Изображение выглядит как графическая вставка, Графика, мультфильм

Автоматически созданное описание

**Зам. директора КМПО**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.Ф. Гасанов**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.**

**ДИПЛОМНЫЙ ПРОЕКТ**

**ТЕМА: Автоматизированная система управления криптоактивами на примере ООО «Ван АйтиКонсалтинг»**

**Специальность 09.02.07 Информационные системы и программировани**

|  |  |  |
| --- | --- | --- |
| **Выполнил студент(ка) группы 41ис-20** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **А.Ш. Алиев** |
| **Руководитель** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Н.А. Бабаева** |
| **Старший консультант** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **О.А. Калашникова** |
| **Консультант по технико-экономическому обоснованию работы** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **М.М. Трифонова** |
| **Нормоконтролер** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Н.А. Бабаева** |

**Москва 2024**

# **РЕФЕРАТ**

Криптовалюта становится все более значимым видом активов, привлекая внимание как индивидуальных инвесторов, так и крупных финансовых институтов. С ростом интереса к криптовалютам возникает потребность в эффективном управлении их портфелями, мониторинге и анализе транзакций, а также в обеспечении безопасности и надежности операций с криптоактивами.

В данной дипломном проекте исследуется проблема управления современными цифровыми активами и предлагается разработка автоматизированной системы управления криптоактивами. Эта система будет предназначена для управления портфелями криптовалют, надежного хранения информации и осуществления анализа рынка.

В проекте рассматриваются основные принципы управления криптоактивами, а также анализируются существующие подходы к автоматизации процессов управления инвестициями в цифровые сбережения. Для разработки системы управления виртуальными финансами используются современные методы и технологии, такие как машинное обучение, анализ больших данных и блокчейн.

Дипломный проект состоит из введения, где определяется актуальность цели и задач выбранной темы; основной части, в которой описывается методология разработки и реализации автоматизированной системы управления криптоактивами; заключения, в котором делаются выводы по результатам работы и формулируются перспективы дальнейших исследований.

СОДЕРЖАНЕ

[РЕФЕРАТ 2](#_Toc168580458)

[Введение 4](#_Toc168580459)

[1 Описание предметной области 5](#_Toc168580460)

[1.1 Анализ существующих криптовалютных сайтов и приложений 5](#_Toc168580461)

[1.2 Описание основных функциональных особенностей автоматизированной системы управления криптоактивной системой 10](#_Toc168580462)

[2 АРХИТЕКТУРА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ 12](#_Toc168580463)

[2.1 Концептуализация проектных предложений 12](#_Toc168580464)

[2.1.1 Схема создания прецедентов 13](#_Toc168580465)

[2.2 Проектирование базы данных 18](#_Toc168580466)

[2.3 Архитектура и ключевые элементы автоматизированной криптоактивной системы управления 20](#_Toc168580467)

[3 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ 25](#_Toc168580468)

[3.1 Настройка серверной среды 25](#_Toc168580469)

[3.3 Основы написания логики 29](#_Toc168580470)

[3.4 Написание кода интерфейса программы 34](#_Toc168580471)

[4 ТЕСТИРОВАНИЕ И ОТЛАДКА 38](#_Toc168580472)

[4.1 Функциональное тестирование автоматизированной системы 38](#_Toc168580473)

[5. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ 41](#_Toc168580474)

[Заключение 46](#_Toc168580475)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc168580476)

# **Введение**

В эпоху цифровизации, финансовые рынки неуклонно трансформируются под влиянием цифровых технологий; значимость последних в различных сферах человеческой деятельности постоянно возрастает. В частности, последние годы ознаменовались заметным явлением — появлением и распространением криптовалют и криптоактивов (таких как Bitcoin, Ethereum и прочие), вызвавших значительный интерес у инвесторов, специалистов финансового сектора и общества в целом. Эти новые формы денег и активов вызывают волну интереса благодаря своей инновационной природе.

С появлением криптовалют возникла неотложная потребность в эффективном управлении такими активами. Инвесторам предстоит не только формировать и управлять собственными крипто-портфелями, но и неустанно отслеживать рыночную конъюнктуру, проводить анализ трендов и осуществлять обоснованные инвестиционные решения. В то же время, важность разработки надёжных инструментов для управления криптоактивами подчеркивается высокой волатильностью и риском, характерными для данной сферы.

Проект, описываемый в данном документе, нацелен на создание системы автоматического управления криптоактивами, предназначенной для того, чтобы помочь инвесторам более эффективно контролировать свои портфели криптовалют и осуществлять осознанные инвестиционные решения. Эта система призвана стать мостом между сложностью криптовалютных активов и потребностями пользователей в простоте и безопасности управления.

# **1 Описание предметной области**

## 1.1 Анализ существующих криптовалютных сайтов и приложений

В современном контексте цифровых транзакций осуществляется глубокий анализ множества платформ, предназначенных для операций с криптовалютой, чья дифференциация обусловлена предоставлением уникальных услуг, нацеленных на инвесторов и трейдеров. Осуществим детальное рассмотрение избранных ресурсов, что позволит выявить структуру их ключевых особенностей и характеристик, предоставляя, таким образом, возможность для более основательного понимания текущих динамик и тенденций данного сектора.

В контексте обширной экосистемы цифровых платформ, особого внимания заслуживает Binance — всемирно признанная площадка, осуществляющая множество услуг в области криптоторговли и инвестирования в цифровые активы. Несмотря на сравнительно молодой возраст, начиная с 2017 года, данная платформа утвердилась в качестве лидера, предоставив инвесторам обширный арсенал торговых пар, многогранность функциональных возможностей и многоуровневую систему защиты.

Детализируя функциональные аспекты и уникальные характеристики Binance, можно выделить следующие пункты:

— Торговый сектор: площадка активизирует проведение операций с различными криптовалютами — от Bitcoin до Ripple, обеспечивая тем самым широкие возможности для транзакционной деятельности.

— Маржинальная торговля: предусмотрена опция использования кредитных ресурсов для максимизации потенциальной выгоды.

— Фьючерсы и опционы: домен, открытый для профессионалов, предлагает инструменты для спекуляций на изменениях стоимости криптовалют.

— Продукция в области стейкинга и децентрализованных финансов (DeFi): интеграция функций для привлечения средств в стейкинг и DeFi-проекты.

— Ликвидность и объемы торгов: благодаря высоким показателям, Binance обеспечивает оперативность и выгодные условия для проведения транзакций.

— Мобильная версия: платформа предлагает удобное управление активами и торговлю через мобильное приложение, доступное в любой точке мира.

— Защита средств: сложная, многоуровневая система обеспечивает высочайшую безопасность с использованием двухфакторной аутентификации и надежного хранения.

— Образовательные ресурсы: разнообразие обучающих материалов способствует улучшению торговых умений и навыков.

Таким образом, благодаря стратегии интегрированного подхода, развитого функционала и безукоризненной защиты, Binance неукоснительно подтверждает свою роль ведущего актора на арене криптовалют.

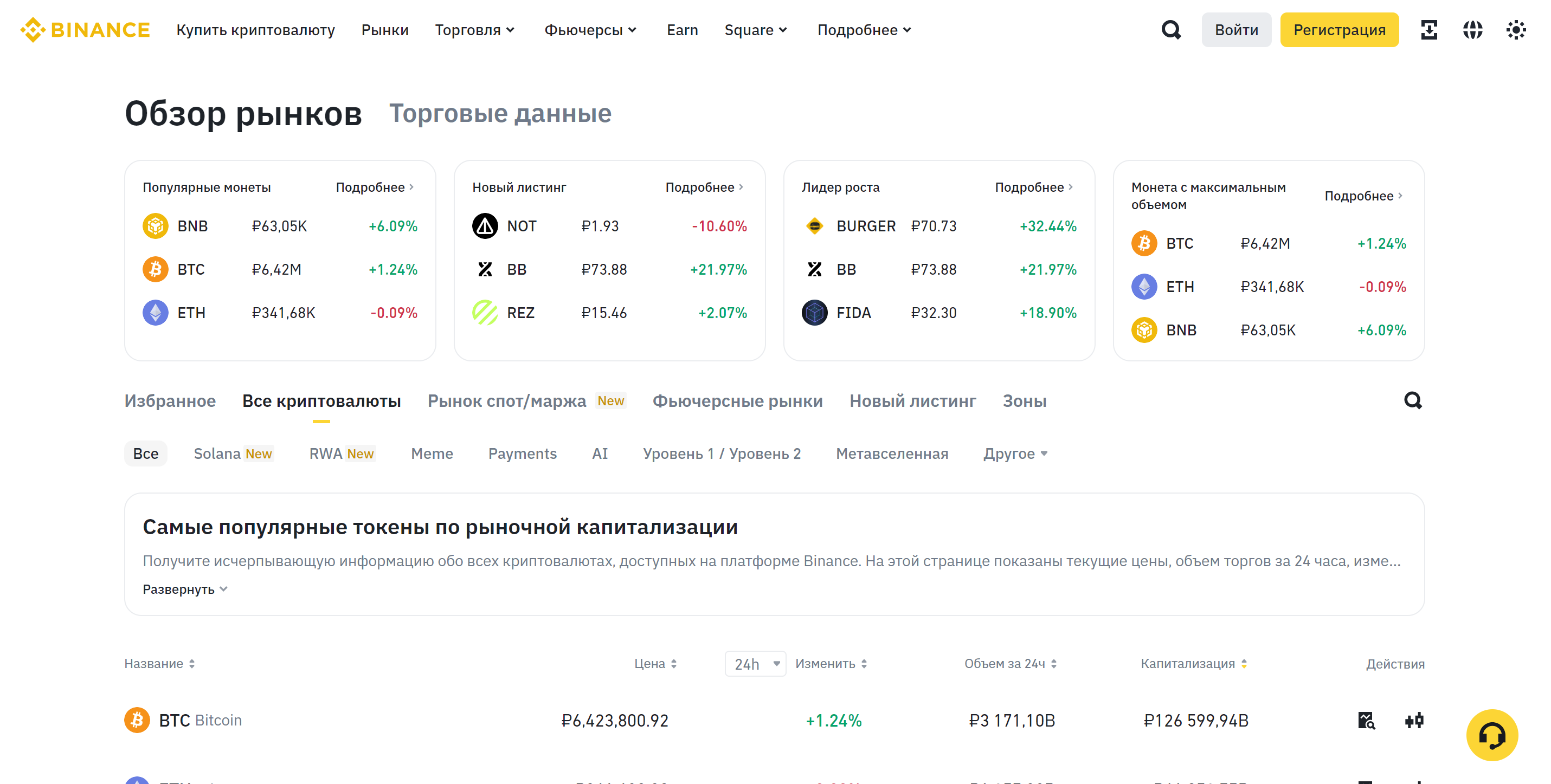


Рисунок 1 Интерфейс Binance.

Инициированная в 2018 году, платформа ByBit скоро превратилась в одну из ключевых бирж, которая привлекает внимание трейдеров, преимущественно ориентированных на маржинальную торговлю фьючерсами криптовалют. Уникальные черты данной платформы, включая интуитивно понятный интерфейс и разнообразие сервисов, способствуют укреплению её позиций в данной сфере.

Ключевые характеристики и функциональные возможности ByBit можно описать следующим образом:

— маржинальная торговля: ByBit предлагает использование маржинальной торговли, значительно повышающей потенциал доходности и диверсификацию портфеля инвестора через фьючерсные контракты на криптовалюты;

— разнообразие торговых пар: обширный спектр доступных торговых пар, включающий ведущие криптовалюты, такие как Bitcoin, Ethereum и EOS, открывает широкие возможности для трейдинга;

— передовые торговые инструменты: комплектация ByBit современными инструментами предназначена для усовершенствования торгового процесса и оптимизации стратегий трейдеров;

— высокая производительность и надежность: благодаря мощной технологической инфраструктуре, платформа обеспечивает стабильность работы, что является критически важным в условиях динамично меняющегося рынка;

— защита активов: ByBit ставит высокий приоритет на защиту активов своих клиентов, используя передовые методы шифрования и многоуровневую систему безопасности;

— образовательные ресурсы: наличие обширного массива обучающих материалов, включая блоги, видеоуроки и руководства, помогает трейдерам повысить их профессиональные умения и понимание тонкостей криптовалютной торговли;

— мобильное приложение: возможность управления торговлей через мобильное приложение ByBit, предоставляющее трейдерам доступ к их операциям в любом месте, что поддерживает высокую активность даже в условиях мобильности.

Таким образом, благодаря этим функциям и инновациям, ByBit продолжает занимать лидирующие позиции среди платформ маржинальной торговли, обеспечивая своим пользователям высококачественный сервис и надежность.

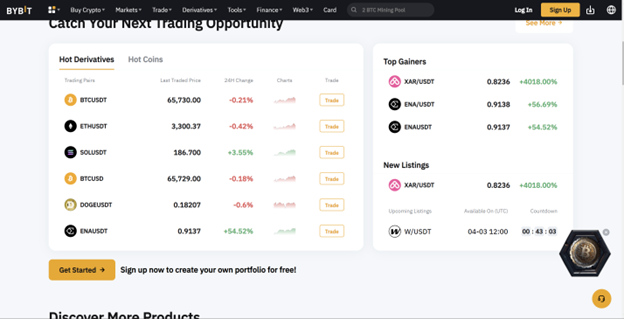


Рисунок 2 – ByBit

В контексте цифровой трансформации финансовых активов, инструмент управления портфелем криптовалют Delta, запущенный в 2017 году, предоставляет трейдерам и инвесторам усовершенствованные возможности для анализа и мониторинга инвестиций в цифровые активы. Эта платформа быстро завоевала популярность благодаря своей простоте использования, многообразию функциональных возможностей и элегантному дизайну.

Детализируем ключевые особенности Delta следующим образом:

1. Мониторинг портфеля: Данный инструмент предоставляет пользователям возможность интегрировать криптоактивы в их портфель и осуществлять их непрерывное отслеживание в реальном времени, что обеспечивает оперативное обновление информации о текущих ценах и динамике их изменений.

2. Интеграция с криптовалютными биржами: Пользователи могут связывать свои учетные записи с различными торговыми платформами, включая такие как Binance, Coinbase и Kraken, что способствует автоматизации процесса отслеживания балансов и транзакций.

3. Аналитические инструменты: Приложение оснащено функциями, предоставляющими детализированные графики цен и различные аналитические инструменты, которые способствуют принятию обоснованных инвестиционных решений.

4. Персонализированные уведомления: Delta предоставляет возможности настройки уведомлений о значительных изменениях цен, объемах торгов и других ключевых рыночных параметрах, помогая пользователям оставаться в курсе важных рыночных событий.

