

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет инженерно-экономический
Кафедра экономической информатики
Дисциплина «Программирование сетевых приложений»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсовой работы
Ассистент кафедры ЭИ
_____._____.2025 Ю.В.Сильванович

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:
**«ПРОГРАММНОЕ СРЕДСТВО ФОРМИРОВАНИЯ
ИНВЕСТИЦИОННОГО ПОРТФЕЛЯ КРИПТОВАЛЮТ НА ОСНОВЕ
МОДЕЛИ МАРКОВИЦА»**

БГУИР КР 6-05-0611-01 056 ПЗ

Выполнил студент группы 372303
Давидюк Максим Викторович

(подпись студента)
Курсовой проект представлен на
проверку _____._____.2025

(подпись студента)

Минск 2025

РЕФЕРАТ

БГУИР КР 6-05-0611-0 056 ПЗ

Давидюк, М.В. Программное средство формирования инвестиционного портфеля криптовалют на основе модели Марковица/ М.В. Давидюк. – Минск: БГУИР, 2025. – 51с.

Пояснительная записка 51 с., 42 рис., 10 источников, 3 приложения

ФОРМИРОВАНИЯ ИНВЕСТИЦИОННОГО ПОРТФЕЛЯ КРИПТОВАЛЮТ НА ОСНОВЕ МОДЕЛИ МАРКОВИЦА С МОДУЛЕМ АНАЛИТИКИ, МОДЕЛИ *UML*, *IDEF0*, *BPMN*, СХЕМЫ АЛГОРИТМОВ, ПРОГРАММНОЕ СРЕДСТВО

Цель курсовой работы: автоматизировать процесс формирования оптимального криптовалютного портфеля на основе модели Марковица. Разработанное программное средство повысит эффективность принятия решений за счет использования методов математического моделирования.

Методология проведения работы: в процессе решения поставленных задач использованы принципы системного подхода, аналитические методы, методы компьютерной обработки данных и математического моделирования. Также выполнена визуализация результатов анализа.

Результаты работы: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработаны модели бизнес-процессов предметной области на основе нотаций *IDEF0*; описаны основные алгоритмы работы программного средства; разработана информационная модель системы, представленная в виде схемы базы данных; реализован программный продукт с возможностью анализа рисков и оценки доходности портфеля; проведено тестирование программного средства, подтвердившее его соответствие функциональным требованиям.

Программный продукт разработан на языке *Java* с применением *IntelliJ IDEA Ultimate Edition 2024.3* и СУБД *PostgreSQL*.

Область применения результатов: разработанное программное средство может использоваться инвесторами, аналитиками и финансовыми консультантами для формирования и оптимизации инвестиционного портфеля криптовалют, также помогать принимать решения в области криптовалютных инвестиций.

СОДЕРЖАНИЕ

Введение.....	5
1 Анализ литературных источников и программных решений	6
1.1 Описание предметной области	6
1.2 Обзор программных аналогов	8
2 Моделирование предметной области и разработка требований к программному средству.....	12
2.1 Анализ и формализация бизнес-процессов предметной области	12
2.2 Анализ требований к программному средству и разработка их спецификации	17
2.3 Образ предлагаемого решения	19
3 Проектирование и разработка программного средства	22
3.1 Архитектурные решения и технологии реализации программного средства.....	22
3.2 Проектирование и разработка пользовательского интерфейса.....	27
3.3 Разработка модели данных.....	30
3.4 Описание статических и динамических аспектов поведения программных объектов	34
3.5 Разработка и описание алгоритмов, реализующих бизнес-логику программного средства	40
3.6 Механизмы обеспечения информационной безопасности	43
4 Тестирование и проверка работоспособности программного средства.....	44
5 Руководство по установке (развертыванию) и использованию программного средства.....	47
5.1 Руководство по установке (развертыванию) программного средства ..	47
5.2 Руководство пользователя.....	47
Заключение	48
Список использованных источников	49
Приложение А (обязательное) Отчет о проверке на заимствование в системе «Антиплагиат».....	50
Приложение Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику	51
Приложение В (обязательное) Листинг SQL-скрипта	54
Приложение Г (обязательное) Схемы алгоритмов	56

ВВЕДЕНИЕ

В современном мире криптовалюты становятся популярным инструментом для инвестиций, привлекая частных инвесторов и финансовые организации. Высокая волатильность рынка создаёт риски, требуя научно обоснованных подходов, таких как модель Марковица. Эта модель оптимизирует соотношение доходности и риска через анализ ожидаемой доходности, дисперсии и корреляции активов. Разработка программного средства для автоматизации формирования инвестиционного портфеля криптовалют на основе модели Марковица — актуальная задача, направленная на повышение эффективности управления инвестициями и снижение рисков.

Актуальность темы обусловлена ростом интереса к криптовалютам и отсутствием централизованных решений, интегрирующих модель Марковица для управления портфелями. Такое средство позволит автоматизировать расчёты, минимизировать ошибки и предоставить удобный интерфейс для анализа.

Целью курсового проекта является разработка клиент-серверного приложения, автоматизирующего формирование оптимального инвестиционного портфеля криптовалют с использованием модели Марковица.

Для достижения поставленной цели необходимо решить следующие задачи:

- описать предметную область и провести анализ существующих программных решений;
- проанализировать изучаемые процессы;
- разработать UML и IDEF0 диаграммы;
- спроектировать архитектуру приложения, пользовательский интерфейс и модель данных;
- разработать программное средство с учётом требований к безопасности и производительности;
- провести тестирование системы и подготовить руководство по её установке и использованию.

Объектом исследования выступает процесс формирования инвестиционного портфеля криптовалют, а предметом — программное средство, реализующее автоматизацию этого процесса на основе модели Марковица. Разработанное приложение будет ориентировано на инвесторов и аналитиков, предоставляя им инструменты для эффективного управления активами на криптовалютном рынке.

