные виды диаграмм *UML* (диаграммы классов, компонентов, объектов и др.) применяются для визуализации разных аспектов устройства или поведения моделируемой системы.

Для описания программного средства в данном курсовом проекте будут использоваться следующие диаграммы:

− диаграмма последовательностей;

− диаграмма состояний;

− диаграмма развертывания;

− диаграмма классов.

Для того чтобы рассмотреть упорядоченное во времени взаимодействие объектов системы, рассмотрим диаграмму последовательностей (рисунок 3.25).

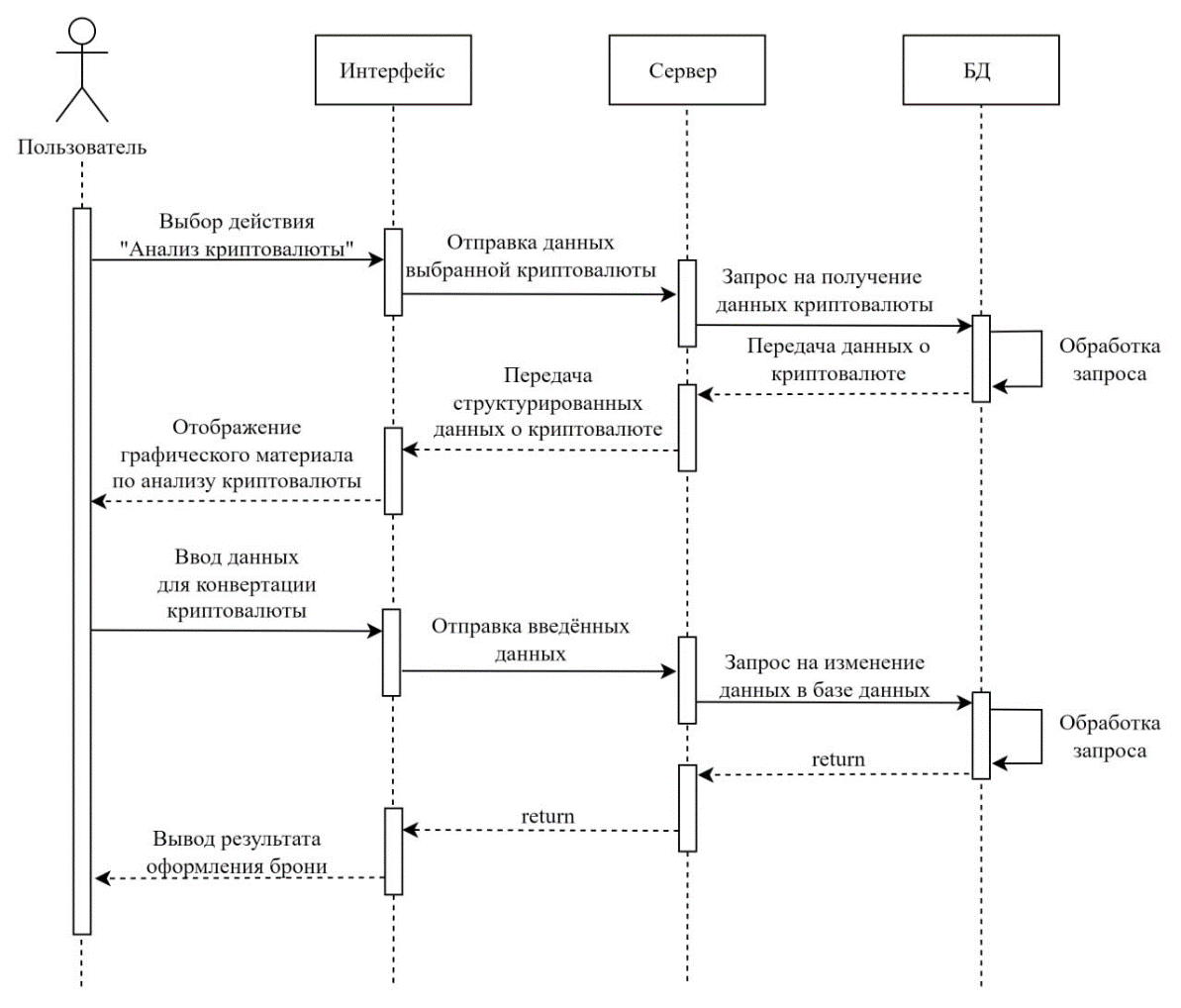
****

Рисунок 3.25 – Диаграмма последовательности

процесса прохождения тестирования

При работе с программой пользователь будет взаимодействовать с программой путем нажатия кнопок. Сайт же в свою очередь либо будет выполнять необходимое действие (Открыть окно), либо, если действие касается работы с данными – посылать на сервер запрос. Серверу в свою очередь для работы необходимо получить информацию из базы данных.

После ее получения сервер выполняет заданное действие и возвращает результат в программу, которая в свою очередь показывает результат пользователю. Действующее лицо – пользователь ПС, который работает с сайтом через специальный пользовательский интерфейс. В таблице 16 приведены примеры отправляемых программным средством сообщений.

Таблица 16 – Описание сообщений

|  |  |  |  |
| --- | --- | --- | --- |
| Номер  сообщения | Объект-  отправитель | Объект-  получатель | Имя сообщения |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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.

Графически диаграмма деятельности представляется в форме графадеятельности, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

Далее необходимо спроектировать диаграмму деятельности. Такая диаграмма по существу представляет блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой. Активности на диаграмме “разбросаны” по беговым дорожкам, каждая из которых соответствует поведению одного из объектов (например, клиента, менеджера, веб-сервера, сервера БД и т.п.). Благодаря этому легко определить, каким из объектов выполняется каждая из активностей.

Диаграмма деятельности представлена на рисунке 3.28. Из данной диаграммы можно сказать, что деятельность прохождения тестирования проходит под контролем 3 объектов: пользователя, программного средства и БД.