5. Функции импорта и экспорта данных: Приложение облегчает процесс передачи данных о транзакциях и портфеле между различными устройствами и платформами, что способствует удобству и гибкости использования.

6. Безопасность и конфиденциальность данных: Delta обеспечивает высокий уровень защиты пользовательских данных и активов посредством применения передовых технологий шифрования и многоуровневых систем безопасности.

7. Многоязычность и доступность: Приложение поддерживает множество языков и доступно для загрузки на различных платформах, включая iOS, Android и веб-интерфейс, что делает его доступным для широкой аудитории.

Благодаря интуитивно понятному интерфейсу и всестороннему набору функций, Delta продолжает утверждаться как предпочтительное решение для трейдеров и инвесторов, стремящихся к эффективному управлению своими криптовалютными портфелями.



Рисунок 3 – Интерфейс Delta

\*\*В контексте цифровой трансформации финансового сектора, мобильное приложение CoinStats, специализирующееся на мониторинге портфеля криптовалют и анализе рыночных тенденций, предлагает пользователям комплексные инструменты для наблюдения за инвестициями в цифровые активы.\*\* Основная задача данного приложения состоит в упрощении процесса управления криптоактивами, обеспечивая пользователям возможности для отслеживания текущих цен, анализа динамики рыночных изменений и получения значимых уведомлений о событиях, влияющих на портфель пользователя.

Демонстрация ключевых особенностей и функций CoinStats выявляет следующие аспекты:

1. Мониторинг портфеля: CoinStats обладает функциональностью, которая позволяет пользователям добавлять информацию о криптовалютных активах в портфель и проводить их систематическое отслеживание, отображая при этом текущие значения, балансы и исторические данные транзакций.

2. Интеграция с криптовалютными биржами: приложение поддерживает связь с различными торговыми платформами, способствуя автоматизации процесса мониторинга балансов и транзакций на всех ассоциированных аккаунтах.

3. Аналитические инструменты и графики: CoinStats предоставляет доступ к диаграммам цен и аналитическим инструментам, включая объемы торгов и технические индикаторы, которые служат для тщательного анализа рыночных тенденций.

4. Оповещения и уведомления: приложение предоставляет возможности для настройки оповещений о ключевых рыночных событиях, таких как изменения цен, объемы торгов и новости, что позволяет пользователям оставаться информированными о важных рыночных изменениях.

5. Интеграция кошелька: функциональность приложения включает в себя возможность прямого мониторинга балансов и транзакций через ассоциированные криптовалютные кошельки.

6. Образовательные ресурсы: CoinStats предлагает пользователям широкий спектр образовательных материалов, аналитических отчетов и исследований в области криптовалют и блокчейна, что способствует обоснованному принятию инвестиционных решений.

Благодаря своей многофункциональности, которая способствует эффективному управлению портфелями криптовалют и глубокому пониманию рыночной динамики, CoinStats продолжает удерживать свои позиции среди предпочтений трейдеров и инвесторов.

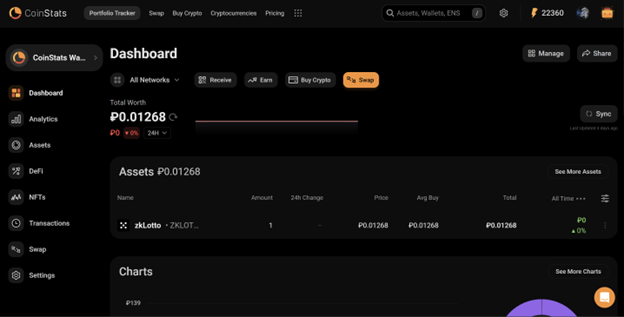


Рисунок 5 – Интерфейс CoinStats

## 1.2 Описание основных функциональных особенностей автоматизированной системы управления криптоактивной системой

В контексте непрерывного развития сферы криптовалют, анализ рыночных предложений выявляет набор критериев, которые необходимо интегрировать в новую автоматизированную систему управления криптоактивами. Предполагается, что данная система будет представлять собой комплексный инструмент, способный обеспечить всесторонний контроль и значительное повышение эффективности управления цифровыми портфелями.

Детализация ключевых функциональных возможностей системы представлена следующим образом:

1. профессиональное управление: Система должна включать возможности для добавления, удаления и редактирования криптовалютных активов в портфеле, а также обеспечивать отслеживание текущих балансов и стоимости активов, включая автоматизированное обновление данных о курсах и ценах.

2. Мониторинг рынка: необходимо обеспечить доступ к актуальной информации о ценах, объемах торговли и другим ключевым параметрам рынка, включая визуализацию данных в виде графиков и диаграмм для анализа динамики рынка.

3. Анализ данных: Система должна предоставлять функционал для технического и фундаментального анализа рынков, используя разнообразные инструменты и индикаторы для прогнозирования рыночных тенденций.

4. Управление торговлей: необходимость в возможности прямой торговли на крупных криптовалютных биржах, включая реализацию ордеров на покупку и продажу по установленным критериям

5. Прогнозирование и оптимизация портфеля: система должна предоставлять инструменты для прогнозирования будущих цен и доходности криптовалют, а также для оптимизации структуры портфеля на основе этих прогнозов.

6. Безопасность и конфиденциальность: обязательное условие — высокий уровень защиты персональных данных и активов пользователей, с использованием передовых технологий шифрования и многоуровневой аутентификации.

7. Отчетность и анализ: система должна предоставлять отчеты о состоянии портфеля, завершенных бизнес-операциях и эффективности инвестиционных стратегий, а также аналитические данные для оценки результатов

Эти функциональные возможности, интегрированные в новую систему, обеспечат пользователям надежные и современные инструменты для эффективного управления криптоактивами, позволяя им адаптироваться к динамике рынка и принимать обоснованные инвестиционные решения.

В условиях растущей популярности криптовалют как значимого инвестиционного актива, создание эффективного инструмента для управления этими активами является крайне важной задачей. Основные этапы разработки включают:

1. Анализ потребностей пользователей: необходимо определить ключевые потребности и требования пользователей посредством детального анализа рынка криптовалют.

2. Проектирование функционала: следует разработать спецификации и интерфейсы для каждой функции, опираясь на результаты анализа потребностей.

3. Выбор технологического стека: необходимо определить наиболее подходящие технологии и инструменты, которые смогут обеспечить масштабируемость, производительность и безопасность системы.

4. Разработка и тестирование: провести функциональное программирование системы, разработку технического задания и реализацию необходимых модификаций.

5. Проектирование и поддержка: организовать ввод системы в эксплуатацию, обеспечить её стабильную работу и техническую поддержку, а также регулярно обновлять и улучшать функционал в соответствии с потребностями пользователей.

6. Обеспечение защиты и конфиденциальности данных: включающее шифрование данных и многоуровневую аутентификацию.

Основная задача разработки состоит в создании надёжной и эффективной системы, которая обеспечит пользователям удобство и безопасность в управлении инвестициями в криптовалюту, тем самым позволяя принимать обоснованные инвестиционные решения.

# **2 АРХИТЕКТУРА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ**

## 2.1 Концептуализация проектных предложений

Концептуализация проекта представляет собой критический этап в процессе архитектурного проектирования автоматизированной системы управления криптографическими активами. На данной стадии происходит создание диаграммы прецедентов и подбор необходимых инструментальных ресурсов для последующей реализации проекта. Эти действия являются необходимыми для точного определения функциональных требований системы и выбора наиболее подходящих технологических решений, которые обеспечат их наиболее эффективное выполнение.

2.1.1 Схема создания прецедентов

Диаграмма вариантов использования является неотъемлемым инструментом в процессе проектирования программного обеспечения. Она применяется для наглядного представления функциональных требований системы, улучшая осведомленность разработчиков и всех заинтересованных сторон о механизмах функционирования системы, а также о задачах, которые система должна выполнять. Примером может служить ООО «ВанИТКонсалтинг», в рамках которого в этом разделе изложены цели и задачи создания диаграммы вариантов использования, процесс её разработки и применения в контексте архитектуры системы.

Основные задачи разработки диаграммы вариантов использования включают в себя:

1. определение функциональных требований: диаграмма вариантов использования способствует точному определению необходимых функций системы и сложных сценариев её использования как конечными пользователями, так и администрацией;

2. Визуализация: диаграмма предоставляет графическое изображение взаимодействий между пользователями (субъектами) и системой, способствуя более глубокому пониманию процессов взаимодействия различных пользователей с системой и возможностей, которые они могут реализовать с помощью платформы управления виртуальными активами;

3. Упрощение коммуникации: диаграмма служит универсальным языком между разработчиками, заказчиками и другими заинтересованными сторонами, что способствует упрощению обсуждения и достижению консенсуса относительно функциональных требований;

4. Идентификация ключевых функций: диаграмма позволяет выявить и систематизировать ключевые функции системы и их взаимосвязи, что способствует более основательному проектированию и последующей реализации системы.

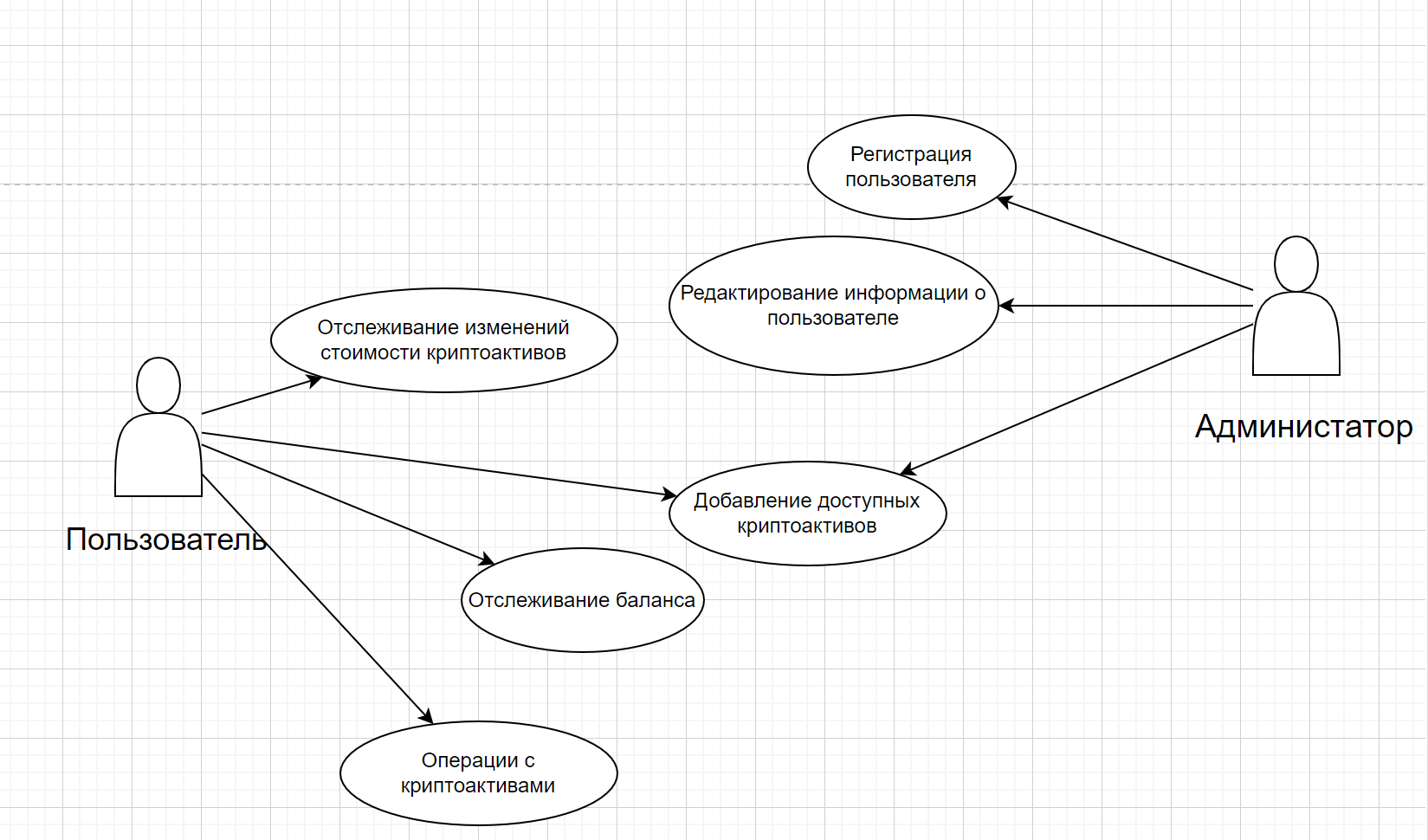


Рисунок 6 – Диаграмма прецедентов автоматизированной системы

ООО «ВайИТКонсалтинг»

Сценарий, обозначенный термином «Регистрация пользователя», охватывает роль системного администратора и сотрудника в процессе, который включает регистрацию новоиспечённого сотрудника в системе. Администратор вносит персональные сведения и данные для логина, а также инициирует создание начального пароля. Сотрудник, обладая начальной информацией для доступа, имеет право на модификацию пароля; при этом, система архивирует данные о таких изменениях. Кульминационным моментом процесса является успешное завершение регистрации сотрудника, обеспечивающее ему доступ к личному кабинету.

В рамках второго сценария, названного «Редактирование информации о пользователе», сотруднику предоставляется возможность адаптации своих личных и аккаунтных данных; системный администратор, в свою очередь, обретает доступ к актуализированным данным. Так, информация о пользователе подвергается обновлению, при этом система продолжает отслеживание истории изменений парольной информации.

В третьем сценарии, озаглавленном «Добавление доступных криптоактивов», системному администратору предоставляются функции по добавлению, модификации и исключению типов криптоактивов и криптовалют, доступных для управления. Сотрудник, со своей стороны, имеет возможность осуществлять просмотр доступных активов и выполнять с ними операции, включая покупку и продажу, что в конечном итоге способствует расширению системы за счёт интеграции новых видов активов и криптовалют.

В четвёртом сценарии, именуемом «Транзакции с криптоактивами», предусмотрена возможность для сотрудника осуществлять операции с доступными криптоактивами, в том числе проведение покупок и продаж. Результатом таких действий становится трансформация баланса в кошельке сотрудника.

Пятый сценарий, обрисовывающий «Отслеживание баланса», дает возможность сотруднику вести наблюдение за состоянием своего кошелька в удобной форме и следить за изменениями на своих счетах, что способствует поддержанию актуальности отображаемых данных о балансе.