1 АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ПРОГРАММНЫХ РЕШЕНИЙ

1.1 Описание предметной области

В современном мире криптовалюты становятся все более популярным инструментом для инвестиций, привлекая как частных инвесторов, так и крупные финансовые организации. Однако высокая волатильность криптовалютного рынка создает значительные риски, что требует от инвесторов применения научно обоснованных подходов к формированию инвестиционных портфелей. Одним из таких подходов является модель Марковица, также известная как теория портфеля, которая позволяет оптимизировать соотношение доходности и риска в инвестициях.

Модель Марковица была предложена Гарри Марковицем в 1952 году и с тех пор стала фундаментальной основой для управления финансовыми активами. Согласно этой модели, инвесторы стремятся максимизировать ожидаемую доходность при заданном уровне риска или минимизировать риск при фиксированной доходности. Для этого необходимо учитывать не только доходность и риск каждого отдельного актива, но и их корреляцию с другими активами в портфеле. Таким образом, диверсификация становится ключевым элементом успешной стратегии инвестирования [1].

Криптовалютный рынок характеризуется уникальными особенностями, которые делают применение модели Марковица особенно актуальным. Во-первых, криптовалюты демонстрируют высокую волатильность, что увеличивает как потенциальную доходность, так и уровень риска. Во-вторых, корреляция между различными криптовалютами может значительно варьироваться, что открывает возможности для эффективной диверсификации. В-третьих, быстрое развитие технологий и появление новых цифровых активов создают динамичную среду, требующую постоянного пересмотра стратегий инвестирования.

Процесс формирования инвестиционного портфеля криптовалют на основе модели Марковица можно условно разделить на несколько этапов. Первый этап – сбор данных. Необходимо получить исторические данные о ценах выбранных криптовалют за определенный период времени. Эти данные используются для расчета ожидаемой доходности и стандартного отклонения (меры риска) каждого актива, а также для определения корреляции между ними. Источниками данных могут служить криптобиржи, такие как Binance, Bybit или Coinbase, а также специализированные API-сервисы.

Второй этап – расчет параметров портфеля. На этом этапе применяются математические методы, такие как оптимизация с использованием квадратичного программирования, чтобы найти оптимальное распределение средств между активами. Результатом является так называемая "эффективная граница" – набор портфелей, обеспечивающих максимальную доходность при

каждом уровне риска. Инвестор может выбрать портфель, соответствующий его аппетиту к риску.

Третий этап – мониторинг и перебалансировка портфеля. Криптовалютный рынок характеризуется высокой изменчивостью, поэтому регулярное обновление данных и корректировка портфеля являются необходимыми условиями для поддержания его оптимальности. Это может включать продажу переоцененных активов и покупку недооцененных, чтобы вернуть портфель к целевому распределению.

Разработка программного средства для автоматизации этих процессов имеет множество преимуществ. Во-первых, автоматизация расчетов значительно снижает вероятность ошибок, связанных с ручной обработкой данных. Это особенно важно в условиях высокой волатильности криптовалютного рынка, где даже небольшие погрешности могут привести к значительным потерям. Во-вторых, программное средство позволяет быстро анализировать большие объемы данных, что существенно ускоряет процесс принятия решений.

В-третьих, наличие аналитического модуля дает возможность глубже исследовать поведение рынка и выявлять скрытые закономерности. Например, инвестор может проанализировать, как изменение макроэкономических факторов влияет на корреляцию между криптовалютами, или оценить влияние новостей на динамику цен. Это помогает принимать более обоснованные решения и адаптироваться к меняющимся рыночным условиям.

Четвертым преимуществом является возможность гибкой настройки системы под конкретные цели и ограничения инвестора. Программное средство может учитывать такие параметры, как минимальный и максимальный размер инвестиций в каждый актив, ограничения по уровню риска или требования к ликвидности портфеля. Это делает его универсальным инструментом для широкого круга пользователей – от начинающих трейдеров до профессиональных управляющих активами.

Наконец, внедрение такого программного обеспечения способствует повышению конкурентоспособности инвесторов на криптовалютном рынке. Современные технологии позволяют не только автоматизировать рутинные процессы, но и предоставляют доступ к передовым методам анализа данных, что создает дополнительные преимущества перед конкурентами.

Таким образом, разработка программного средства для формирования инвестиционного портфеля криптовалют на основе модели Марковица представляет собой важную и актуальную задачу. Она позволяет инвесторам эффективно управлять рисками, максимизировать доходность и адаптироваться к динамичным условиям криптовалютного рынка. Эти преимущества делают систему незаменимым инструментом для успешного инвестирования в цифровые активы.

1.2 Обзор программных аналогов

В современном мире управления инвестициями ключевым фактором успеха является использование передовых технологий и программных решений для анализа и оптимизации портфелей. Особенно это актуально для криптовалютного рынка, который характеризуется высокой волатильностью и динамичностью. Программные средства, разработанные для управления криптовалютными активами, позволяют не только автоматизировать процессы отслеживания цен и расчетов доходности, но и предоставляют инструменты для минимизации рисков и максимизации прибыли. Такие системы способствуют более эффективному управлению инвестиционными ресурсами, обеспечивают прозрачность данных и помогают инвесторам принимать обоснованные решения на основе актуальной информации. Однако выбор подходящего программного решения зависит от конкретных задач инвестора, таких как уровень контроля над портфелем, возможность интеграции с биржами и поддержка сложных моделей оптимизации, например, модели Марковица.

Рассмотрим такой программный аналог как «Portfolio Performance». «Portfolio Performance» – это открытая платформа для управления финансовыми портфелями, которая поддерживает как традиционные акции и облигации, так и криптовалюты. Она особенно популярна среди инвесторов, предпочитающих гибкость и контроль над своими данными [2].

Рассмотренная программа предоставляет следующие основные функции:

- расчет доходности и рисков портфеля с использованием различных методов;
- импорт данных из CSV-файлов или API-интерфейсов;
- поддержка пользовательских метрик для анализа портфеля;
- создание графиков и диаграмм для визуализации данных;
- интеграция с моделями оптимизации, такими как модель Марковица;
- экспорт отчетов в различные форматы для дальнейшего анализа.

На рисунке 1.1 представлен интерфейс программы «Portfolio Performance».