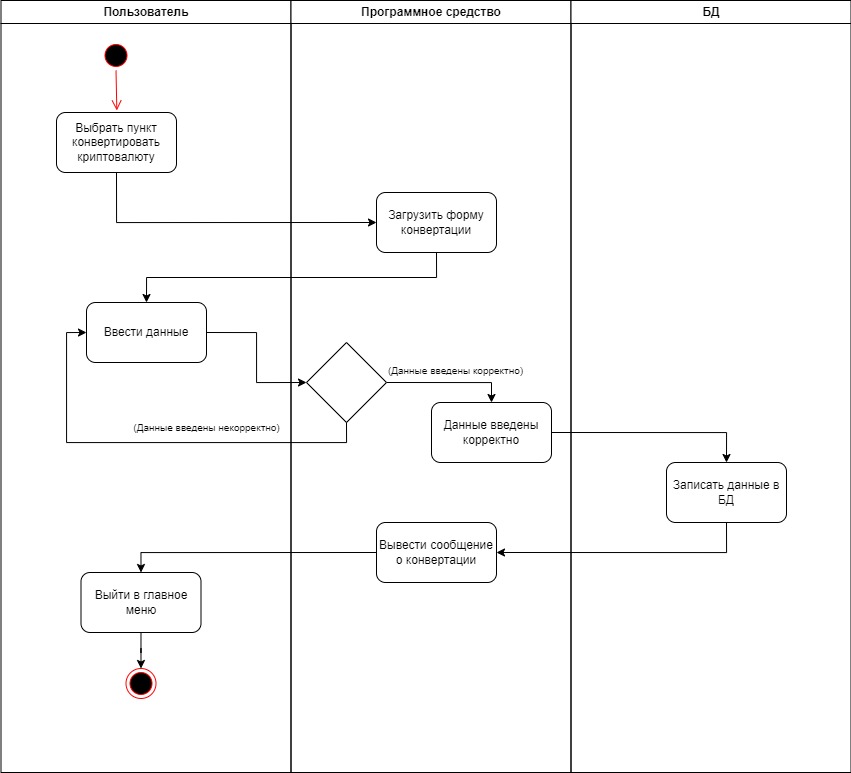


Рисунок 3.28 – Диаграмма деятельности конвертации криптовалюты

Таким образом, были описаны динамические аспекты объектов системы, составлена диаграмма деятельности и последовательности процессов прохождения тестирования.

## **3.6 Описание и разработка алгоритмов, реализующих бизнес-логику разрабатываемого программного средства**

Блок-схема – распространённый тип схем (графических моделей), описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединённых между собой линиями, указывающими направление последовательности. Алгоритм в виде блок-схемы представляет собой совокупность символов, соответствующих этапам работы алгоритма и соединяющих их линий.

Блок-схема алгоритма отправки криптовалюты от одного пользователя к другому изображена на рисунке 3. . Перед выполнением операции необходимо проверить, что отправитель и получатель не являются одним и тем же пользователем, а также что количество отправляемой криптовалюты больше нуля. Затем необходимо проверить, достаточно ли криптовалюты в кошельке отправителя для выполнения операции. Если недостаточно, то операция прерывается с сообщением об ошибке. Если же достаточно, то необходимо вычесть количество отправляемой криптовалюты из кошелька отправителя и добавить ее в кошелек получателя. Также необходимо записать транзакцию в базу данных. После успешного выполнения операции, возвращается сообщение об успешном выполнении.