Шестой сценарий, озаглавленный «Мониторинг изменения стоимости криптоактива», предоставляет сотруднику инструментарий для визуализации динамики стоимости определенного криптоактива за выбранный период в форме графика, что позволяет в конечном итоге формировать представление о тенденциях изменения стоимости актива.

В различных фазах проектирования и разработки системы, диаграмма вариантов использования выполняет множество функций: она облегчает сбор и анализ информации о функционале будущей системы на этапе анализа требований, становится основой для определения необходимых модулей и компонентов на этапе проектирования архитектуры, предоставляет разработчикам руководство по реализации функциональности и созданию тестовых сценариев на этапе разработки и тестирования, а также служит неоценимым ресурсом для обучения новых сотрудников основам работы системы на этапе документирования и обучения.

Для реализации системы были выбраны такие инструменты, как Python, Qt Designer и MySQL; выбор этих инструментов, по сравнению с такими альтернативами, как ванильный JavaScript с HTML/CSS, Bootstrap и SQLite, подтверждается всесторонним анализом их функциональных возможностей и ограничений.

Таблица 1 – Выбор языка программирования и фреймворка

| Критерий | Python + Qt Designer | Vanilla JavaScript + HTML/CSS |
| --- | --- | --- |
| |  | | --- | | Облегчение процесса обучения |  |  | | --- | |  | | Исключительно высокое (благодаря интуитивно понятному синтаксису Python и визуальным инструментам Qt Designer) | Относительно низкое (сложность и разнообразие синтаксических структур) |
| Темпы разработки | |  | | --- | | Превосходная скорость (обусловленная лаконичностью кода и развитой библиотечной системой) |  |  | | --- | |  | | |  | | --- | | Сниженные (за счет необходимости ручного управления DOM и отсутствия стандартизированных библиотек) |  |  | | --- | |  | |
| Поддержка асинхронных операций | |  | | --- | | Достаточно эффективная (реализация через asyncio и concurrent.futures) |  |  | | --- | |  | | |  | | --- | | Ограниченная (требует ручного контроля асинхронности через коллбэки) |  |  | | --- | |  | |
| Экосистема и сообщество | |  | | --- | | Масштабное сообщество, обширное количество библиотек |  |  | | --- | |  | | |  |  |  | | --- | --- | --- | | |  | | --- | | Умеренное (с ограниченным набором библиотек, отсутствием стандартизации) |  |  | | --- | |  | |  |  | | --- | |  | |
| Поддержка интерфейсов | |  | | --- | | Высочайший уровень поддержки через PyQt и Qt Designer |  |  | | --- | |  | | |  | | --- | | Низкая (требует ручного контроля над стилями и структурами) |  |  | | --- | |  | |

Таблица 2 - Система управления базами данных

| Критерий | MySQL | SQLlite |
| --- | --- | --- |
| |  | | --- | | Производительность |  |  | | --- | |  | | Весьма высокая (особенно ценится в многопользовательских системах и системах с большим объемом данных) | Относительно низкая (оптимальна для маломасштабных и встроенных приложений) |
| |  | | --- | | Поддержка сложных запросов |  |  | | --- | |  | | Превосходная (с возможностью оптимизации запросов и использования индексации) | Ограниченная (не предназначена для выполнения сложных аналитических запросов) |
| |  | | --- | | Сообщество и документация |  |  | | --- | |  | | Обширное сообщество, выдающаяся документация | Обширное сообщество, выдающаяся документация |
| |  | | --- | | Поддержка ACID |  |  | | --- | |  | | Интегральная поддержка ACID транзакций | Интегральная поддержка ACID транзакций, однако с ограниченной эффективностью в контексте многопользовательских операций |
| |  | | --- | | Масштабируемость |  |  | | --- | |  | | Высокая (поддержка кластеризации, репликации и шардинга) | Существенно ограниченная (преимущественно используется) |

Выбор инструментов таких как Python и Qt Designer для разработки серверной и клиентской частей архитектурного комплекса, а также применение MySQL для администрирования баз данных, оказался продиктован множеством критически важных факторов:

- Python, благодаря своей выдающейся читаемости кода, простоте в освоении, а также за счет обширной библиотеки, выступает в роли идеального средства для быстрого создания прототипов и реализации высоконадежных системных решений.

- Qt Designer является инструментом, предоставляющим расширенные функциональные возможности для конструирования сложных пользовательских интерфейсов — аспекта, который приобретает особую значимость в условиях стационарного использования приложений.

- MySQL проявляет высокую производительность вместе с простотой интеграции, что ставит его в ряд лидирующих решений для широкого спектра приложений, особенно тех, которым не требуется выполнение обширных аналитических запросов.

При сопоставлении с альтернативными вариантами, такими как применение JavaScript в сочетании с HTML/CSS для клиентской части и SQLite для управления базами данных, выбранные технологии предоставляют оптимальное сочетание удобства в разработке, высокой производительности и функциональности. Эти аспекты делают их предпочтительным выбором для реализации замыслов данного проекта, обосновывая их преимущество перед конкурентами.

## 2.2 Проектирование базы данных

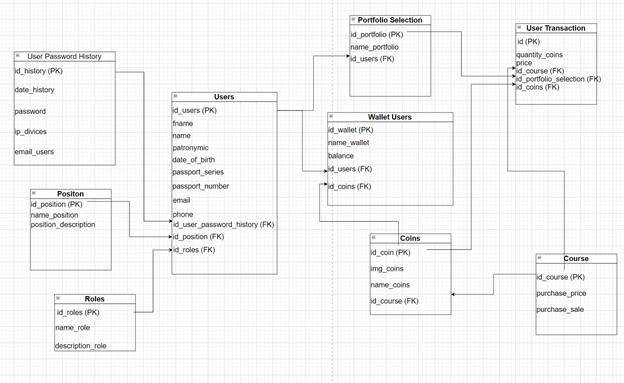
Период проектирования базы данных занимает центральное место в процессе создания автоматизированной системы управления криптоактивами. В эту критическую фазу определяются структура данных, методики их архивации, организации и взаимодействия. Основная задача, стоящая перед проектированием базы данных, состоит в разработке эффективной, надежной и масштабируемой системы для архивации и обработки информации, связанной с криптовалютными активами.

Рисунок 7 – ER диаграмма базы данных автоматизированной системы управления криптоактивами ООО «Ван АйтиКонсалтинг»

Фундаментальные принципы, заложенные в основу процесса проектирования базы данных для управления криптоактивами, включают следующие ключевые аспекты:

- Оптимизация процесса хранения данных является критически важной: необходимо гарантировать, что архитектура базы данных реализует наиболее эффективные методы для архивации информации, что включает в себя данные о пользователях, криптоактивах, транзакциях и других актуальных объектах.

- Обеспечение целостности данных критично: важно поддерживать точность и консистентность данных в базе, в том числе обеспечение ссылочной целостности — например, корреляция между пользователями и их кошельками, а также соблюдение всех установленных ограничений для данных.

- Гарантирование быстрого и надежного доступа к данным\*\* необходимо: база данных должна быть спроектирована так, чтобы обеспечивать пользователям мгновенный доступ к требуемой информации для различных операций, включая просмотр балансов, осуществление транзакций и доступ к актуальным рыночным данным.

- Масштабируемость базы данных обязательна: она должна адаптироваться к растущему числу пользователей и увеличению объема данных, сохраняя при этом стабильность системы при возрастающей нагрузке.

- Безопасность необходима: конфиденциальная информация, включая личные данные пользователей и финансовые сведения, должна быть защищена от несанкционированного доступа и утечек данных.

- Поддержка резервного копирования и восстановления данных является обязательной: наличие надежных систем резервного копирования и восстановления данных критически важно для предотвращения потери данных в случае технических сбоев или системных катастроф.

Преимущества качественно спроектированной базы данных включают:

- Улучшенная производительность: оптимизированные структуры данных и индексы способствуют более быстрому доступу к часто запрашиваемой информации.

-Высокая надежность: стабильная работа и надежные механизмы резервного копирования минимизируют риски потери данных.

- Простота использования: эффективно спроектированная база данных упрощает задачи администрирования, включая добавление новых функций, обновление схем данных и управление доступом.

- Гибкость и адаптируемость: база данных должна быть способна к легкой модификации и расширению в соответствии с изменяющимися требованиями и условиями эксплуатации.

- Соответствие требованиям безопасности: реализация защитных механизмов данных, включая шифрование и контроль доступа, обеспечивает защиту конфиденциальной информации.

Значимость проектирования базы данных для системы управления криптоактивами неоспорима, поскольку она представляет собой фундамент для успешной реализации и долгосрочной стабильности проекта.

Эффективное и безопасное хранение данных, подкрепленное тщательно спланированной структурой данных, а также применением принципов нормализации и определением адекватных связей между таблицами, является залогом надежной и продуктивной работы системы на протяжении всего срока ее эксплуатации. Внедрение необходимых механизмов также играет критическую роль в обеспечении стабильности и функциональности системы.

## 2.3 Архитектура и ключевые элементы автоматизированной криптоактивной системы управления

Архитектура автоматизированной системы управления криптографическими активами создаёт основу для эффективной работы и координации между ключевыми компонентами системы, что является критически важным для достижения проектных целей и обеспечения функциональности. Описание основных компонентов системы и их взаимодействий представлено далее.

Основные компоненты системы состоят из:

1. Пользовательский интерфейс (UI): этот модуль, с которым взаимодействуют пользователи, предоставляет инструменты для просмотра информации о криптоактивах, выполнения транзакций, настройки оповещений и проведения других операций.

2. Бизнес-логика: этот компонент отвечает за обработку и управление ключевыми бизнес-процессами системы, включая верификацию данных, выполнение операций с криптографическими активами, создание оповещений и другие важные функции.

3. Серверное приложение (Backend): серверная часть системы выполняет функции посредника между пользовательским интерфейсом и базой данных, обрабатывая запросы от клиентских приложений, реализуя бизнес-логику и обеспечивая доступ к данным.

4. База данных: она выполняет функции систематического хранения и организации информации о пользователях, криптографических активах, транзакциях и других необходимых объектах, являясь центральным хранилищем данных системы.

5. Внешние службы: система может взаимодействовать с внешними сервисами, такими как API-интерфейсы криптовалютных бирж, рыночные данные или системы оповещений, для расширения своей функциональности.

Процесс взаимодействия компонентов системы характеризуется следующими аспектами:

- Между пользовательским интерфейсом и бизнес-логикой: интерфейс пользователя инициирует запросы на выполнение разнообразных операций, например, приобретение криптографических активов; данные запросы обрабатываются бизнес-логикой, которая возвращает результаты выполнения.

- Между бизнес-логикой и серверным приложением: бизнес-логика, находящаяся на серверной стороне, получает запросы от интерфейса пользователя, обрабатывает их, и результаты обработки направляются обратно.

- Между серверным приложением и базой данных: серверное приложение осуществляет взаимодействие с базой данных посредством отправки запросов на чтение и запись информации, а также получения соответствующих ответов.

- Между серверным приложением и внешними сервисами: серверное приложение может вступать в коммуникацию с внешними сервисами с целью получения актуальных данных или отправки уведомлений пользователям.

Преимущества описанной архитектуры включают:

- Модульность: разделение системы на отдельные компоненты облегчает её масштабирование, обслуживание и повторное использование в других проектах.

- Надёжность: изоляция бизнес-логики от пользовательского интерфейса и базы данных способствует повышению целостности системы и упрощает управление возможными сбоями.

- Гибкость: такая архитектура позволяет без труда вносить изменения в систему, добавлять новые функции или модифицировать существующие, минимально влияя на другие компоненты.

- Безопасность: разделение компонентов усиливает защиту данных, обеспечивает более эффективный контроль доступа и ограничивает функциональность каждого отдельного модуля.

Эта архитектурная модель, благодаря своей модульной структуре, способствует не только упрощению масштабирования, но и облегчает процесс обслуживания системы, обеспечивая при этом повторное использование её компонентов в различных проектах. В то же время изоляция бизнес-логики от пользовательского интерфейса и базы данных усиливает целостность системы и упрощает управление потенциальными сбоями, что существенно повышает общую надёжность.

Гибкость архитектуры проявляется в способности легко вносить изменения, добавлять новые функции или модифицировать существующие без существенного влияния на другие части системы, что делает её адаптивной и способной быстро реагировать на изменяющиеся требования. Кроме того, разделение компонентов играет ключевую роль в обеспечении безопасности: оно не только улучшает защиту данных, но и позволяет осуществлять более эффективный контроль доступа, одновременно ограничивая функциональность каждого модуля, что дополнительно снижает риски компрометации системы.

Таким образом, рассматриваемая архитектура представляет собой сбалансированное сочетание модульности, надёжности, гибкости и безопасности, что делает её эффективным решением для разработки современных информационных систем.

2.4.2 Разработка интерфейса администратора

Разработка административного интерфейса для автоматизированной системы управления криптоактивами представляет собой критически важный элемент, который способствует значительному повышению эффективности управления и мониторинга всех ключевых компонентов данной системы. Этот интерфейс предоставляет доступ к расширенному функционалу и настройкам, которые необходимы для комплексного управления пользователями, активами, транзакциями, а также другими критически важными аспектами системы.

В числе основных функций административного интерфейса можно выделить следующие

- Управление пользователями: возможность создавать, редактировать и удалять учетные записи, а также просматривать и модифицировать профили пользователей, настраивая права доступа и определяя их роли. Это включает в себя как настройку разрешений, так и распределение ролей, что позволяет эффективно управлять доступом к различным частям системы.

- Управление криптоактивами: функции добавления, изъятия и удаления криптоактивов, а также просмотр их характеристик, таких как цена, объем и капитализация. Дополнительно, это включает настройку параметров активов, таких как комиссии и установление лимитов, что обеспечивает детальное управление финансовыми аспектами криптоактивов.