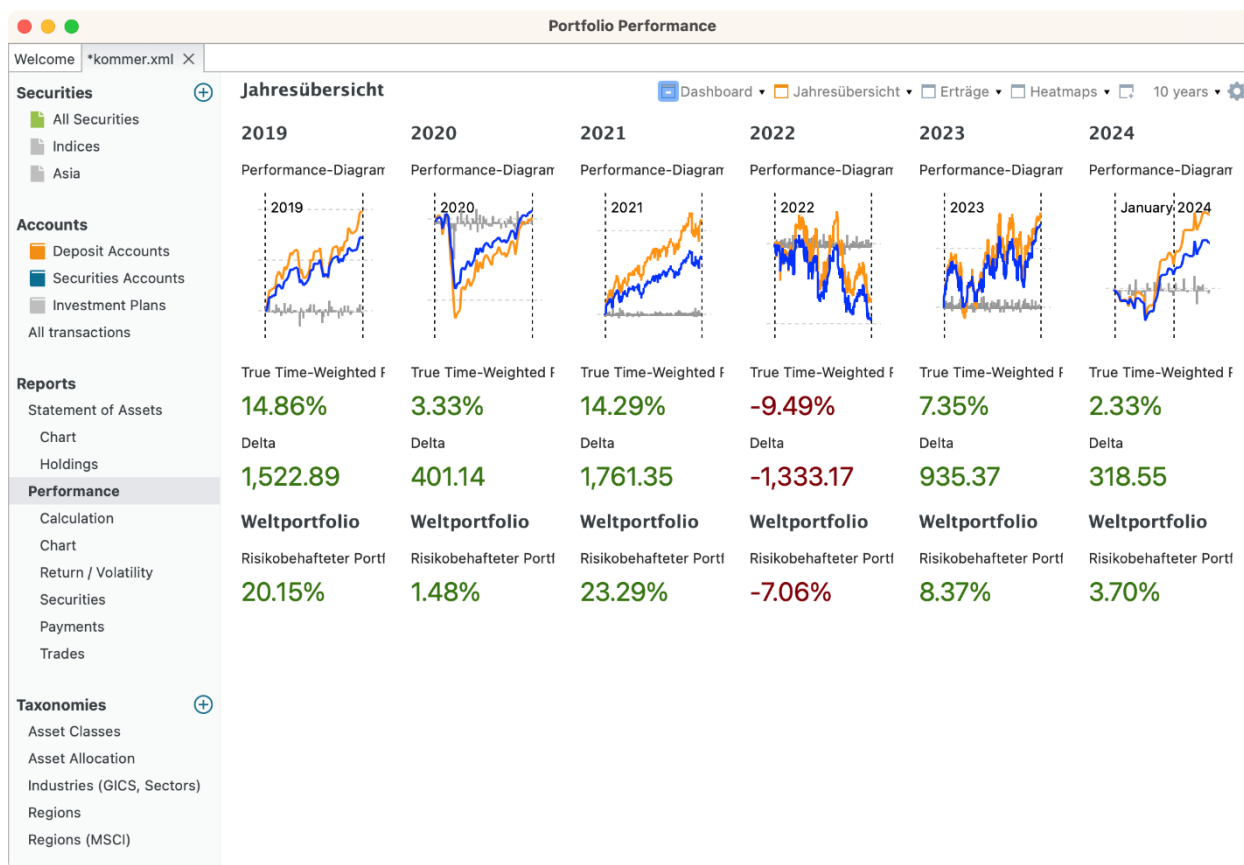


Рисунок 1.1 – Интерфейс «Portfolio Performance»

«Portfolio Performance» является наиболее подходящим аналогом, так как позволяет интегрировать пользовательские модели оптимизации, включая модель Марковица. Это делает её идеальным выбором для инвесторов, стремящихся к научно обоснованному управлению портфелем.

Теперь рассмотрим такую программу как «CoinStats». Это многофункциональное приложение для отслеживания криптовалютных портфелей, которое предлагает инструменты для анализа и управления активами. Программа поддерживает интеграцию с более чем 300 криптобиржами и кошельками, что делает её универсальным решением для инвесторов [3].

Данная программа предоставляет следующие основные функции:

- отслеживание портфеля в реальном времени;
- анализ доходности и рисков по каждому активу;
- интеграция с популярными биржами (Binance, Coinbase, Kraken и др.);
- генерация отчетов о доходности и изменении стоимости активов;
- поддержка мультивалютного учета (криптовалюты, фиатные валюты);
- уведомления о значительных изменениях цен или событиях на рынке;
- аналитика рыночных трендов и новостей.

Интерфейс программы «CoinStats» представлен на рисунке 1.2.