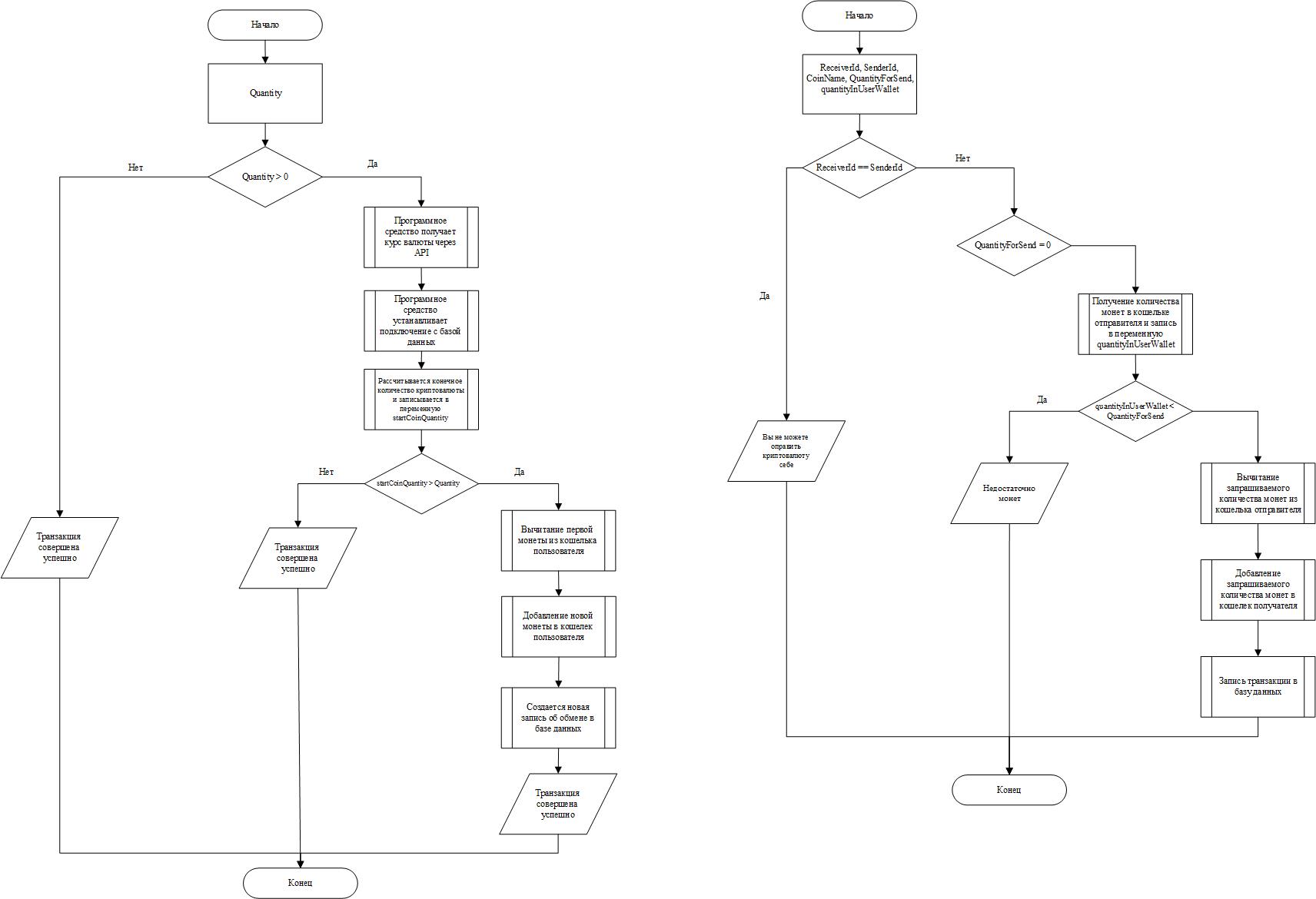


Рисунок 3 – «Блок-схема алгоритма отправки криптовалюты»

Алгоритм конвертации криптовалюты изображен на рисунке 3. . Если количество криптовалюты, которую пользователь хочет сконвертировать, равно 0, то возвращается ответ с ошибкой. Затем происходит запрос к API, чтобы получить коэффициент конвертации между двумя валютами. Если запрос выполнен успешно, то из ответа извлекается соответствующее значение.

Далее происходит проверка, достаточно ли у пользователя валюты, которую он хочет сконвертировать, для завершения конвертации. Если нет, то возвращается ответ с ошибкой. Если пользователь имеет достаточно валюты, то из его кошелька списывается необходимое количество валюты, а на его счет добавляется эквивалентное количество валюты, в которую он конвертировал свою исходную валюту. Также происходит запись новой транзакции в базу данных.

В конце метод возвращает ответ об успешном выполнении операции.

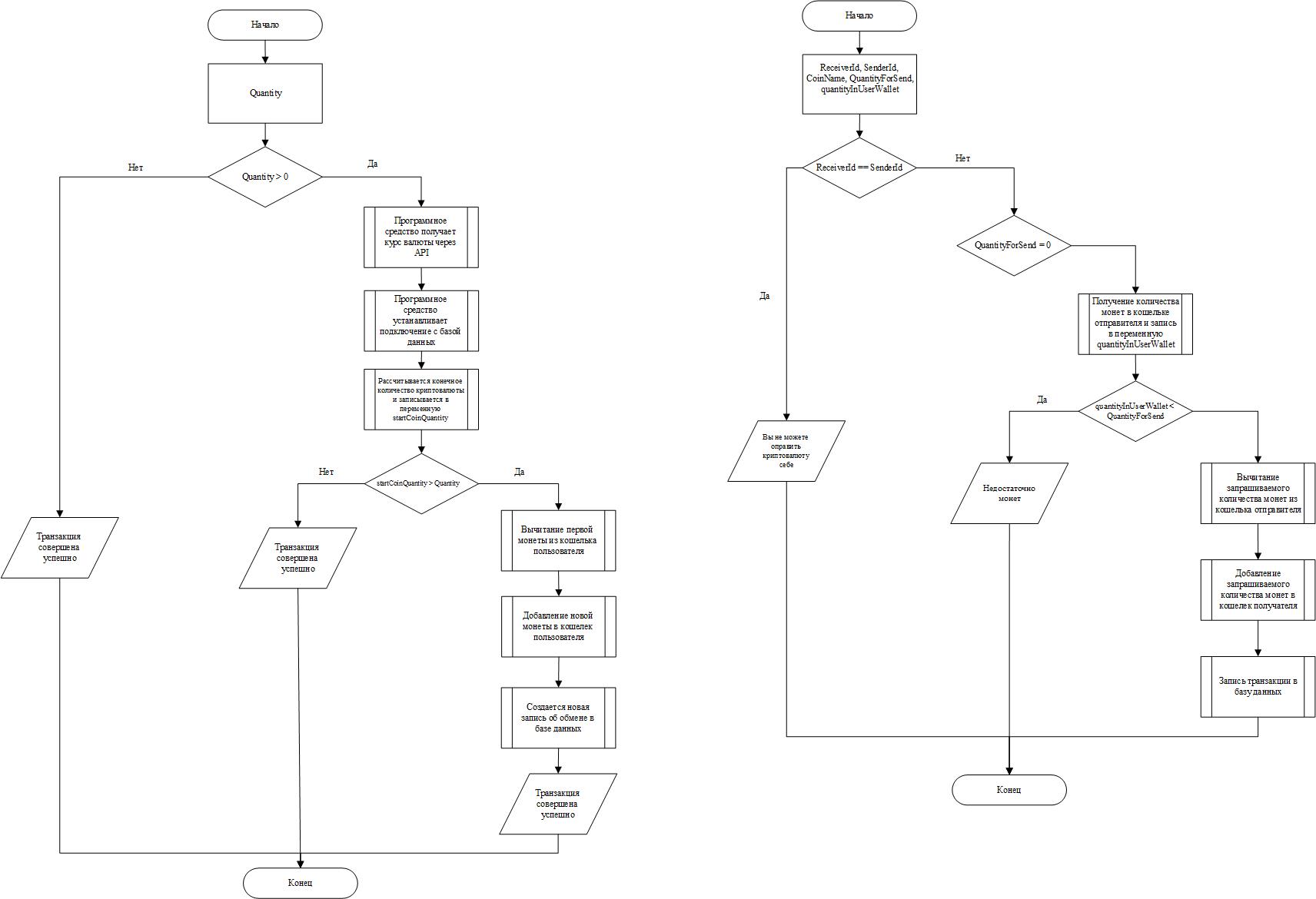


Рисунок 3. – «Блок-схема алгоритма конвертации криптовалюты»

В данном разделе были описаны алгоритмы, необходимые для реализации бизнес-логики разрабатываемого программного средства. Были описаны алгоритмы для выполнения конвертации и передачи криптовалюты между пользователями.