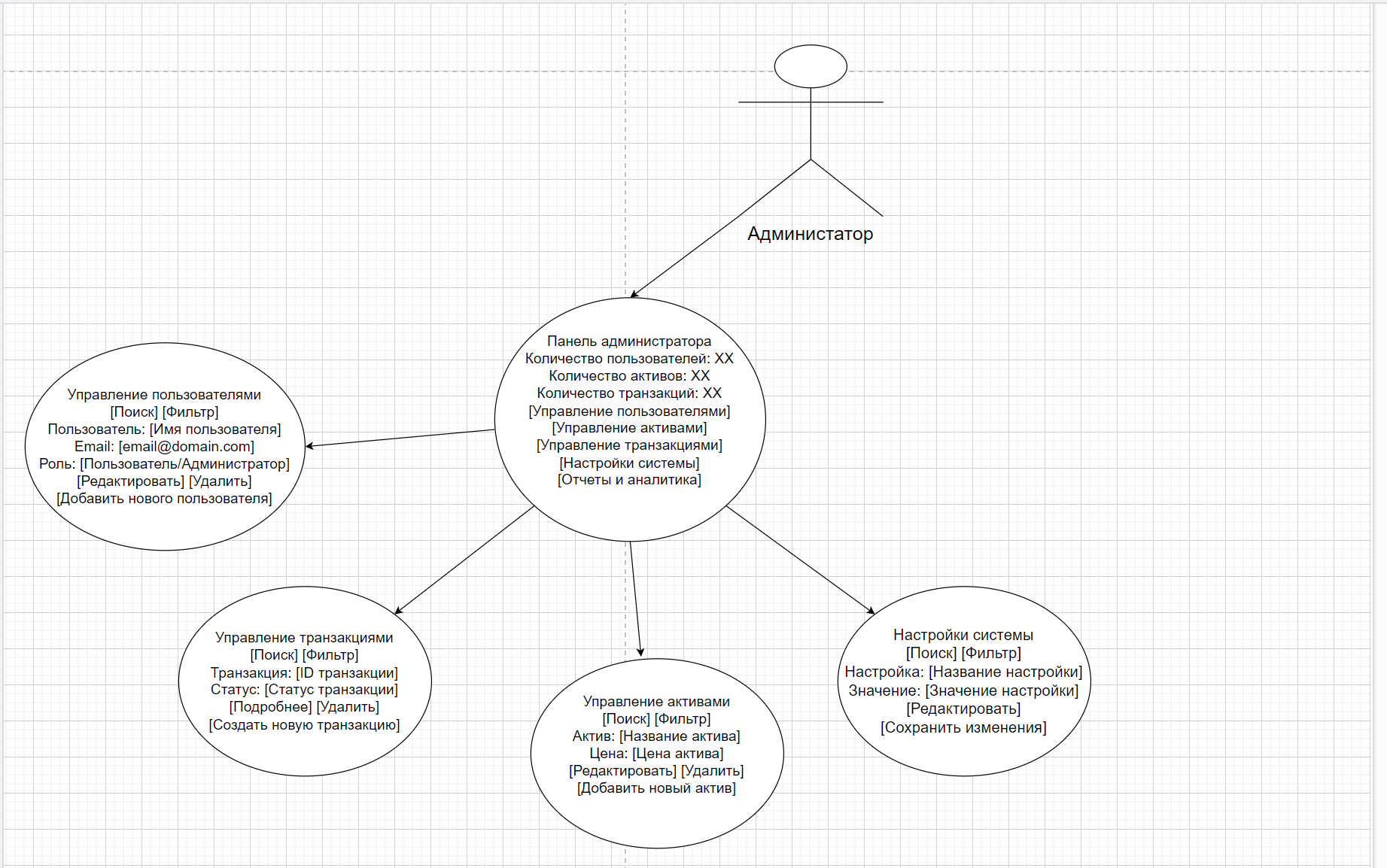
Следовательно, разработка административного интерфейса, наряду с проектированием макетов экранных форм, становится неотъемлемой частью процесса создания автоматизированной системы управления криптоактивами. Эти компоненты способствуют существенному улучшению как функциональности, так и управляемости всей системы, обеспечивая более высокий уровень контроля и оптимизации процессов.

Рисунок 8 – Диаграмма прецедентов для администатра автоматизированной системы управления криптоактивами ООО «Ван АйтиКонсалтинг»

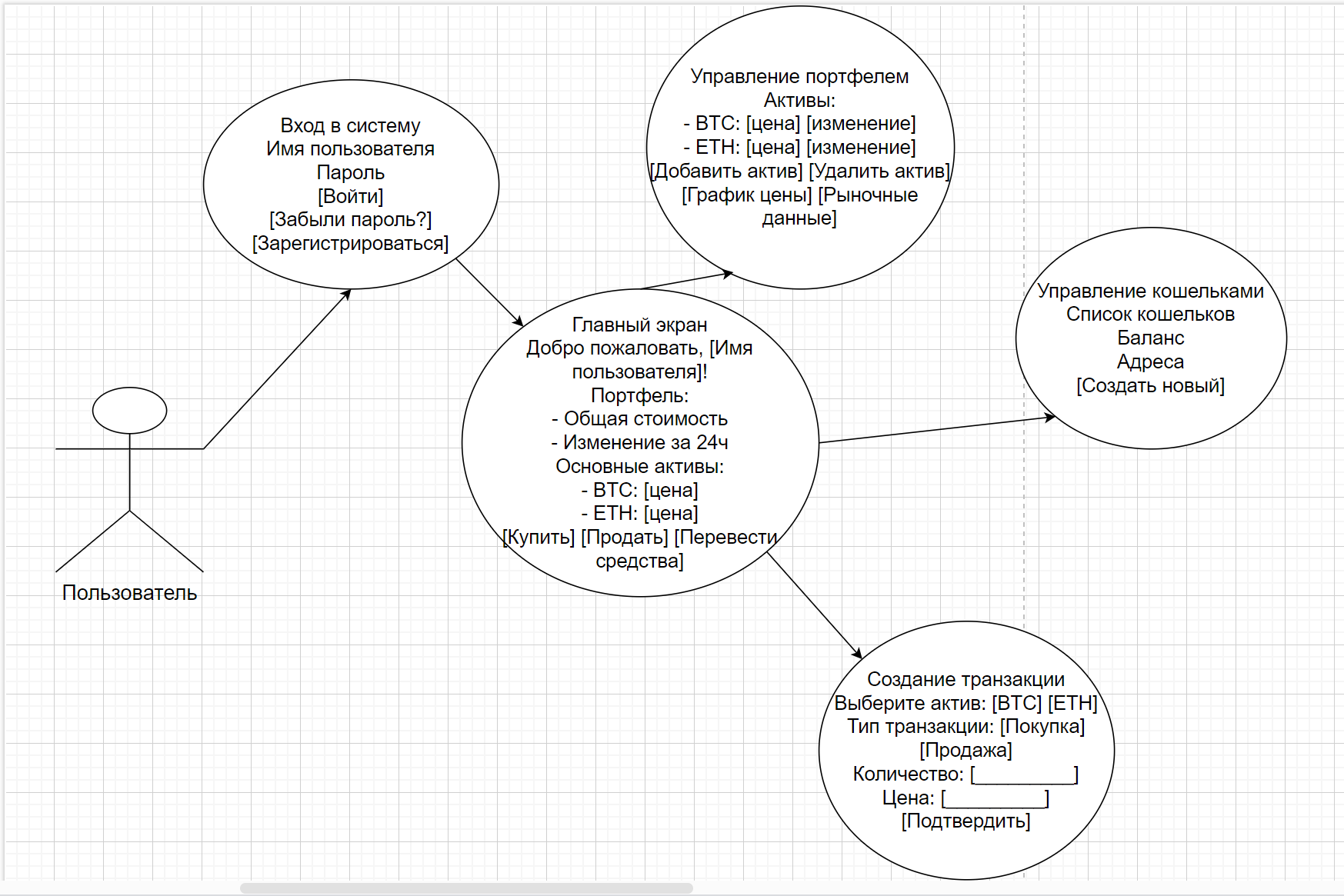


Рисунок 8 – Диаграмма прецедентов для пользователя автоматизированной системы управления криптоактивами ООО «Ван АйтиКонсалтинг»

Основные аспекты пользовательского интерфейса включают в себя следующие элементы:

1. Регистрация и аутентификация: данный модуль предусматривает интерфейс для регистрации новых пользователей и аутентификации уже зарегистрированных, а также обеспечивает функционал для восстановления забытых паролей.

2. Панель управления: эта секция предоставляет сводную информацию о текущем состоянии портфеля пользователя, отображая общую стоимость портфеля и его динамику за определенные временные периоды, а также содержит краткую информацию о ведущих криптоактивах.

3. Управление портфелем: данный элемент интерфейса позволяет пользователю добавлять, удалять и редактировать криптоактивы в своем портфеле, предоставляя доступ к детализированной информации о каждом активе, включая графики цен и рыночные данные.

4. Транзакции: этот модуль обеспечивает возможность создания новых транзакций (покупка, продажа, передача) и позволяет просматривать и фильтровать историю транзакций, предоставляя детализированную информацию о каждой из них.

5. Управление кошельком: функциональность данного модуля включает управление криптокошельками, что позволяет пользователю просматривать балансы и транзакции в наиболее часто используемых кошельках.

6. Настройки пользователя: этот компонент позволяет пользователю управлять своей личной информацией (имя, адрес электронной почты, номер телефона), а также настройками безопасности, такими как пароль и двухфакторная аутентификация; кроме того, он предоставляет возможность настраивать оповещения и уведомления.

Каждый из вышеупомянутых компонентов играет важную роль в обеспечении эффективного взаимодействия пользователя с системой, что гарантирует высокий уровень удобства использования, безопасности и функциональности. Эти атрибуты являются фундаментально важными для успешного управления криптоактивами в рамках автоматизированной системы. Таким образом, интерфейс, с его продуманной структурой и широким спектром функциональных возможностей, способствует оптимизации пользовательского опыта и повышению общей эффективности системы.

# **3 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ**

## 3.1 Настройка серверной среды

Настройка серверной среды является сложным и ответственным процессом, требующим тщательного учета множества аспектов для обеспечения стабильности, безопасности и высокой производительности системы управления криптоактивами. В ходе данного процесса необходимо пройти через несколько ключевых этапов, включающих выбор подходящей операционной системы, установку и конфигурацию серверного программного обеспечения, защиту данных, а также реализацию систем мониторинга и резервного копирования.

Первым этапом конфигурации является определение подходящей операционной системы, что требует тщательного анализа потребностей серверной среды. Среди наиболее часто рассматриваемых вариантов выделяются:

- Linux (включая такие популярные дистрибутивы, как Ubuntu Server, CentOS, Debian), который высоко ценится за свою стабильность, производительность и надежность.

- Windows Server — предпочтительный вариант для тех, кто интегрирован в экосистему Microsoft и требует совместимости с другими продуктами компании.

Выбор операционной системы должен основываться на специфических требованиях проекта, поддержке необходимых инструментов разработки и предпочтениях команды разработчиков.

Создание базы данных представляет собой критически важный этап в организации и эффективном хранении данных в системе управления криптоактивами. Основные шаги включают проектирование структуры базы данных, её реализацию, настройку параметров безопасности и оптимизацию производительности.

Проектирование схемы базы данных включает:

- Сбор требований: на данном этапе необходимо определить все данные, которые система должна хранить и обрабатывать, устанавливая структуру сущностей, атрибутов и их взаимосвязей.

- Создание диаграммы ER (Entity-Relationship), которая является визуальным представлением структуры базы данных, демонстрируя объекты (такие как пользователи, транзакции, кошельки), их атрибуты и взаимосвязи в виде отношений типа «один к одному», «один ко многим» и «многие ко многим».

Таким образом, эффективная настройка серверной среды и разработка базы данных являются неотъемлемыми частями создания надежной, масштабируемой и функциональной системы управления криптографическими активами. Эти процессы обеспечивают основу для построения системы, способной выдерживать высокие нагрузки, защищать данные и обеспечивать пользователям удобный и безопасный доступ к функционалу.

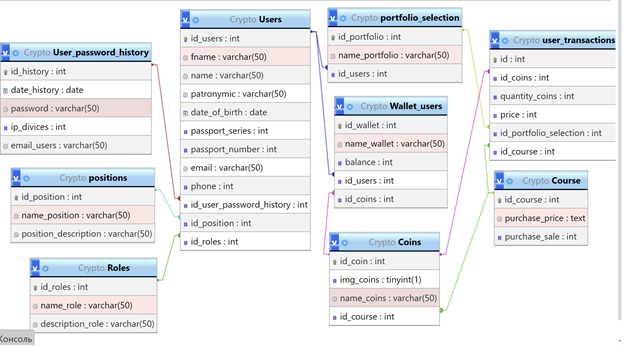
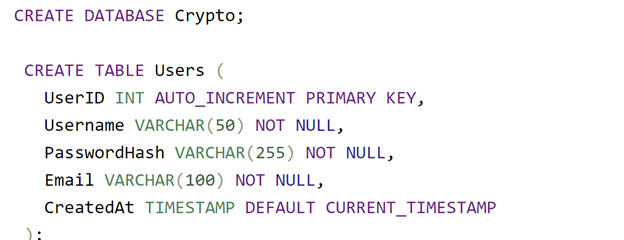


Рисунок 9. Схема базы данных.

Определение типов данных является критически важным аспектом в процессе проектирования базы данных, требующим детального подбора подходящего типа данных для каждого атрибута. В качестве примеров типов данных можно привести INT (целочисленные значения), VARCHAR (текст переменной длины), DATE (даты), DECIMAL (десятичные числа) и другие. Этот выбор занимает центральное место в оптимизации процессов архивации информации и повышении эффективности её обработки. Тщательно продуманный подход к определению типов данных не только фаворизирует усиление производительности базы данных, но и обеспечивает необходимую точность и целостность сохраняемой информации, что становится неотъемлемой основой для достижения высокой функциональности автоматизированных систем управления.



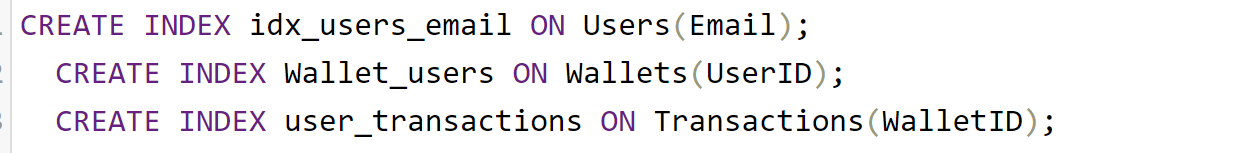


Рисунок 11: Настройка индексов и ключей для базы данных

Имплементация схемы базы данных начинается с использования системы управления базами данных MySQL, что следует за завершением этапа проектирования схемы. Этап создания базы данных включает физическое определение структуры данных на сервере, согласно разработанной и утвержденной схеме.

Настройка индексов и ключей играет ключевую роль в обеспечении высокой производительности и надежности базы данных. Конфигурация включает следующие критически важные элементы:

- Первичные ключи: определение уникальных идентификаторов для каждой таблицы, что позволяет однозначно идентифицировать каждую запись в системе.

- Внешние ключи: создание внешних ключей для поддержки ссылочной целостности между таблицами; например, ассоциирование «UserID» из таблицы «Пользователи» с таблицей «Кошельки», что обеспечивает назначение каждого кошелька только существующему пользователю.