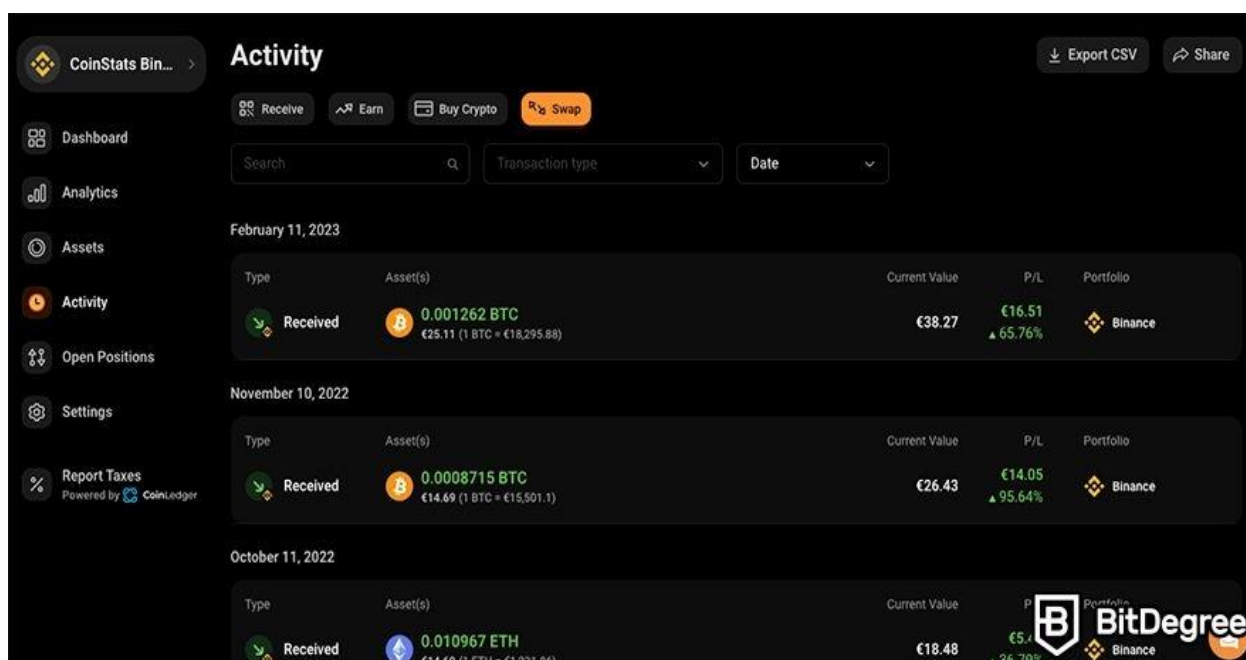


Рисунок 1.2 – Интерфейс «CoinStats»

«CoinStats» подходит для начинающих и опытных инвесторов благодаря своей простоте и широкой функциональности. Хотя она не предоставляет встроенной реализации модели Марковица, её инструменты для анализа доходности и рисков могут быть полезны для предварительной оценки портфеля.

Рассмотрим ещё один аналог «CryptoCompare». Данная платформа представляет аналитические данные о криптовалютах и их взаимосвязях. Хотя она не является полноценным инструментом для управления портфелем, её данные могут быть использованы для создания собственных моделей оптимизации. На рисунке 1.3 изображен интерфейс платформы «CryptoCompare» [4].

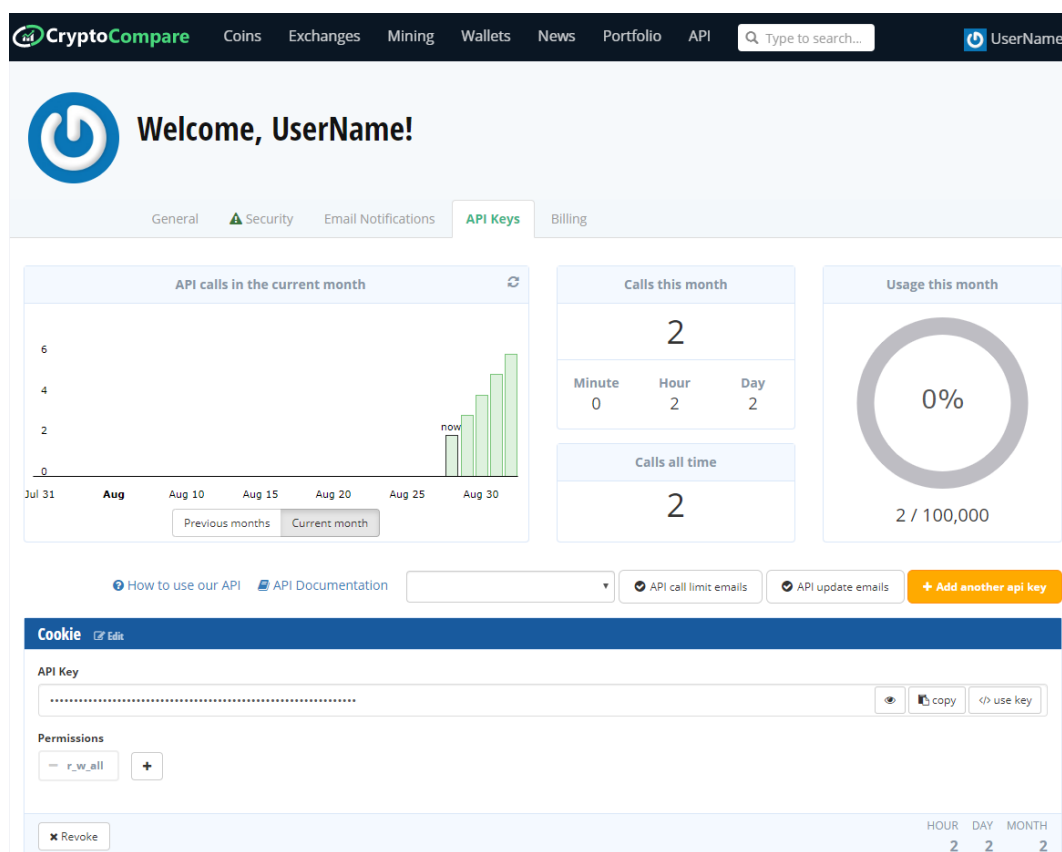


Рисунок 1.3 – Интерфейс «CryptoCompare»

Основные функции «CryptoCompare» включают:

- детальный анализ корреляций между криптовалютами;
- сбор исторических данных о ценах и объемах торгов;
- предоставление API для интеграции с другими системами;
- генерация отчетов о рыночных трендах;
- инструменты для сравнения производительности различных активов.

«CryptoCompare» является отличным выбором для инвесторов, которые хотят глубже понять взаимосвязи между активами и использовать эти данные для построения оптимального портфеля на основе модели Марковица.

Таким образом, рассмотренные программы представляют собой наиболее подходящие решения для управления криптовалютными портфелями. Однако ни одна из них не предоставляет полной реализации модели Марковица как основного механизма оптимизации портфеля. Это создает пространство для разработки нового программного средства, которое объединит преимущества существующих решений с научно обоснованными подходами к управлению рисками и доходностью.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ

2.1 Анализ и формализация бизнес-процессов предметной области

Функциональное моделирование по стандарту IDEF0 позволяет анализировать и визуализировать бизнес-процессы, детализируя работу системы и взаимосвязь ее элементов. В контексте автоматизированной системы формирования инвестиционного портфеля криптовалют данный метод необходим для описания ключевых операций, их входных и выходных данных, а также механизмов реализации [5].