Также были приведены графические схемы для визуализации основных этапов алгоритмов, что позволяет лучше понимать их работу и использование в программном средстве.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была создана программа для конвертации и анализа тренда криптовалют в режиме реального времени. Был проведен анализ предметной области, что позволило определить требования к программному обеспечению и выбрать наиболее эффективный подход для решения поставленных задач.

Разработанное программное средство имеет интуитивно понятный интерфейс для пользователей и администраторов, что обеспечивает удобство использования и повышает эффективность работы с системой. Были выполнены все поставленные задачи, а программа остается готовой к дополнительным усовершенствованиям и обновлениям в будущем.

В процессе разработки проекта были значительно усовершенствованы навыки работы с языком программирования C# и базами данных, а также объектно-ориентированными языками программирования. Были использованы основные принципы ООП, что позволило создать программу, представленную в виде совокупности взаимодействующих объектов.

Итоговый результат работы - успешно созданное программное средство, которое может быть использовано для конвертации и анализа тренда криптовалют в режиме реального времени.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] What is .NET Framework? [Электронный ресурс]. – Режим доступа: <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework>. – Дата доступа: 10.11.2022.

[2] Подключение к базе данных Postgre [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/adonetcore/4.1.php. – Дата доступа: 9.11.2022.

[3] Начало работы. Visual Studio [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/tutorial/1.2.php. – Дата доступа: 9.11.2022.

[4] Создайте приложение Windows Forms в Visual Studio с помощью C# [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022. – Дата доступа: 11.11.2022.

[5] Grid View [Электронный ресурс]. – Режим доступа: https://docs.devexpress.com/WindowsForms/3464/controls-and-libraries/data-grid/views/grid-view. – Дата доступа: 12.10.2022.

[6] MediaPlayer Класс [Электронный ресурс]. – Режим доступа https://learn.microsoft.com/ru/dotnet/api/system.windows.media.mediaplayer?view=windowsdesktop-7.0. – Дата доступа: 13.10.2022.

[7] UML-диаграммы классов [Электронный ресурс]. – Режим доступа: https://prog-cpp.ru/uml-classes/. – Дата доступа: 17.11.2022.

[8] Диаграмма состояний [Электронный ресурс]. – Режим доступа: https://nationalteam.worldskills.ru/skills/proektirovanie-diagrammy-sostoyaniy-uml-statechart-diagram/ – Дата доступа: 11.10.2022.

ПРИЛОЖЕНИЕ А

(обязательное)

Отчет о проверке на заимствования в системе «Антиплагиат»

Оригинальность 3%

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг кода алгоритмов, реализующих бизнес-логику

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json.Linq;

using UP.DTO;

using UP.Models;

namespace UP.Controllers

{

[ApiController]

[Route("[controller]")]

public class TransactionController: ControllerBase

{

private readonly ILogger<TransactionController> \_logger;

public TransactionController(ILogger<TransactionController> logger)

{

\_logger = logger;

}

[HttpGet, Route("getCoinQuantity")]

public async Task<ActionResult> GetCoinQuantity(string coinName, double quantityUSD)

{

var cr = new Repositories.CurrencyRepository();

try

{

return Ok(await cr.GetCoinQuantity(quantityUSD, coinName));

}

catch(Exception)

{

\_logger.LogInformation($"Unable to return coin quantity");

return BadRequest("Unable to return coin quantity");

}

}

[HttpGet, Route("getUserConversationsHistory")]

public async Task<ActionResult> GetUserList(int id)

{

var tr = new Repositories.TransactionsRepository();

try

{

return Ok(tr.GetUserConversionsHistory(id));

}

catch(Exception)

{

\_logger.LogInformation($"Unable to return user transactions history");

return BadRequest("Unable to return user transactions history");

}

}

[HttpGet, Route("getUserDepositHistory")]

public async Task<ActionResult> GetUserDepositHistory(int id)

{

try

{

var tr = new Repositories.TransactionsRepository();

return Ok(tr.GetUserDepositHistory(id));

}

catch(Exception)

{

\_logger.LogInformation($"Unable to return user transactions history");

return BadRequest("Unable to return user transactions history");

}

}

[HttpPost, Route("convert")]

public async Task<ActionResult> Convert([FromBody] ConvertRequest request)

{

try

{

if (request.Quantity == 0)

{

return BadRequest("Error. Quantity must be above than zero");

}

// TODO govnokod

//double priceRatio = await GetPriceRatio(shortNameStart, shortNameFinal);

string apiKey = "4da2c4791b9c285b22c1bf08bc36f304ab2ca80bc901504742b9a42a814c4614";

using var httpClient = new HttpClient();

httpClient.DefaultRequestHeaders.Add("X-MBX-APIKEY", apiKey);

string url = $"https://min-api.cryptocompare.com/data/price?fsym=" + request.ShortNameStart + "&tsyms=" + request.ShortNameFinal;

var response = await httpClient.GetAsync(url);

var responseContent = await response.Content.ReadAsStringAsync();