- Индексы: разработка индексов для атрибутов, которые часто используются в запросах, способствует ускорению процессов чтения данных из базы данных.

Каждый из этих шагов играет решающую роль в создании структурированной, эффективной и надежной базы данных, служащей фундаментом для успешной работы автоматизированной системы управления криптографическими активами.

Безопасность базы данных

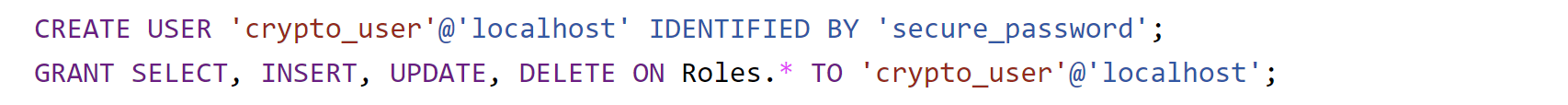
Безопасность баз данных является важным аспектом, требующим строгого контроля и применения мер по защите данных от несанкционированного доступа и утечки информации. и других угроз. К основным мерам безопасностиотносятся:

Рисунок 12: Создание пользователей с ограниченными правами

Шифрование данных:

В контексте безопасности баз данных, шифрование выполняет ключевую функцию по защите конфиденциальной информации от несанкционированного доступа. Шифрование данных, особенно конфиденциальных элементов, таких как пароли, необходимо для обеспечения конфиденциальности и безопасности. Этот процесс обеспечивает, что даже в случае утечки данных, информация останется защищенной, поскольку будет доступна только тем, кто обладает соответствующими ключами для её расшифровки.

Создание пользователей с ограниченными правами:

Важным аспектом управления безопасностью баз данных является создание пользователей с ограниченными правами доступа. Ограничение прав пользователей минимизирует риски, связанные с потенциальными нарушениями безопасности, так как доступ к критически важным данным и функциональностям будет строго контролироваться и ограничен только необходимыми для выполнения задач правами. Это не только повышает уровень безопасности системы, но и способствует её общей устойчивости к атакам.

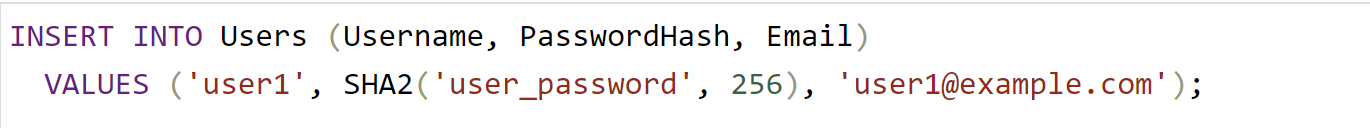
Шифрование, наряду с созданием пользователей с ограниченными правами, составляет фундаментальный элемент стратегии обеспечения безопасности баз данных. Такие меры гарантируют высокий уровень защиты данных, что критически важно для поддержания доверия пользователей и соблюдения юридических норм и стандартов в сфере информационной безопасности.

Рис. 13. Создание зашифрованных данных

## 3.3 Основы написания логики

Разработка серверной логики для программы управления криптоактивами представляет собой центральный элемент создания автоматизированной системы управления криптоактивами.

В рамках данного этапа происходит реализация основной логики приложения, которая включает обработку данных, взаимодействие с базой данных MySQL, а также исполнение бизнес-процессов. В процессе разработки применяется язык программирования Python; для создания пользовательского интерфейса используются инструменты PyQt5 и Qt Designer в среде PyCharm, что существенно упрощает проектирование графического пользовательского интерфейса (GUI).

Настройка среды разработки

Перед началом программирования критически важно тщательно настроить среду разработки для обеспечения эффективности и непрерывности рабочего процесса. Данный процесс включает в себя несколько основных шагов:

- Установка Python: проверьте, установлена ли на вашем компьютере последняя версия Python, обратившись к официальному веб-сайту Python и следуя рекомендациям по установке.

- Установка необходимых библиотек: используйте диспетчер пакетов pip для установки библиотек PyQt5 и PyMySQL, которые необходимы для разработки пользовательского интерфейса и взаимодействия с базой данных. Введите следующие команды в командной строке или терминале для установки этих библиотек.

Каждый из указанных шагов способствует формированию оптимизированной среды разработки, что является обязательным условием для эффективного и точного проектирования системы управления криптографическими активами.

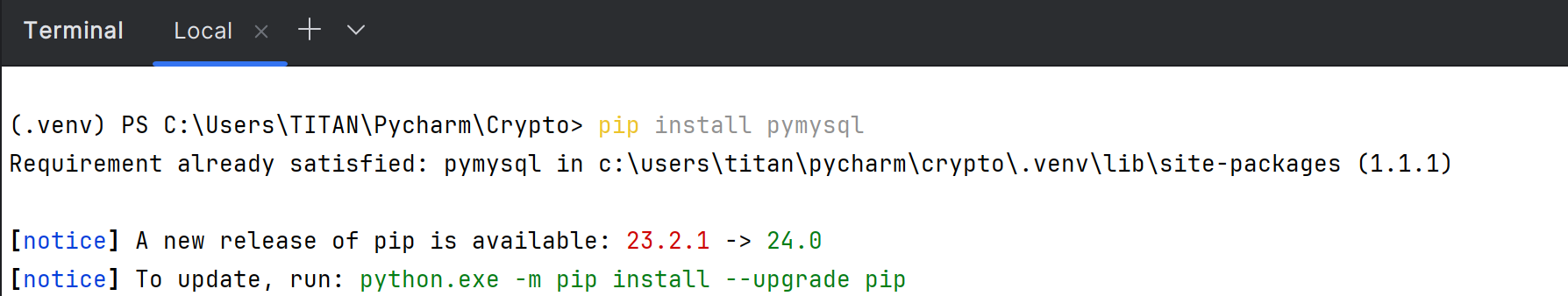
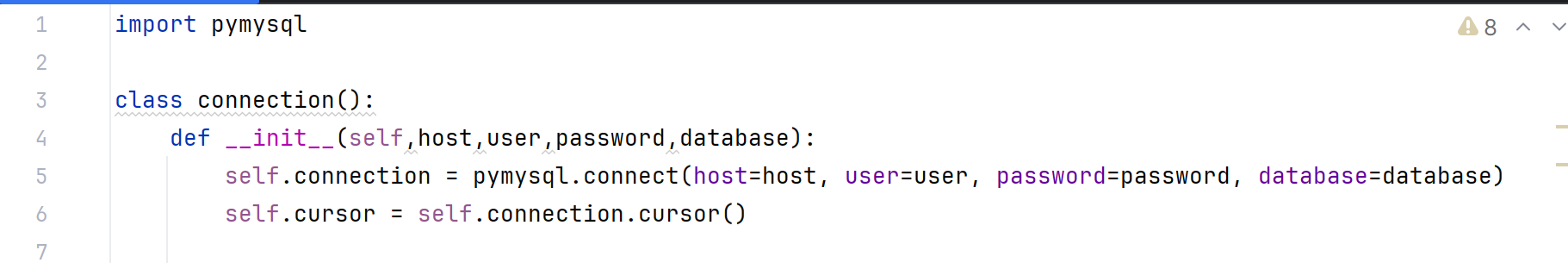


Рисунок 14. Загрузка библиотеки

Подключение к базе данных MySQL

Первым шагом в разработке серверной части является создание подключения к базе данных MySQL. Вот для чего нужна библиотека PyMySQL:

- настройки подключения: определение параметров подключения к базе данных (хост, имя базы данных, пользователь, пароль).

**Рисунок 15: Подключение к базе данных**

**Реализация основных функций**

После установления соединения с базой данных мы приступаем к реализации основных функций серверной логики. Эти функции охватывают операции, связанные с пользователями, кошельками и транзакциями, и являются неотъемлемыми компонентами для обеспечения функциональности бэкенда.

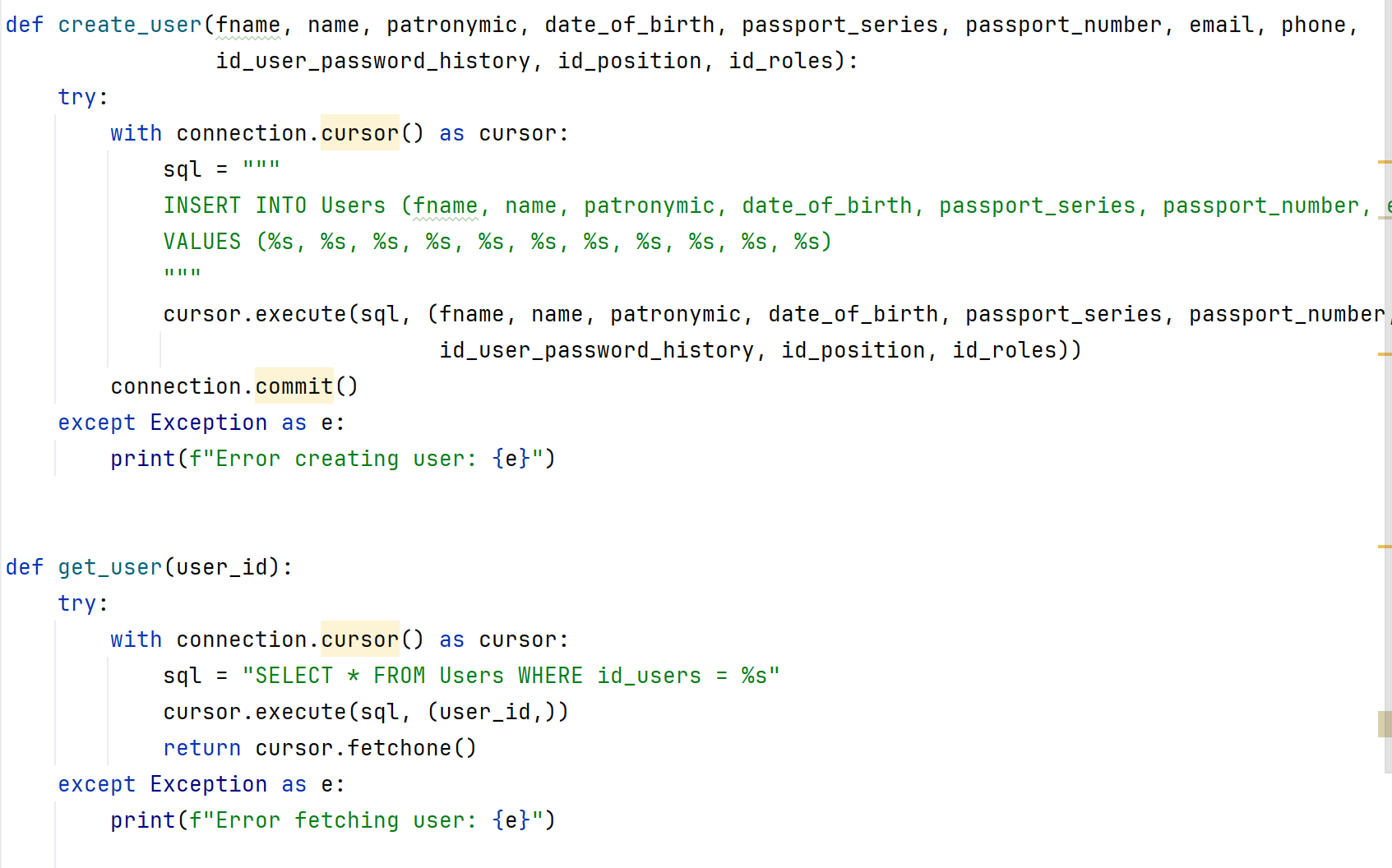


Рисунок. 15 Взаимодействие с пользователем

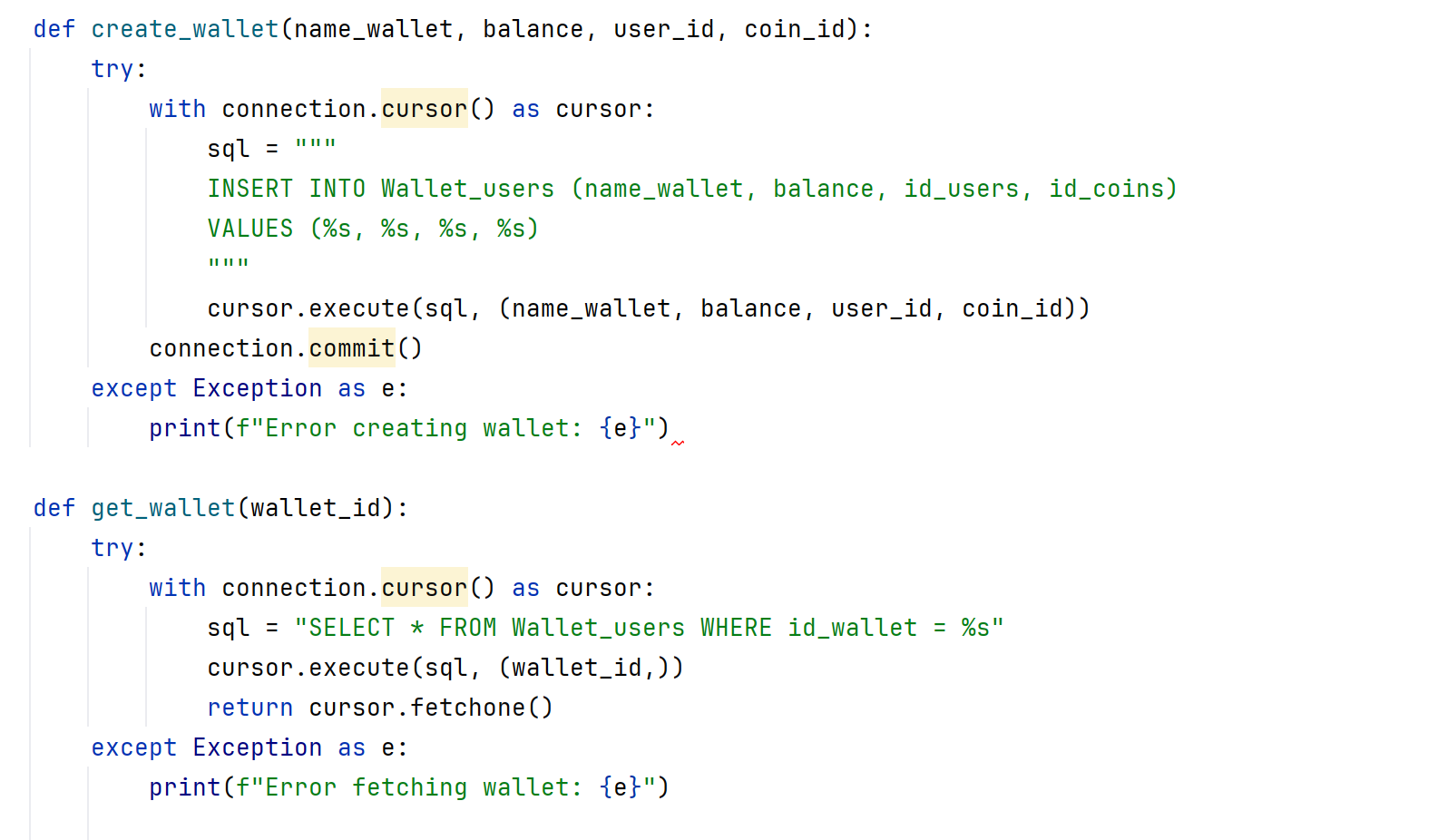
- Работа с кошельками.