Основным процессом курсового проекта является формирование оптимального инвестиционного портфеля. Этот процесс включает последовательные этапы от сбора и анализа исходных данных до расчета ключевых показателей, выявления ограничений и формирования итоговой структуры портфеля, максимально соответствующей целям инвестора.

На рисунке 2.1 представлена контекстная диаграмма процесса «Сформировать оптимальный инвестиционный портфель». Этот процесс начинается с анализа данных о криптовалютах, учета рыночных ограничений и инвестиционных целей, таких как желаемая доходность и допустимый уровень риска. Управление процессом осуществляется через регуляторные нормы для криптоактивов в Республике Беларусь, принятую «Инвестиционную стратегию» и международный стандарт ISO 8000, обеспечивающий качество данных. Для выполнения задач задействуются «Бизнес-аналитик», аналитическое программное обеспечение, системы отчетности и визуализации, а также модель оптимизации. В результате создаются «Оптимальный инвестиционный портфель», графики риска и доходности, «Инвестиционные рекомендации» и «Отчет о составе портфеля», которые служат основой для принятия решений.

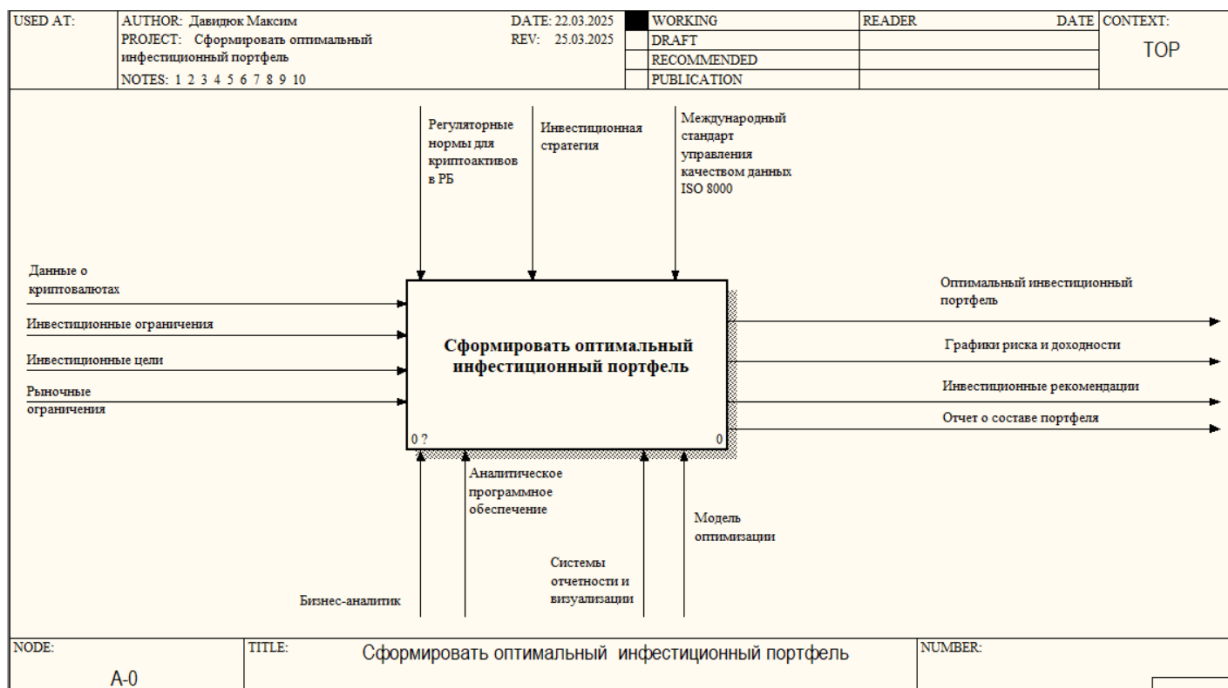


Рисунок 2.1 – Контекстная диаграмма процесса формирования оптимального инвестиционного портфеля

Далее рассмотрим декомпозицию процесса «Сформировать оптимальный инвестиционный портфель», представленную на рисунке 2.2. В данном случае работа делится на четыре этапа: «Сбор данных о криптовалютах», «Определить перечень возможных активов», «Провести анализ рисков и доходностей активов» и «Сформировать итоговый портфель». На каждом этапе используются соответствующие механизмы и управления. Например, при сборе данных ключевую роль играет аналитическое программное обеспечение, а при формировании портфеля — модель оптимизации. После завершения всех этапов формируется окончательный результат, включающий оптимальный портфель и сопутствующие документы.