JObject json = JObject.Parse(responseContent);

double priceRatio = (double)json[request.ShortNameFinal.ToUpper()];

double finalQuantity = priceRatio \* request.Quantity;

var ur = new Repositories.UserRepository();

double startCoinQuantityInUserWallet = ur.GetCoinQuantityInUserWallet(request.UserId, request.ShortNameStart);

if (startCoinQuantityInUserWallet < request.Quantity)

{

return BadRequest("The user doesn't have enough coins to complete the conversion");

}

var cr = new Repositories.CurrencyRepository();

cr.SubtractCoinFromUser(request.UserId, request.ShortNameStart, request.Quantity);

cr.AddCryptoToUserWallet(request.UserId, request.ShortNameFinal, finalQuantity);

var tr = new Repositories.TransactionsRepository();

tr.WriteNewConversionDataToDatabase(new Conversion(1, 0, request.Quantity, finalQuantity, await cr.GetCoinPrice(request.Quantity, request.ShortNameStart),

request.ShortNameStart, request.ShortNameFinal, request.UserId, DateTime.Now));

return Ok("Converted successfully");

}

catch (Exception e)

{

\_logger.LogInformation($"Error. Currencies have not been converted");

return BadRequest("Error. Currencies have not been converted");

}

}

/\*public async Task<double> GetPriceRatio(string shortNameStart, string shortNameFinal)

{

string apiKey = "4da2c4791b9c285b22c1bf08bc36f304ab2ca80bc901504742b9a42a814c4614";

using var httpClient = new HttpClient();

httpClient.DefaultRequestHeaders.Add("X-MBX-APIKEY", apiKey);

string url = $"https://min-api.cryptocompare.com/data/price?fsym=" + shortNameStart + "&tsyms=" + shortNameFinal;

var response = await httpClient.GetAsync(url);

var responseContent = await response.Content.ReadAsStringAsync();

JObject json = JObject.Parse(responseContent);

return (double)json[shortNameFinal.ToUpper()];

}\*/

[HttpPost, Route("buyCrypto")]

public async Task<ActionResult> BuyCrypto([FromBody] BuyCryptoRequest request)

{

/\*try

{

if (request.Quantity == 0)

{

\_logger.LogInformation($"Quantity must be above than zero");

return UnprocessableEntity("Quantity must be above than zero");

}

var ur = new Repositories.UserRepository();

var cr = new Repositories.CurrencyRepository();

double quantityUSDTInUserWallet = ur.GetCoinQuantityInUserWallet(request.UserId, "usdt");

if (quantityUSDTInUserWallet < await cr.GetCoinPrice(request.Quantity, request.CoinName))

{

\_logger.LogInformation($"Not enough balance");

return UnprocessableEntity("Not enough balance");

}

cr.SubtractCoinFromUser(request.UserId, "usdt", await cr.GetCoinPrice(request.Quantity, request.CoinName));

cr.AddCryptoToUserWallet(request.UserId, request.CoinName, request.Quantity);

\_logger.LogInformation($"UserId(" + request.UserId + ") bought " + request.Quantity + " " + request.CoinName);

return Ok("Transaction completed successfully");

}

catch (Exception e)

{

\_logger.LogInformation($"Transaction wasn't completed");

return BadRequest("Transaction wasn't completed");

}\*/

try

{

\_logger.LogInformation($"UserId:" + request.UserId + "\nCoin quantity:" + request.Quantity + "\nCoin name: " + request.CoinName);

if (request.Quantity == 0)

{

\_logger.LogInformation($"Quantity must be above than zero");

return UnprocessableEntity("Quantity must be above than zero");

}

var ur = new Repositories.UserRepository();

var cr = new Repositories.CurrencyRepository();

double quantityUSDTInUserWallet = ur.GetCoinQuantityInUserWallet(request.UserId, "usdt");

if (quantityUSDTInUserWallet < request.Quantity)

{

\_logger.LogInformation($"Not enough balance");

return UnprocessableEntity("Not enough balance");

}

cr.SubtractCoinFromUser(request.UserId, "usdt", await cr.GetCoinPrice(request.Quantity, request.CoinName));

double coinQuantity = await cr.GetCoinQuantity(request.Quantity, request.CoinName);

cr.AddCryptoToUserWallet(request.UserId, request.CoinName, coinQuantity);

\_logger.LogInformation($"UserId(" + request.UserId + ") bought " + coinQuantity + " " + request.CoinName);

return Ok("Transaction completed successfully");

}

catch (Exception e)

{

\_logger.LogInformation($"Transaction wasn't completed");

return BadRequest("Transaction wasn't completed");

}

}

[HttpPut, Route("sellCrypto")]

public async Task<ActionResult> SellCrypto([FromBody] SellCryptoRequest request)

{

try

{

if (request.QuantityForSell == 0)

{

\_logger.LogInformation($"Quantity must be above than zero");

return UnprocessableEntity("Quantity must be above than zero");

}

var ur = new Repositories.UserRepository();

var cr = new Repositories.CurrencyRepository();

double quantityInUserWallet = ur.GetCoinQuantityInUserWallet(request.UserId, request.CoinName);

if (quantityInUserWallet < request.QuantityForSell)

{

\_logger.LogInformation($"Not enough coins");

return UnprocessableEntity("Not enough coins");

}

cr.SubtractCoinFromUser(request.UserId, request.CoinName, request.QuantityForSell);

cr.AddCryptoToUserWallet(request.UserId, "usdt", await cr.GetCoinPrice(request.QuantityForSell, request.CoinName));

return Ok("Transaction completed successfully");

}

catch (Exception e)

{

\_logger.LogInformation($"Transaction wasn't completed");

return BadRequest("Transaction wasn't completed");

}

}

[HttpPost, Route("sendCrypto")]

public async Task<ActionResult> SendCrypto([FromBody] SendCryptoRequest request)

{

try

{

if (request.ReceiverId == request.SenderId)

{

return UnprocessableEntity("You can't send cryptocurrency to yourself");

}

if (request.QuantityForSend == 0)