Рисунок 17 Работа с кошельком

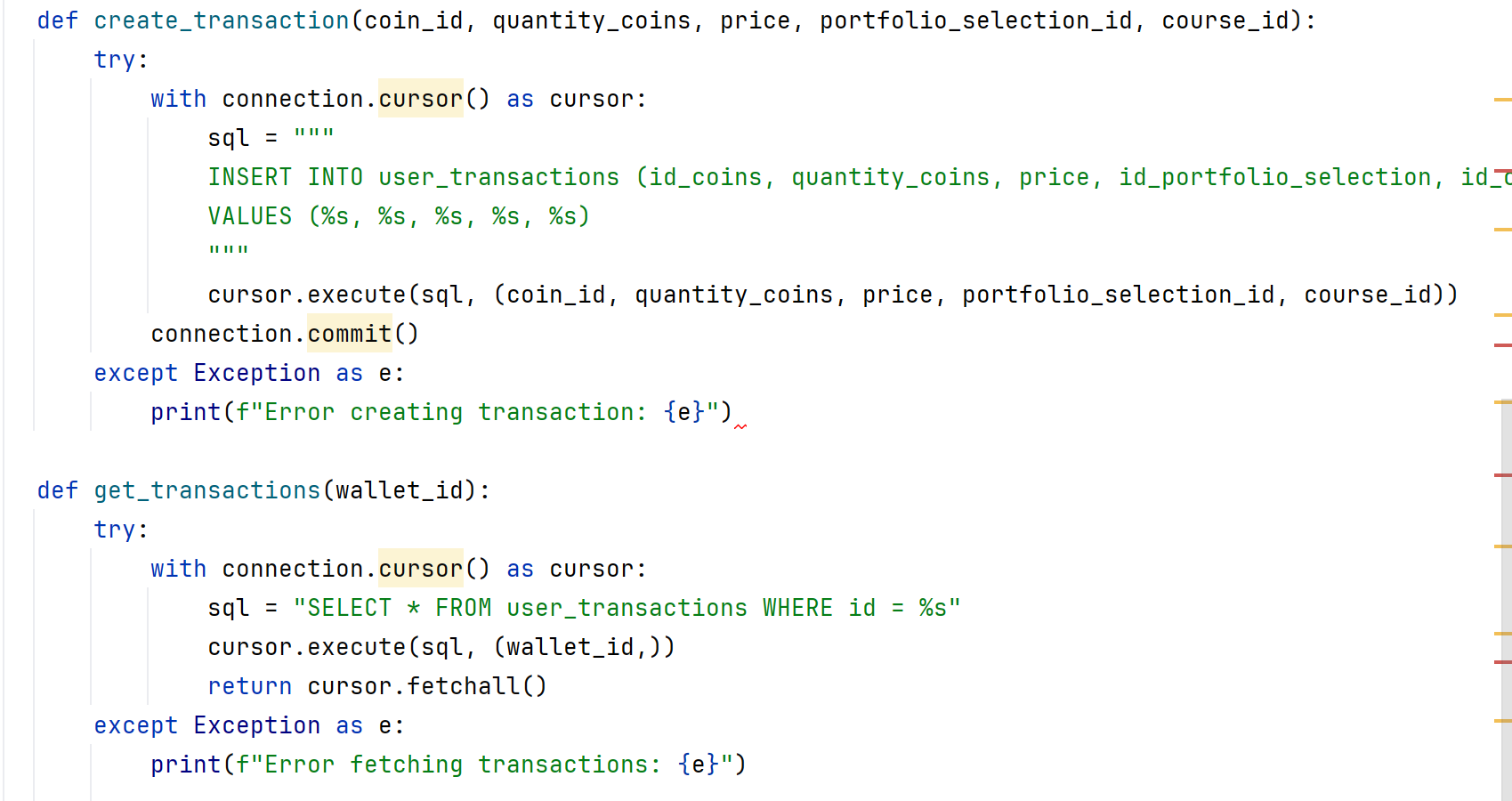
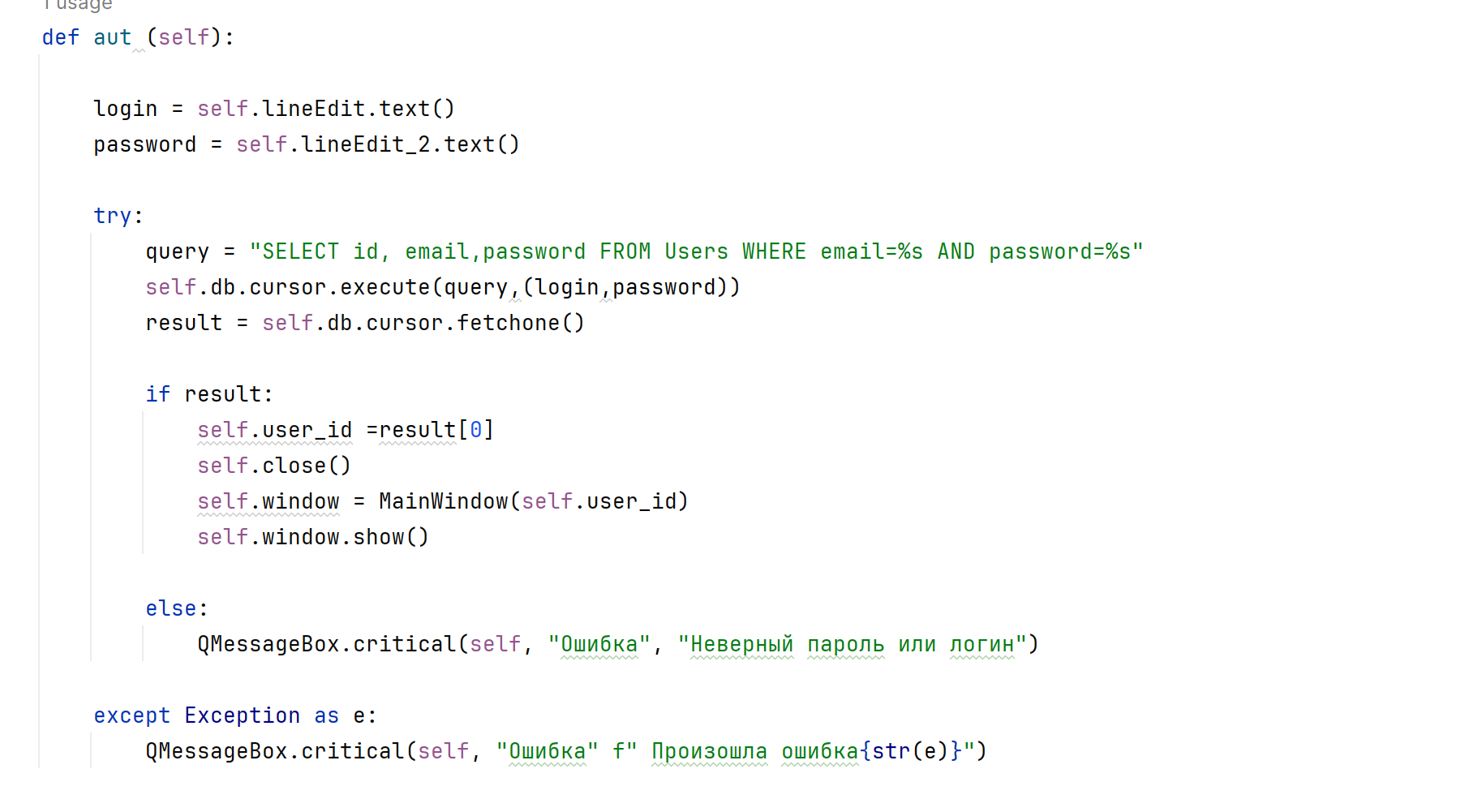
- Работа с транзакциями 

Рисунок 18 Работа с транзакциями

Обработка и протоколирование ошибок

Реализация бизнес-логики представляет собой критический компонент в разработке системы управления криптоактивами, включающий внедрение процессов, специфичных для данной системы. Эти процессы охватывают управление криптографическими активами, обработку транзакций, аутентификацию и авторизацию пользователей. Основные аспекты бизнес-логики включают: аутентификация пользователя:



Риск 20 Авторизация пользователя

- Управление криптографическими активами:

Реализация функций работы с внешними API криптовалютных бирж для получения курсов и обработки транзакций.

Рисунок 21 Выбор криптовалютных монет

Интеграция с интерфейсом

Рисунок 22: Вывод данных в форму

Завершающий этап разработки включает интеграцию серверной логики с пользовательским интерфейсом, реализованным при помощи PyQt5. Этот процесс охватывает сопряжение серверных функций с элементами графического интерфейса, что необходимо для обеспечения полноценной функциональности системы. Предпринятые действия направлены на создание стабильной и функционально обогащённой серверной логики, которая будет осуществлять эффективное взаимодействие с пользовательским интерфейсом и базой данных, предоставляя пользователям все необходимые инструменты для управления криптографическими активами. Эти шаги не только повышают удобство использования системы, но также улучшают её надёжность и производительность, гарантируя безопасное и результативное управление активами.

## 3.4 Написание кода интерфейса программы

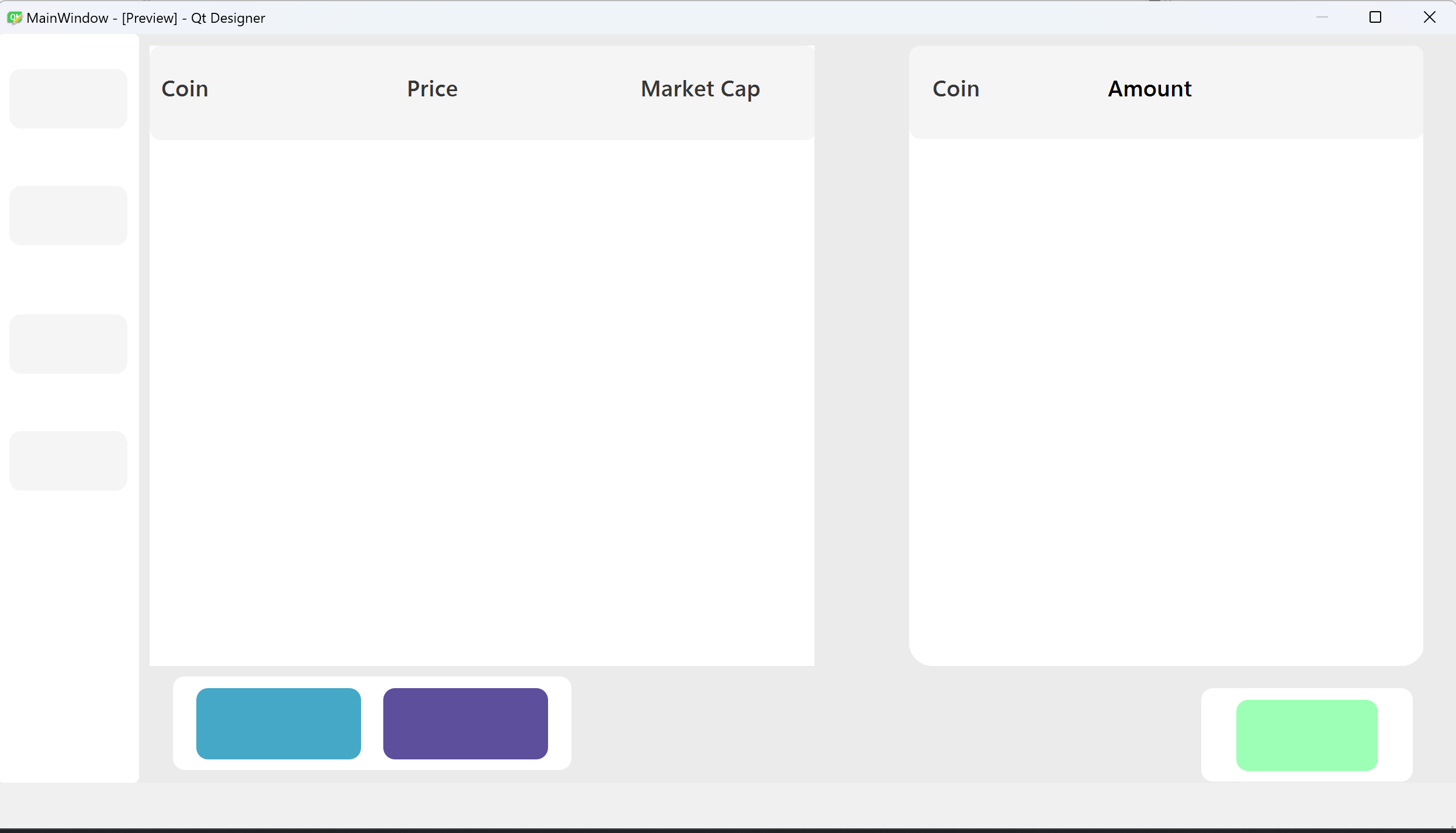
Разработка программного интерфейса является центральным этапом в процессе создания автоматизированной системы управления криптоактивами. На данном этапе осуществляется разработка пользовательского интерфейса, который обеспечивает комфортное и эффективное взаимодействие пользователя с системой. Для достижения поставленных целей применяется комплект инструментов PyQt5 в сочетании с Qt Designer в рамках среды PyCharm, что способствует быстрому и качественному созданию графического пользовательского интерфейса. Такой метод не только повышает визуальную привлекательность системы, но также расширяет её функциональные возможности и упрощает взаимодействие пользователя с основными функциями управления криптоактивами.