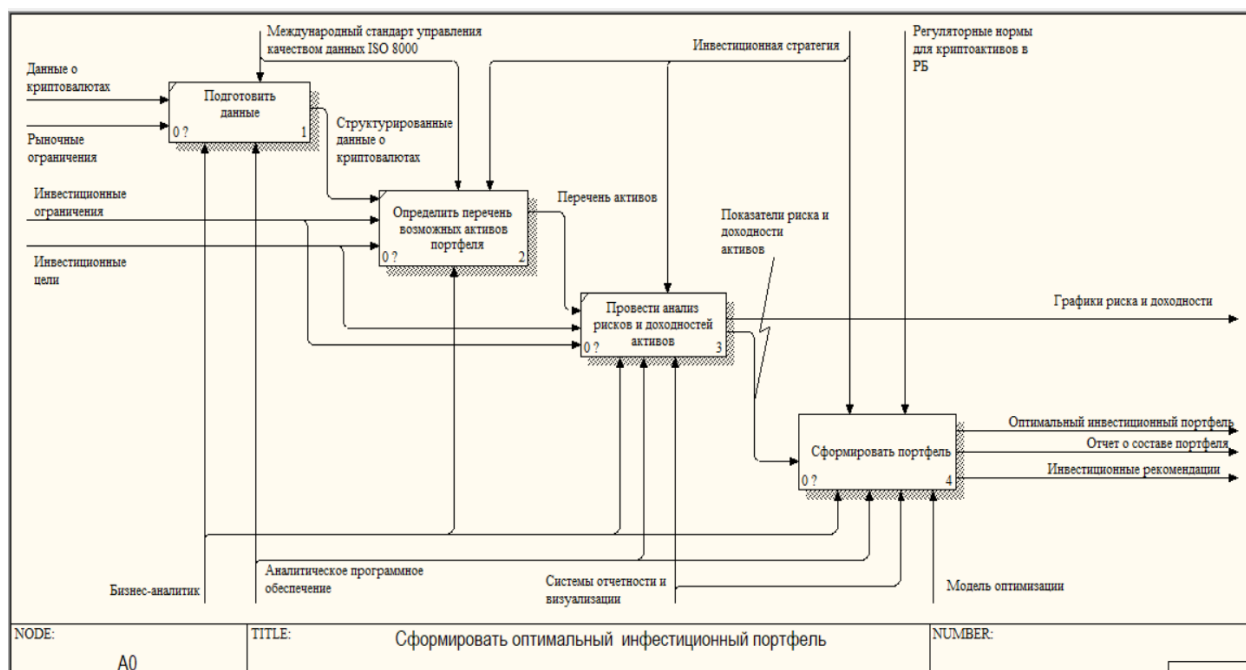


Рисунок 2.2 – Декомпозиция контекстной диаграммы

На рисунке 2.3 представлена декомпозиция процесса «Сформировать оптимальный портфель». Работа разбивается на пять этапов: «Определить целевые параметры портфеля», «Рассчитать границу эффективности», «Оптимизировать структуру портфеля», «Проверить устойчивость к изменениям рынка» и «Выполнить отчет о составе портфеля». На этом уровне особое внимание уделяется взаимодействию между этапами, где выход одного процесса становится входом для следующего. Например, результаты расчета границы эффективности используются для оптимизации распределения активов. По завершении всех этапов формируются итоговые документы, такие как «Инвестиционные рекомендации» и «Отчет о составе портфеля».

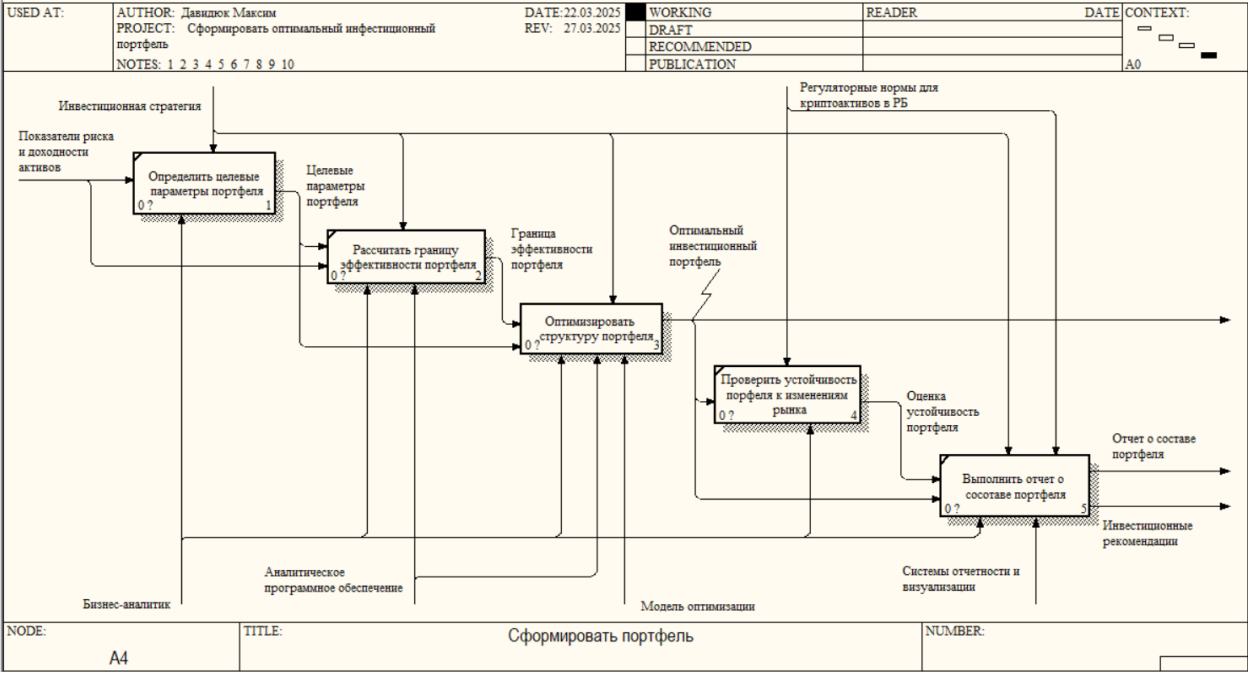


Рисунок 2.3 – Декомпозиции процесса «Сформировать портфель»

На рисунке 2.4 описана оптимизация структуры портфеля. Процесс разбивается на следующие блоки: «Сформулировать цель оптимизации», «Определить ограничения оптимизации» и «Оптимизировать распределение активов». Управление строится на базе инвестиционной стратегии, а механизмы включают аналитиков, модели и программное обеспечение.