{

return UnprocessableEntity("Quantity must be above than zero");

}

var ur = new Repositories.UserRepository();

var cr = new Repositories.CurrencyRepository();

double quantityInUserWallet = ur.GetCoinQuantityInUserWallet(request.SenderId, request.CoinName);

if (quantityInUserWallet < request.QuantityForSend)

{

return UnprocessableEntity("Not enough coins");

}

cr.SubtractCoinFromUser(request.SenderId, request.CoinName, request.QuantityForSend);

cr.AddCryptoToUserWallet(request.ReceiverId, request.CoinName, request.QuantityForSend);

cr.WriteTransactionToDatabase(request.CoinName, request.QuantityForSend, request.SenderId, request.ReceiverId);

return Ok("Transfer completed successfully");

}

catch (Exception e)

{

\_logger.LogInformation($"Transfer wasn't completed");

return BadRequest("Transfer wasn't completed");

}

}

[HttpPost, Route("replenishTheBalance")]

public async Task<ActionResult> ReplenishTheBalance(int userId, double quantityUsd)

{

var tr = new Repositories.TransactionsRepository();

try

{

tr.ReplenishTheBalance(userId, quantityUsd);

return Ok("Balance replenished successfully");

}

catch(Exception)

{

\_logger.LogInformation($"Unable to replenish the balance");

return BadRequest("Unable to replenish the balance");

}

}

[HttpPut, Route("withdrawUSDT")]

public async Task<ActionResult> WithdrawUSDT(int userId, double quantityForWithdraw)

{

var tr = new Repositories.TransactionsRepository();

try

{

if (quantityForWithdraw == 0)

{

return UnprocessableEntity("Quantity must be above than zero");

}

var ur = new Repositories.UserRepository();

var cr = new Repositories.CurrencyRepository();

double quantityInUserWallet = ur.GetCoinQuantityInUserWallet(userId, "usdt");

if (quantityInUserWallet < quantityForWithdraw)

{

\_logger.LogInformation($"Not enough balance");

return UnprocessableEntity("Not enough balance");

}

cr.SubtractCoinFromUser(userId, "usdt", quantityForWithdraw);

return Ok("Transaction was successful");

}

catch(Exception)

{

\_logger.LogInformation($"Unable to withdraw the balance");

return BadRequest("Unable to withdraw the balance");

}

}

[HttpGet, Route("getUserWithdrawalsHistory")]

public async Task<IActionResult> GetUserWithdrawalsHistory(int userId)

{

var tr = new Repositories.TransactionsRepository();

try

{

return Ok(tr.GetUserWithdrawalsHistory(userId));

}

catch(Exception)

{

\_logger.LogInformation($"Unable to get user withdrawals history");

return BadRequest("Unable to get user withdrawals history");

}

}

}

}

using Newtonsoft.Json.Linq;

using Npgsql;

using UP.Models;

using UP.Models.Base;

namespace UP.Repositories

{

public class CurrencyRepository: RepositoryBase

{

public void BuyCrypto(int userId, string shortname, double quantity)

{

AddCryptoToUserWallet(userId, shortname, quantity);

}

public void AddCryptoToUserWallet(int userId, string shortname, double quantity)

{

var ur = new UserRepository();

List<Coin> coins = ur.GetUserCoins(userId);

if (IsCoinAlreadyPurchased(coins, shortname)) {

int coinId = GetPurchasedCoinId(coins, shortname);

int coinIdInTheList = GetPurchasedCoinNumberInTheList(coins, shortname);

if (coinId != -1)

{

double finalQuantity = coins[coinIdInTheList].Quantity + quantity;

UpdateCoinQuantity(coins[coinIdInTheList].Id, finalQuantity);

}

}

else

{

using var connection = new NpgsqlConnection(connectionString);

OpenConnection(connection);

var sql = "INSERT INTO coins (shortname, quantity) " + //create coin

"VALUES (@shortname, @quantity)";

using var command = new NpgsqlCommand(sql, connection);

command.Parameters.AddWithValue("@shortname", shortname);

command.Parameters.AddWithValue("@quantity", quantity);

command.ExecuteNonQuery();

sql = "SELECT id FROM coins ORDER BY id DESC LIMIT 1;";//get coin id

using var command1 = new NpgsqlCommand(sql, connection);

int coinId = Convert.ToInt32(command1.ExecuteScalar());

command1.ExecuteNonQuery();

sql = "INSERT INTO l\_users\_coins (user\_id, coin\_id) " +//unite

"VALUES (@user\_id, @coin\_id)";

using var command2 = new NpgsqlCommand(sql, connection);

command2.Parameters.AddWithValue("@user\_id", userId);

command2.Parameters.AddWithValue("@coin\_id", coinId);

command2.ExecuteNonQuery();

CloseConnection(connection);

}

}

public void SendCrypto(int receiverId, int senderId, string shortname, double quantity)

{

SubtractCoinFromUser(senderId, shortname, quantity);

AddCryptoToUserWallet(receiverId, shortname, quantity);

}

public bool IsCoinAlreadyPurchased(List<Coin> coins, string shortName)

{

try

{

foreach (var i in coins)

{

if (i.ShortName == shortName)

{

return true;

}

}

return false;

}

catch (Exception e)

{

return false;

}

}

private int GetPurchasedCoinId(List<Coin> coins, string shortName)

{

try

{

foreach (var i in coins)

{

if (i.ShortName == shortName)

{

return i.Id;

}

}

return -1;

}

catch (Exception e)

{

return -1;

}

}

private int GetPurchasedCoinNumberInTheList(List<Coin> coins, string shortName)

{

try

{

int j = 0;

foreach (var i in coins)

{

if (i.ShortName == shortName)

{

return j;

}

j++;

}

return -1;

}

catch (Exception e)

{

return -1;

}

}

public async void SellCrypto(int userId, string shortname, double quantityForSale)