Рисунок 23 Форма Dashboard.ui

Перед началом разработки интерфейса настоятельно рекомендуется провести детальное проектирование макета экранной формы и точно определить структуру и размещение элементов управления. Такая методология способствует глубокому пониманию пользовательских требований и способствует созданию удобного и интуитивно понятного пользовательского интерфейса. Применение этой стратегии позволяет не только повысить функциональные возможности интерфейса, но и оптимизировать процесс взаимодействия пользователя с системой, что, в свою очередь, увеличивает эффективность работы конечного пользователя с автоматизированной системой управления криптоактивами.

Пример макета главного окна (Authorization.ui)

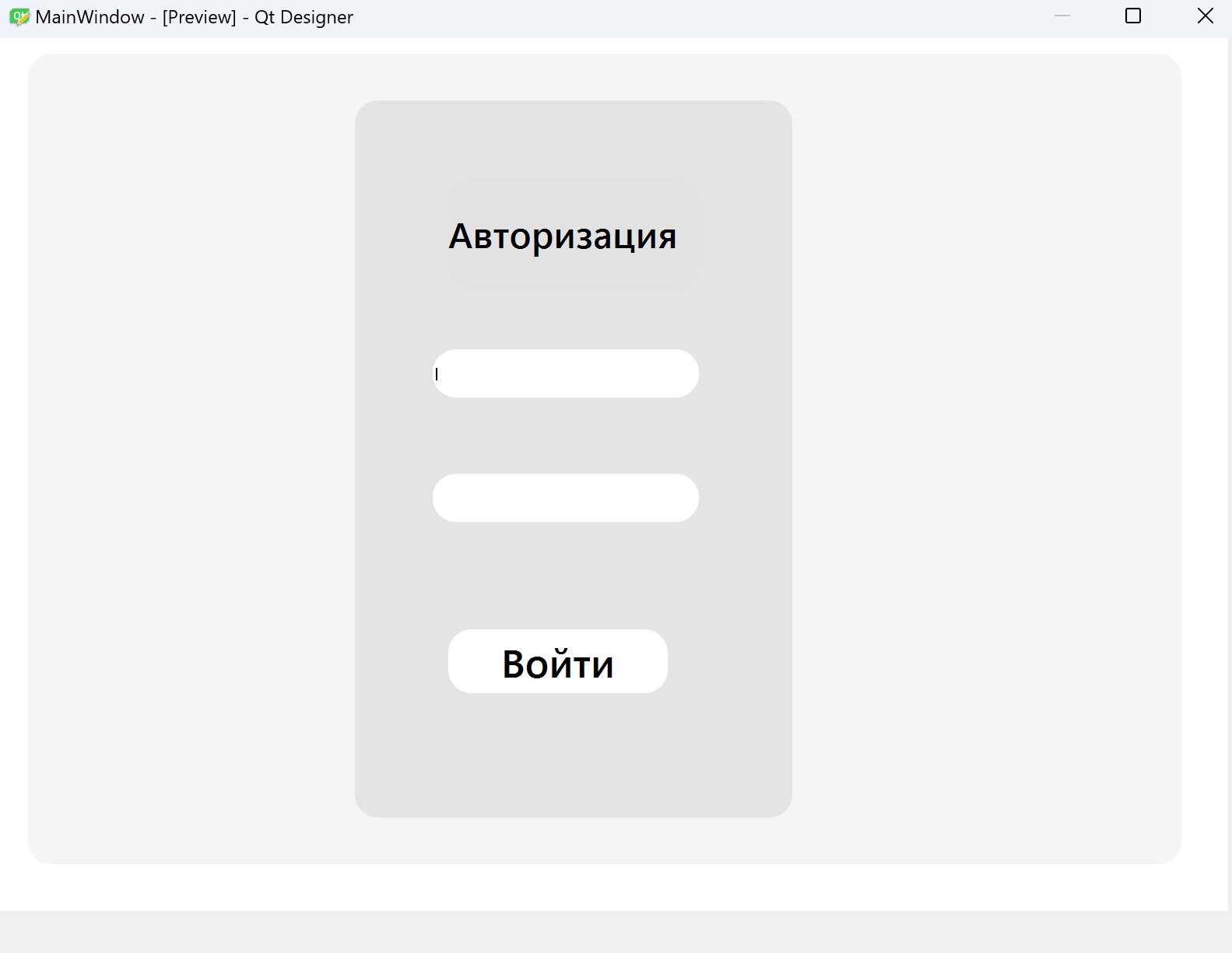


Рисунок 24 Форма Authorization.ui

Разработка интерфейса

После завершения этапа проектирования экранного макета начинается процесс разработки интерфейса с применением PyQt5. Конфигурация элементов управления и их стратегическое размещение на форме происходит через файл с расширением .ui, который генерируется в Qt Designer. Последующая интеграция этого файла в код Python способствует созданию функционального пользовательского интерфейса. Такой подход не только гарантирует эффективную интеграцию визуальных и функциональных компонентов системы, но также способствует повышению общей производительности и улучшению удобства использования разрабатываемой системы управления криптографическими активами.

Пример кода для загрузки и отображения формы входа (Dashboard.py)



Рисунок 25. Запуск формы Dashboard.ui.

Реализация функциональности

После проектирования интерфейса, следует реализация функциональностей, связанных с каждым элементом управления. Этот этап включает обработку событий, валидацию входных данных и взаимодействие с серверной частью системы.

Тестирование и отладка

Завершение разработки интерфейса требует проведения тщательных тестирований и отладки для подтверждения корректной функциональности программы. Важно уделить внимание проверке эффективности всех элементов управления, а также выявлению и исправлению возникающих ошибок и исключений.

Интеграция с серверной частью

Конечный этап включает интеграцию разработанного интерфейса с серверной частью системы. Процесс включает вызов функций системы в ответ на пользовательские действия, передачу данных между пользовательским интерфейсом и серверной частью, а также обновление интерфейса пользователя на основе полученных данных.

Эти шаги способствуют созданию эффективной, надежной и ориентированной на пользователя системы, способной адекватно реагировать на запросы пользователя и обеспечивать оперативную обработку и передачу данных в контексте автоматизированной системы управления криптоактивами.

Пример вызова внутренней функции при входе пользователя в систему (Authorization.py):



Риск 26 Авторизация пользователя

Эти шаги позволят создать удобный и функциональный интерфейс программы, который позволит пользователям эффективно управлять своими криптоактивами.

# **4 ТЕСТИРОВАНИЕ И ОТЛАДКА**

## 4.1 Функциональное тестирование автоматизированной системы

Функциональное тестирование направлено на проверку соответствия работы системы ее функциональным требованиям. Основная цель данного вида тестирования - убедиться, что система выполняет все заявленные функции корректно и без ошибок. В контексте нашей автоматизированной системы для управления базой данных (например, системой управления портфелями криптовалют), функциональное тестирование будет включать несколько основных шагов:

**идентификация функциональных требований:** определение основных функций системы (CRUD-операции для различных сущностей, аутентификация и авторизация пользователей, обработка транзакций и т.д.).

Составление списка тест-кейсов, покрывающих все функциональные требования.

**Разработка тест-кейсов:** каждый тест-кейс должен содержать четкое описание действий, ожидаемый результат и критерии успешности. Тест-кейсы должны покрывать как положительные, так и отрицательные сценарии использования системы.

**Выполнение тест-кейсов:** проведение тестирования вручную или с использованием автоматизированных тестов. Фиксация результатов выполнения тест-кейсов (пройден, не пройден, блокирован и т.д.).

**Анализ и отчетность:** анализ результатов тестирования.

Составление отчетов о дефектах и их передача разработчикам для исправления. Повторное тестирование после исправления дефектов.

Примеры тест-кейсов для системы управления базой данных

1. Тест-кейс: Создание нового пользователя

**Описание:** Проверка функции создания нового пользователя в системе.

**Шаги:**

открыть форму регистрации пользователя.

Заполнить все обязательные поля (имя, фамилия, email, пароль и т.д.).

Нажать кнопку "Регистрация".

**Ожидаемый результат:**

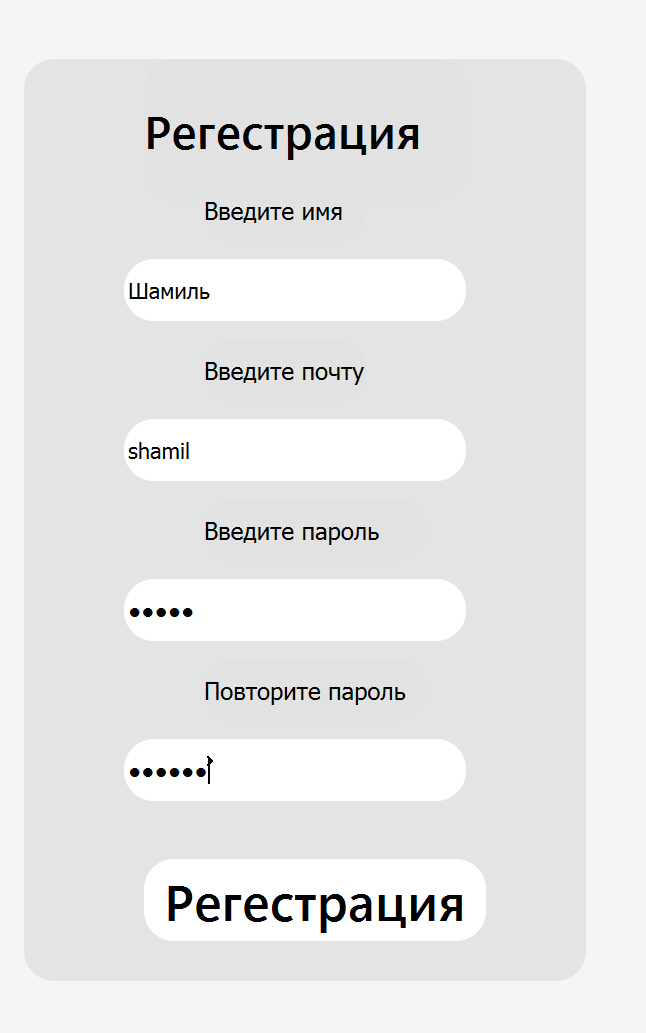


Рисунок 27 Форма регестрации

Пользователь успешно создается и появляется запись в таблице "Users".

Пользователь получает подтверждение о успешной регистрации.

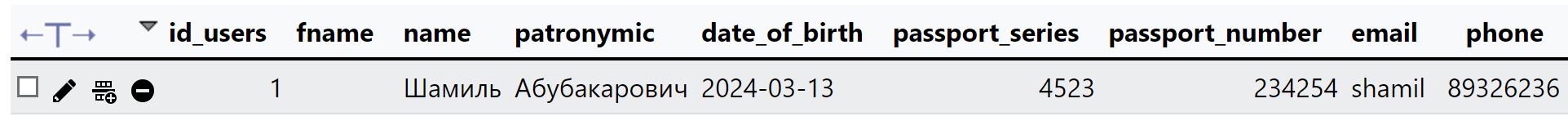


Рисунок 28 Добавление нового пользователя в базу данных

2. Тест-кейс: Аутентификация пользователя

**Описание:** проверка функции входа в систему.

**Шаги:**

Открыть форму входа в систему.

Ввести корректные учетные данные (email и пароль).

Нажать кнопку "Войти".

**Ожидаемый результат:**

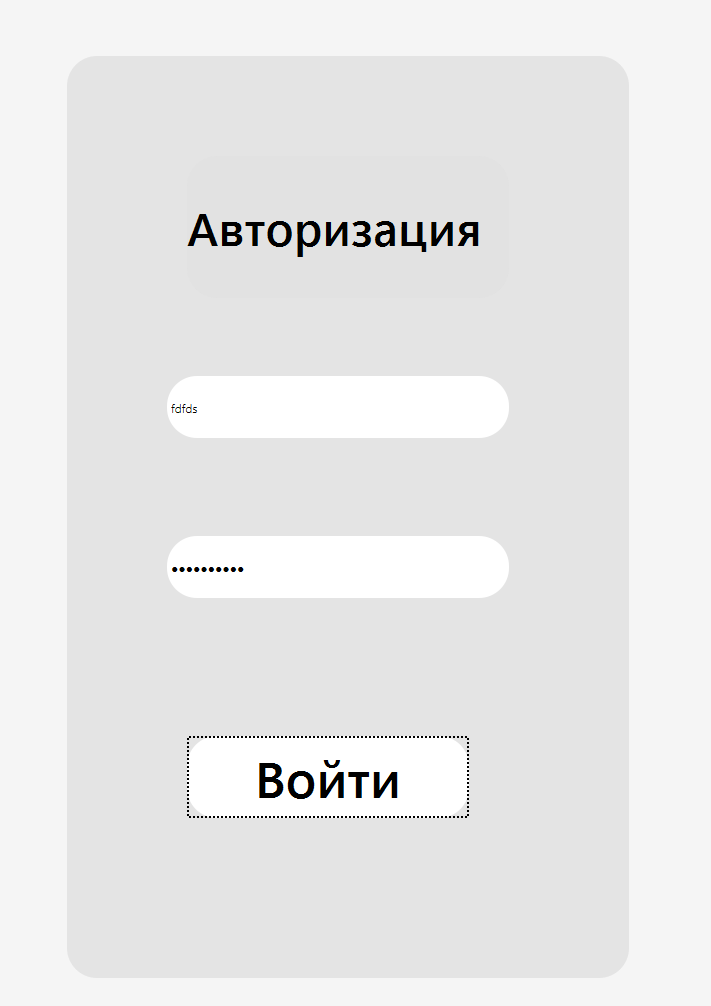


Рисунок 29 Форма авторизации

Функциональное тестирование является важным этапом в процессе разработки автоматизированной системы. Оно позволяет выявить и устранить дефекты на ранних стадиях, обеспечивая высокое качество конечного продукта и удовлетворенность пользователей. Регулярное проведение функционального тестирования и анализ его результатов способствуют непрерывному улучшению системы и ее функциональности.

# **5. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ**

1. Определение времени на разработку и комплексные испытания программного обеспечения:

1.1. Время на разработку программного обеспечения

где КДп – длительность разработки, согласованная с заказчиком в днях,

Вр – длительность рабочего дня в часах (Вр = 8 часов);

1.2. Время на комплексные испытания программного обеспечения.

Срок опытной эксплуатации (КДоэ) устанавливается в размере 50 % от длительности разработки, согласованной с заказчиком);

где КДоэ – длительность опытной эксплуатации в днях,

Н – норматив трудоемкости при проведении опытной эксплуатации, в часах (Н = 0,0004 часа);

1.3. Общее время на разработку и комплексные испытания программного обеспечения:

где Тп - время на разработку программного обеспечения в часах;

Тоэ - время на комплексные испытания программного обеспечения

2. Расчет фонда оплаты труда за разработку и комплексные испытания программного обеспечения.