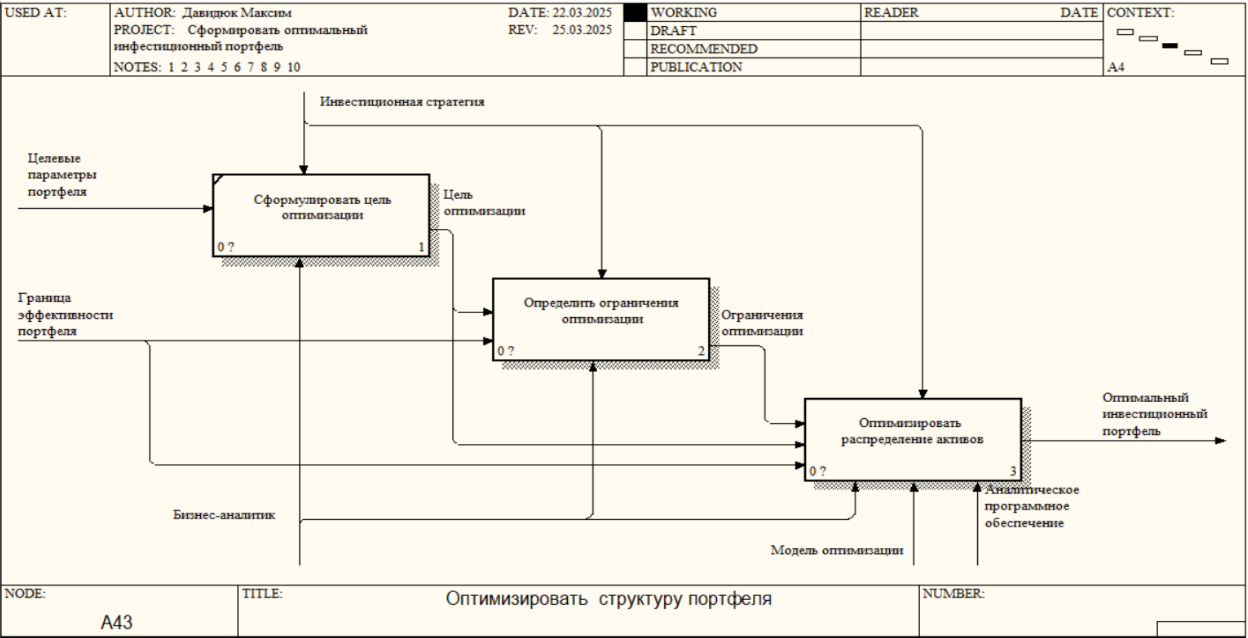


Рисунок 2.4 – Декомпозиция процесса «Оптимизировать структуру портфеля»

Наконец, на рисунке 2.5 показана декомпозиция процесса «Оптимизировать распределение активов». Работа включает три этапа: «Выполнить предварительный расчет», «Проверить соответствие заданным критериям» и «Сформировать окончательный портфель». Каждый этап контролируется «Инвестиционной стратегией» и подкрепляется профессиональной экспертизой бизнес-аналитика. После успешного завершения всех шагов формируется «Оптимальный инвестиционный портфель», готовый к использованию.

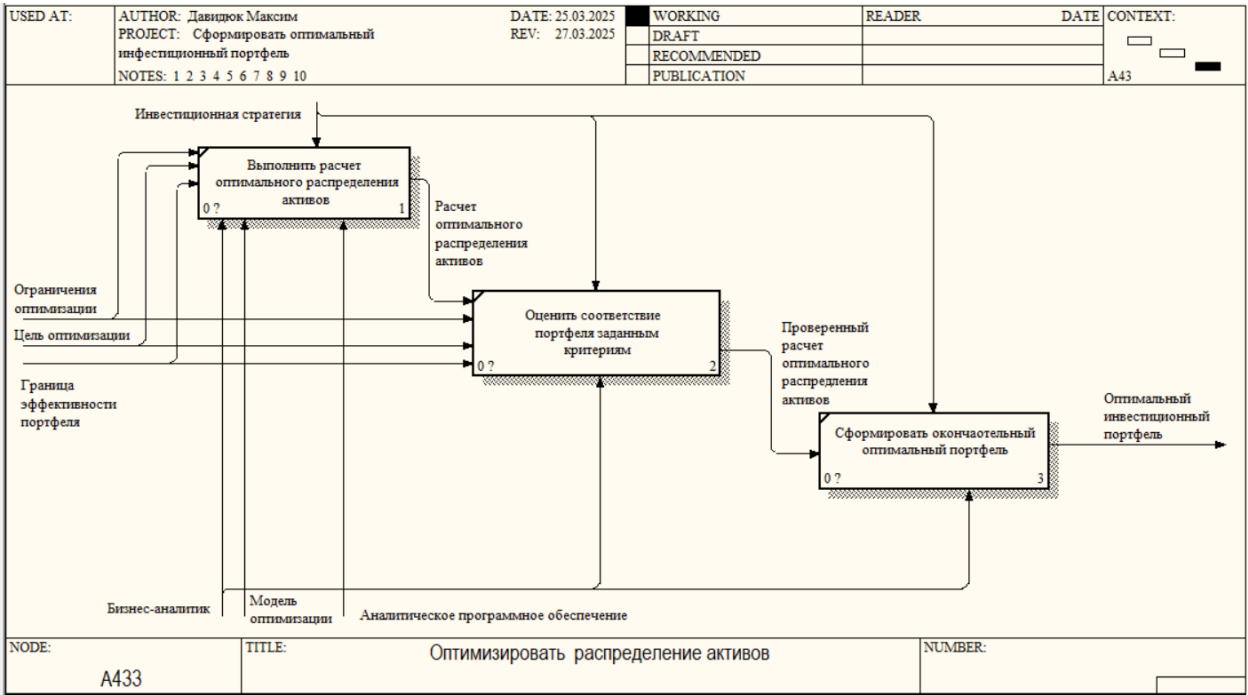


Рисунок 2.5 – Декомпозиции процесса «Оптимизировать распределение активов»

Таким образом, построение функциональной модели процесса «Сформировать оптимальный инвестиционный портфель» позволяет получить четкое представление о последовательности этапов, необходимых для достижения цели. Эта модель помогает не только определить ключевые шаги, но и обеспечить их логическую связь, что способствует повышению эффективности и точности принимаемых решений.

2.2 Анализ требований к программному средству и разработка их спецификации

Анализ требований к программному средству является ключевым этапом разработки системы управления портфелями криптовалют, так как он позволяет определить функциональные и нефункциональные потребности, обеспечивая соответствие системы ожиданиям пользователей. Для структурирования требований и наглядного представления взаимодействия пользователей с системой используется диаграмма вариантов использования (Use Case Diagram), которая является частью методологии UML (Unified Modeling Language). Диаграмма вариантов использования позволяет выделить основные роли пользователей, их цели и действия в системе, а также определить границы функциональности программного средства. Использование таких диаграмм способствует повышению прозрачности требований и упрощению коммуникации между разработчиками и заинтересованными сторонами, что делает их незаменимым инструментом в процессе проектирования [6].