{

var ur = new UserRepository();

var cr = new CurrencyRepository();

double quantityInUserWallet = ur.GetCoinQuantityInUserWallet(userId, "usdt");

if (await cr.GetCoinPrice(quantityInUserWallet, "usdt") < await cr.GetCoinPrice(quantityForSale, shortname))

{

return;

}

SubtractCoinFromUser(userId, shortname, quantityForSale);

}

public void SubtractCoinFromUser(int userId, string shortname, double quantityForSubtract)

{

var ur = new UserRepository();

List<Coin> coins = ur.GetUserCoins(userId);

int coinId = GetPurchasedCoinId(coins, shortname);

int coinIdInTheList = GetPurchasedCoinNumberInTheList(coins, shortname);

if (coinId != -1)

{

var coin = new Coin(coins[coinIdInTheList].Id, coins[coinIdInTheList].Quantity, coins[coinIdInTheList].ShortName);

double finalQuantity = coin.Quantity - quantityForSubtract;

if (finalQuantity == 0) {

DeleteCoin(coin.Id);

}else if(finalQuantity > 0) {

UpdateCoinQuantity(coin.Id, finalQuantity);

}

}

}

public async Task<double> GetCoinPrice(double quantity, string shortName)

{

string apiKey = "4da2c4791b9c285b22c1bf08bc36f304ab2ca80bc901504742b9a42a814c4614";

using var httpClient = new HttpClient();

httpClient.DefaultRequestHeaders.Add("X-MBX-APIKEY", apiKey);

string url = $"https://min-api.cryptocompare.com/data/price?fsym=" + shortName + "&tsyms=USD";

var response = await httpClient.GetAsync(url);

var responseContent = await response.Content.ReadAsStringAsync();

Console.WriteLine(responseContent);

JObject json = JObject.Parse(responseContent);

double price = (double)json["USD"] \* quantity;

return price;

}

public async Task<double> GetCoinQuantity(double quantityUSD, string shortName)

{

double price = await GetCoinPrice(1 , shortName);

return quantityUSD / price;

}

public async Task<double> GetUserBalance(int userId)

{

var ur = new UserRepository();

List<Coin> coins = ur.GetUserCoins(userId);

double balance = 0;

foreach (var i in coins)

{

balance += await GetCoinPrice(i.Quantity, i.ShortName);

}

return balance;

}

public void DeleteCoin(int coinId)

{

using var connection = new NpgsqlConnection(connectionString);

var sql = "DELETE FROM coins WHERE id = @id";

using var command = new NpgsqlCommand(sql, connection);

command.Parameters.AddWithValue("@id", coinId);

OpenConnection(connection);

command.ExecuteNonQuery();

CloseConnection(connection);

}

//3 sec

public async Task<double> GetDailyPriceImpact(string shortName)

{

string apiKey = "4da2c4791b9c285b22c1bf08bc36f304ab2ca80bc901504742b9a42a814c4614";

using var httpClient = new HttpClient();

httpClient.DefaultRequestHeaders.Add("X-MBX-APIKEY", apiKey);

string url = $"https://min-api.cryptocompare.com/data/pricemultifull?fsyms=" + shortName + "&tsyms=USD";

var response = await httpClient.GetAsync(url);

var responseContent = await response.Content.ReadAsStringAsync();

var data = JObject.Parse(responseContent);

var priceData = data["RAW"]["BTC"]["USD"];

var priceChange = (double)priceData["CHANGEDAY"];

return priceChange;

}

/\*public async Task<CoinsInformation> GetFullCoinInformation(string shortName) 10 sec

{

var coin = new CoinsInformation();

Dictionary<string, string> cryptoDictionary = CoinList.GetCryptoDictionary();

string fullName = cryptoDictionary[shortName];

string apiKey = "4da2c4791b9c285b22c1bf08bc36f304ab2ca80bc901504742b9a42a814c4614";

using var httpClient = new HttpClient();

httpClient.DefaultRequestHeaders.Add("X-MBX-APIKEY", apiKey);

string url = $"https://min-api.cryptocompare.com/data/pricemultifull?fsyms=" + shortName + "&tsyms=USD";

var response = await httpClient.GetAsync(url);

var responseContent = await response.Content.ReadAsStringAsync();//server response

var data = JObject.Parse(responseContent);

var priceData = data["RAW"][shortName.ToUpper()]["USD"];

var priceChange = (double)priceData["CHANGEDAY"];

var dailyVolume = (double)priceData["VOLUME24HOUR"];

var price = (double)priceData["PRICE"];

return new CoinsInformation(fullName, shortName, @"C:\НЕ СИСТЕМА\BSUIR\второй курс\UP\cryptoicons\_png\128\" + shortName.ToUpper(), dailyVolume, priceChange, price);

}\*/

private static readonly HttpClient httpClient = new HttpClient();

private const string ApiKey = "4da2c4791b9c285b22c1bf08bc36f304ab2ca80bc901504742b9a42a814c4614";

private const string CryptoCompareApiUrl = "https://min-api.cryptocompare.com";

public async Task<CoinsInformation> GetFullCoinInformation(string shortName)

{

Dictionary<string, string> cryptoDictionary = CoinList.GetCryptoDictionary();

string fullName = cryptoDictionary[shortName];

string url = $"{CryptoCompareApiUrl}/data/pricemultifull?fsyms={shortName}&tsyms=USD";

using var request = new HttpRequestMessage(HttpMethod.Get, url);

request.Headers.Add("X-MBX-APIKEY", ApiKey);

using var response = await httpClient.SendAsync(request, HttpCompletionOption.ResponseHeadersRead);

if (!response.IsSuccessStatusCode)

{

throw new HttpRequestException($"Failed to get coin information: {response.StatusCode}");