2.1. Расчет оплаты труда за разработку программного обеспечения.

где Тп – время на разработку программного обеспечения, час.

СЧп – стоимость часа работы программиста, руб. (Чп: от 500 руб. до 800 руб. за один час)

2.2. Расчет оплаты труда за проведение опытной эксплуатации программной системы.

где Тоэ – время на комплексные испытания программного обеспечения, в

часах

СЧоэ – стоимость часа работы тестировщика, руб. (Чоэ: от 350 руб. до 600 руб. за один час)

2.3. Оплата труда за разработку и комплексные испытания программного обеспечения

где ЗПп – оплаты труда за разработку программного обеспечения, руб.

ЗПоэ – оплаты труда за проведение опытной эксплуатации программной системы.

2.4. Расчет страховых взносов от оплаты труда за разработку и комплексные испытания программного обеспечения

где СВзп – сумма страховых взносов от заработной платы, в рублях.

%Озп - процент отчислений от заработной платы (%Озп = 30 %).

2.5. Фонд оплаты труда за разработку и комплексные испытания программного обеспечения.

где ЗПобщ - Оплата труда за разработку и комплексные испытания программного обеспечения

СВзп – сумма страховых взносов от заработной платы, в рублях.

3. Расчет накладных и прочих расходов при разработке и комплексных испытаниях программного обеспечения.

3.1. Расчет накладных расходов.

К накладным расходам относятся расходы на управление подразделением, содержание и ремонт зданий, оборудования, коммунальные услуги и т. п.

Накладные расходы (%Нр) принимаются в размере 40% от оплаты труда за разработку и комплексные испытания программного обеспечения (ЗПобщ).

3.2. Расчет прочих затрат организации.

Прочие затраты (%Пз) составляют 25% от оплаты труда за разработку и комплексные испытания программного обеспечения (ЗПобщ).

4. Составление калькуляции разработки и комплексных испытаний программного обеспечения.

Таблица 2- Калькуляция себестоимости разработки и комплексных испытаний программного обеспечения.

|  |  |  |
| --- | --- | --- |
| №  п/п | Наименование статьи расходов | Сумма, руб. |
| 1 | Фонд оплаты труда за разработку и комплексные испытания программного обеспечения () | 218.403,09 |
| 2 | Накладные расходы () | 67.201,2 |
| 3 | Прочие затраты () | 42.000,75 |
| 4 | Затраты на проведение работ (Затр) | ∑ 327.695,04 |

5. Расчет финансовых показателей разработки и комплексного испытания программного обеспечения.

5.1. Расчет прибыли за выполнение работ:

Пр = 374.407,02 × 30% = 98.308,512 руб.

Процент прибыли (%Пр.) составляет от 20 до 30% от затрат на проведение работ (Затр.)

5.2. Налог на прибыль

Нпр = 112.322,11 × 20% = 19.661,70 руб.

где: Снп – ставка налога на прибыль (Снп = 20%).

5.3. Чистая прибыль

Пч = 98.308,512 – 19.661,70 = 78.646,812 руб.

где: Пр. – сумма прибыли за выполнение работ, руб.

Нпр. – сумма налога на прибыль, руб.

5.4. Цена выполненной работы:

Цр = 327.695,04 + 98.302,512 = 425.997,552 руб.

где: Затр. – затраты на проведение работ, руб.

Пр – сумма прибыли за выполнение работ, руб.

6. Финансово-экономические показатели выполнения работ

Таблица 3 - Финансово-экономические показатели выполненных работ

|  |  |  |
| --- | --- | --- |
| Наименование | Единица измерения | Значения показателя |
| Общее время на разработку и комплексные испытания программного обеспечения (Т) | час. | 240,006 |
| Затраты на проведение работ | руб. | 327.695,04 |
| Процент прибыли при реализации проекта | % | 30 |
| Чистая прибыль | руб. | 78.648,812 |
| Цена выполненной работы | руб. | 425.997,522 |

# **Заключение**

В ходе разработки автоматизированной системы управления криптоактивами для ООО "Ван АйтиКонсалтинг" были выполнены ряд ключевых этапов, начиная с анализа предметной области и постановки задачи, заканчивая практической реализацией и тестированием системы.

На основе анализа существующих криптовалютных сайтов и приложений были выделены основные функциональные требования к системе. Задачи на разработку автоматизированной системы управления криптоактивами были сформулированы с учетом потребностей и ожиданий заказчика.

В процессе проектирования автоматизированной системы были разработаны предварительные проектные решения, включая диаграмму прецедентов, выбор инструментальных средств реализации, проектирование базы данных и разработку пользовательского интерфейса. Была учтена архитектура системы и определены основные компоненты.

Практическая реализация системы включала настройку серверного окружения, создание базы данных с использованием MySQL и написание бэкенд логики программы на языке Python. Для создания пользовательского интерфейса был использован инструмент PyQT5 с Qt Designer на PyCharm.

Важным этапом было тестирование и отладка разработанной системы. Проведено функциональное тестирование для проверки соответствия системы заявленным требованиям.

Наконец, технико-экономическое обоснование позволило оценить эффективность и целесообразность разработки системы управления криптоактивами для ООО "Ван АйтиКонсалтинг".

В результате выполненных работ была создана функциональная и надежная автоматизированная система управления криптоактивами, которая позволит заказчику эффективно управлять своими криптовалютными активами и повысить уровень безопасности и удобства работы с ними.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления: Стандартинформ, 2017
2. ГОСТ 7.1—2003 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления
3. ГОСТ 7.9—95 Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования
4. ГОСТ 7.11—2004 (ИСО 832:1994) Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках
5. ГОСТ 7.12—93 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила
6. ГОСТ 7.80—2000 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Заголовок. Общие требования и правила составления
7. ГОСТ 7.82—2001 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления
8. Binance [Электронный ресурс]. – Режим доступа: https://www.binance.com/ru
9. FTX [Электронный ресурс]. – Режим доступа: <https://www.ftx.com/ru>
10. COSMOS [Электронный ресурс]. – Режим доступа: <https://cosmos.network/>
11. Polkadot [Электронный ресурс]. – Режим доступа: <https://polkadot.network/ru/>
12. Python [Электронный ресурс]. – Режим доступа: <https://www.python.org/>
13. C#[Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/>
14. Java [Электронный ресурс]. – Режим доступа: <https://www.java.com/ru/>
15. PyCharm[Электронный ресурс]. – Режим доступа: <https://www.jetbrains.com/ru-ru/pycharm/>
16. VisualStudio [Электронный ресурс]. – Режим доступа: <https://code.visualstudio.com/>
17. Atom [Электронный ресурс]. – Режим доступа: <https://atom.io/>
18. Tradingview [Электронный ресурс]. – Режим доступа: <https://ru.tradingview.com>
19. Coinmarketcap [Электронный ресурс]. – Режим доступа: <https://coinmarketcap.com/ru/>
20. Dropstab [Электронный ресурс]. – Режим доступа: <https://dropstab.com/>
21. Glassnode [Электронный ресурс]. – Режим доступа: <https://glassnode.com/>
22. Cryptoquant [Электронный ресурс]. – Режим доступа: <https://cryptoquant.com>
23. Messari [Электронный ресурс]. – Режим доступа: https://messari.io

**Листинг 1**

*import sys  
import requests  
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QMessageBox  
from PyQt5.uic import loadUi  
  
class PriceQuotation(QMainWindow):  
 def \_\_init\_\_(self):  
 super(PriceQuotation, self).\_\_init\_\_()  
 loadUi("../ui\_forms/untitled.ui", self)  
 self.loadBitcoinPrices()  
  
 def loadBitcoinPrices(self):  
 try:  
 url = "https://api.coingecko.com/api/v3/simple/price?ids=bitcoin,ethereum,tether,binancecoin,solana,usd-coin,aptos,sui,strike,aave,flow,arbitrum,hashflow&vs\_currencies=usd&include\_market\_cap=true"  
 response = requests.get(url)  
 data = response.json()  
  
 coins = {  
 "bitcoin": "",  
 "ethereum": "",  
 "tether": "",  
 "binancecoin": "",  
 "solana": "",  
 "usd-coin": "",  
 "aptos": "",  
 "sui": "",  
 "strike": "",  
 "aave": "",  
 "flow": "",  
 "arbitrum": "",  
 "hashflow": ""  
 }  
  
 y\_offset = 30  
 for coin\_id, symbol in coins.items():  
 price = data[coin\_id]['usd']  
 market\_cap = data[coin\_id]['usd\_market\_cap']  
 self.displayCoinData(symbol, price, market\_cap, y\_offset)  
 y\_offset += 30 # Update offset for next row  
  
 except Exception as e:  
 QMessageBox.critical(self, "Error", str(e))  
  
 def displayCoinData(self, symbol, price, market\_cap, y\_offset):  
 try:  
 label\_price = QLabel(f"{symbol} {price:,.2f}", self.scrollAreaWidgetContents)  
 label\_price.setGeometry(220, y\_offset, 100, 150)  
 label\_price.setStyleSheet("color: rgb(50, 50, 50);")  
  
 label\_market\_cap = QLabel(f"{symbol} {market\_cap:,.2f}", self.scrollAreaWidgetContents)  
 label\_market\_cap.setGeometry(400, y\_offset, 200, 150)  
 label\_market\_cap.setStyleSheet("color: rgb(50, 50, 50);")  
  
 # Make sure to show the new labels  
 label\_price.show()  
 label\_market\_cap.show()  
 except Exception as e:  
 QMessageBox.critical(self, "Error", str(e))  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = PriceQuotation()  
 window.show()  
 sys.exit(app.exec())*

*import logging  
from database.connection import connection  
  
logging.basicConfig(filename='app.log', level=logging.INFO)  
  
  
def log\_error(message):  
 logging.error(message)  
  
  
def log\_info(message):  
 logging.info(message)  
  
  
  
def create\_user(fname, name, patronymic, date\_of\_birth, passport\_series, passport\_number, email, phone,  
 id\_user\_password\_history, id\_position, id\_roles):  
 try:  
 with connection.cursor() as cursor:  
 sql = """  
 INSERT INTO Users (fname, name, patronymic, date\_of\_birth, passport\_series, passport\_number, email, phone, id\_user\_password\_history, id\_position, id\_roles)  
 VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)  
 """  
 cursor.execute(sql, (fname, name, patronymic, date\_of\_birth, passport\_series, passport\_number, email, phone,  
 id\_user\_password\_history, id\_position, id\_roles))  
 connection.commit()  
 except Exception as e:  
 print(f"Error creating user: {e}")  
  
  
def get\_user(user\_id):  
 try:  
 with connection.cursor() as cursor:  
 sql = "SELECT \* FROM Users WHERE id\_users = %s"  
 cursor.execute(sql, (user\_id,))  
 return cursor.fetchone()  
 except Exception as e:  
 print(f"Error fetching user: {e}")  
  
 def create\_wallet(name\_wallet, balance, user\_id, coin\_id):  
 try:  
 with connection.cursor() as cursor:  
 sql = """  
 INSERT INTO Wallet\_users (name\_wallet, balance, id\_users, id\_coins)  
 VALUES (%s, %s, %s, %s)  
 """  
 cursor.execute(sql, (name\_wallet, balance, user\_id, coin\_id))  
 connection.commit()  
 except Exception as e:  
 print(f"Error creating wallet: {e}")  
  
 def get\_wallet(wallet\_id):  
 try:  
 with connection.cursor() as cursor:  
 sql = "SELECT \* FROM Wallet\_users WHERE id\_wallet = %s"  
 cursor.execute(sql, (wallet\_id,))  
 return cursor.fetchone()  
 except Exception as e:  
 print(f"Error fetching wallet: {e}")  
  
  
 def create\_transaction(coin\_id, quantity\_coins, price, portfolio\_selection\_id, course\_id):  
 try:  
 with connection.cursor() as cursor:  
 sql = """  
 INSERT INTO user\_transactions (id\_coins, quantity\_coins, price, id\_portfolio\_selection, id\_course)  
 VALUES (%s, %s, %s, %s, %s)  
 """  
 cursor.execute(sql, (coin\_id, quantity\_coins, price, portfolio\_selection\_id, course\_id))  
 connection.commit()  
 except Exception as e:  
 print(f"Error creating transaction: {e}")  
  
 def get\_transactions(wallet\_id):  
 try:  
 with connection.cursor() as cursor:  
 sql = "SELECT \* FROM user\_transactions WHERE id = %s"  
 cursor.execute(sql, (wallet\_id,))  
 return cursor.fetchall()  
 except Exception as e:  
 print(f"Error fetching transactions: {e}")  
  
  
 def authenticate\_user(email, password):  
 try:  
 with connection.cursor() as cursor:  
 sql = "SELECT \* FROM Users WHERE email = %s AND id\_user\_password\_history = (SELECT id\_history FROM User\_password\_history WHERE password = %s)"  
 cursor.execute(sql, (email, password))  
 return cursor.fetchone() is not None  
 except Exception as e:  
 log\_error(f"Error authenticating user: {e}")  
 return False*

*import sys  
import requests  
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QMessageBox  
from PyQt5.uic import loadUi  
from dashboard import PriceQuotation  
class Aut(QMainWindow):  
 def \_\_init\_\_(self):  
 super(Aut, self).\_\_init\_\_()  
 loadUi("../ui\_forms/Authorization.ui",self)  
 self.pushButton.clicked.connect(self.aut)  
  
 def aut (self):  
  
 login = self.lineEdit.text()  
 password = self.lineEdit\_2.text()  
  
 try:  
 query = "SELECT id, email,password FROM Users WHERE email=%s AND password=%s"  
 self.db.cursor.execute(query,(login,password))  
 result = self.db.cursor.fetchone()  
  
 if result:  
 self.user\_id =result[0]  
 self.close()  
 self.window = PriceQuotation(self.user\_id)  
 self.window.show()  
  
 else:  
 QMessageBox.critical(self, "Ошибка", "Неверный пароль или логин")  
  
 except Exception as e:  
 QMessageBox.critical(self, "Ошибка" f" Произошла ошибка{str(e)}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = Aut()  
 window.show()  
 sys.exit(app.exec())*

*import pymysql  
  
class connection():  
 def \_\_init\_\_(self,host,user,password,database):  
 self.connection = pymysql.connect(host=host, user=user, password=password, database=database)  
 self.cursor = self.connection.cursor()*