}

var responseContent = await response.Content.ReadAsStringAsync();

var data = JObject.Parse(responseContent);

var priceData = data["RAW"][shortName.ToUpper()]["USD"];

var priceChange = (double)priceData["CHANGEDAY"];

var dailyVolume = (double)priceData["VOLUME24HOUR"];

double number = dailyVolume;

var price = (double)priceData["PRICE"];

var previousPrice = price - priceChange;

var percentagePriceChangePerDay = (priceChange / previousPrice) \* 100;

return new CoinsInformation(fullName, shortName, @"C:\НЕ СИСТЕМА\BSUIR\второй курс\UP\cryptoicons\_png\128\" + shortName.ToLower(), dailyVolume, priceChange, price, percentagePriceChangePerDay);

}

public void UpdateCoinQuantity(int id, double quantity)

{

using var connection = new NpgsqlConnection(connectionString);

var sql = "UPDATE coins SET quantity = @quantity WHERE id = @id";

using var command = new NpgsqlCommand(sql, connection);

command.Parameters.AddWithValue("@id", id);

command.Parameters.AddWithValue("@quantity", quantity);

OpenConnection(connection);

command.ExecuteNonQuery();

CloseConnection(connection);

}

public void WriteTransactionToDatabase(string coinName, double quantity, int senderId, int receiverId)

{

using var connection = new NpgsqlConnection(connectionString);

var sql = "INSERT INTO transactions (coin\_name, quantity, date, sender\_id, receiver\_id) VALUES " +

"(@coin\_name, @quantity, @date, @sender\_id, @receiver\_id)";

using var command = new NpgsqlCommand(sql, connection);

command.Parameters.AddWithValue("@coin\_name", coinName);

command.Parameters.AddWithValue("@quantity", quantity);

command.Parameters.AddWithValue("@date", DateTime.Now);

command.Parameters.AddWithValue("@sender\_id", senderId);

command.Parameters.AddWithValue("@receiver\_id", receiverId);

OpenConnection(connection);

command.ExecuteNonQuery();

CloseConnection(connection);

}

}

}

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг скрипта генерации базы данных

create table users

(

id serial

primary key,

login varchar(255) not null,

password varchar(255) not null,

email varchar(255) not null,

creation\_date timestamp not null,

modification\_date timestamp not null,

is\_deleted boolean not null,

role\_id integer not null,

is\_blocked boolean not null,

salt varchar not null

);

alter table users

owner to postgres;

create table withdrawals

(

id bigint generated always as identity (minvalue 0 cycle)

primary key,

date date not null,

quantity double precision not null,

commission double precision not null,

user\_id bigint not null

constraint withdrawals\_userid\_fkey

references users

on update cascade on delete cascade

);

alter table withdrawals

owner to postgres;

create table conversions

(

id bigint generated always as identity (minvalue 0 cycle)

constraint table\_name\_pkey

primary key,

commission double precision not null,

begin\_coin\_quantity double precision not null,

end\_coin\_quantity double precision not null,

quantity\_usd double precision not null,

begin\_coin\_shortname varchar not null,

end\_coin\_shortname varchar not null,

user\_id bigint not null

constraint table\_name\_user\_id\_fkey

references users

on update cascade on delete cascade,

date timestamp not null

);

alter table conversions

owner to postgres;

create table login\_history

(

id bigint generated always as identity

primary key,

ip varchar(255) not null,

date date not null,

user\_id bigint not null

references users

on update cascade on delete cascade

);

alter table login\_history

owner to postgres;

create table replenishment

(

id bigint generated always as identity (minvalue 0 cycle)

primary key,

date timestamp not null,

quantity double precision not null,

commission double precision not null,

user\_id bigint not null

references users

on update cascade on delete cascade

);

alter table replenishment

owner to postgres;

create table transactions

(

id bigint generated always as identity (minvalue 0)

primary key,

coin\_name varchar(255) not null,

quantity double precision not null,

date timestamp not null,

sender\_id bigint not null

references users

on update cascade on delete cascade,

receiver\_id bigint not null

constraint transactions\_reciever\_id\_fkey

references users

on update cascade on delete cascade

);

alter table transactions

owner to postgres;

create table previos\_passwords

(

id bigint generated always as identity (minvalue 0 cycle)

primary key,

user\_id bigint not null

constraint previos\_passwords\_users\_id\_fk

references users

on update cascade on delete cascade,

password varchar(25) not null

);

alter table previos\_passwords

owner to postgres;

create table blocking

(

id bigint generated always as identity

primary key,

cause varchar(25) not null,

user\_id bigint not null

constraint blocking\_users\_id\_fk

references users

);

alter table blocking

owner to postgres;

create table coins\_info

(

id integer not null

constraint coins\_info\_pk

primary key,

fullname varchar not null,

shortname varchar not null,

iconpath varchar not null,

daily\_volume double precision not null,

daily\_impact double precision not null

);

alter table coins\_info

owner to postgres;

create table coins

(

id bigint generated always as identity (minvalue 0 maxvalue 1000000 cycle)

constraint coins\_pk

primary key,

quantity double precision not null,

shortname varchar not null

);

alter table coins

owner to postgres;

create table l\_users\_coins

(

id bigint generated always as identity (maxvalue 100000000 cycle)

constraint l\_users\_coins\_pk

primary key,

user\_id bigint not null

constraint l\_users\_coins\_users\_id\_fk

references users

on update cascade on delete cascade,

coin\_id bigint not null

constraint l\_coins\_users\_id\_fk

references coins

on update cascade on delete cascade

);

alter table l\_users\_coins

owner to postgres;

create table cards

(

id bigint generated always as identity (minvalue 0 cycle)

constraint cards\_pk

primary key,

number varchar not null,

security\_code integer not null,

validity varchar not null,

name varchar not null,

secnd\_name varchar not null,

user\_id bigint not null

constraint cards\_users\_id\_fk

references users

on update cascade on delete cascade

);

alter table cards

owner to postgres;