Системой могут пользоваться два типа пользователей: пользователь, который представляет собой инвестора или аналитика, и администратор, отвечающий за техническое сопровождение и управление данными. Для каждой из этих ролей определён свой набор действий, которые они могут выполнять внутри системы, что отражено в диаграмме вариантов использования.

Варианты использования, доступные для пользователя, включают:

- регистрация в системе;
- авторизация в личном кабинете;
- загрузка ранее сохраненного портфеля;
- сохранение портфеля;
- выбор криптовалют для анализа;
- расчет ожидаемой доходности;
- расчет риска;
- построение эффективной границы;
- просмотр графиков цен криптовалют;
- загрузка ранее сохраненного портфеля;
- создание отчета.

Администратор наследует все варианты использования пользователя, а также определяет дополнительный набор функций:

- управление пользователями;
- управление списком криптовалют;
- назначение роли пользователю.

На рисунке 2.6 представлена диаграмма вариантов использования, которая демонстрирует все функциональные возможности системы и показывает взаимодействие между пользователями и приложением. С

помощью этой диаграммы можно понять, какие именно действия могут выполнять участники системы и как они связаны с её компонентами.

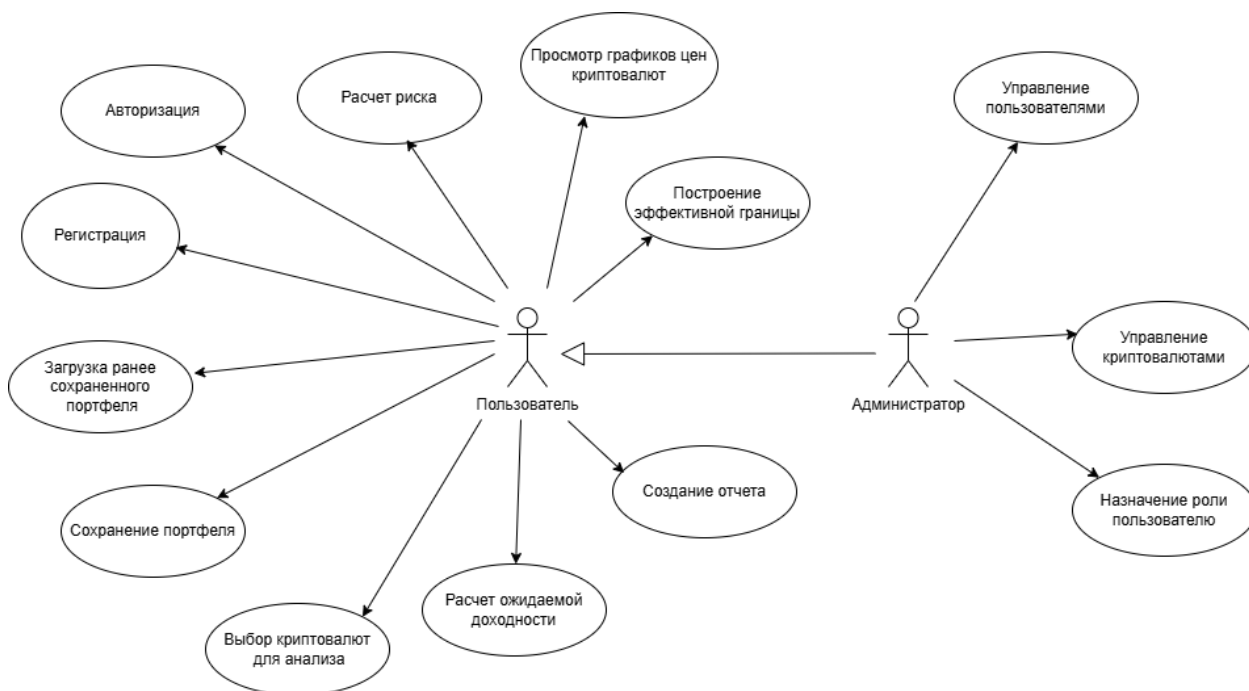


Рисунок 2.6 – Диаграмма вариантов использования

Диаграмма позволяет заранее структурировать функциональные требования, выявить потенциальные проблемы взаимодействия и правильно распределить ответственность между различными ролями. Это особенно важно при разработке сложных финансовых приложений, где каждая операция требует высокой точности и чёткой организации.

Таким образом, применение диаграммы вариантов использования способствует более грамотному проектированию программного средства, снижает вероятность возникновения ошибок на ранних этапах разработки и помогает создать понятное и логически выстроенное приложение. Особенно это актуально в случае создания системы управления инвестиционным портфелем на рынке криптовалют, где требуется учитывать множество факторов и обеспечивать максимальную прозрачность действий пользователя.

2.3 Образ предлагаемого решения

Современные инвестиционные процессы в сфере криптовалют требуют высокой скорости принятия решений, гибкости анализа и автоматизации рутинных операций. Однако в условиях отсутствия единой программной платформы участники инвестиционной деятельности сталкиваются с рядом трудностей — от ограниченного доступа к данным до необходимости ручной обработки информации. Это существенно снижает эффективность управления инвестиционными портфелями и увеличивает нагрузку на административный персонал.

В результате проведенного анализа были выявлены следующие ключевые бизнес-проблемы:

- отсутствие централизованного программного инструмента, обеспечивающего эффективное взаимодействие между инвесторами и аналитиками, что затрудняет координацию процессов управления портфелями;

- ограниченная возможность инвесторов и аналитиков самостоятельно получать доступ к данным о портфелях, аналитических показателях и истории цен криптовалют, что требует постоянного обращения к техническому специалисту;

- отсутствие автоматизированного механизма для подачи запросов на обновление данных или корректировку портфелей, что замедляет принятие инвестиционных решений;

- недостаток инструментов для анализа и фильтрации данных о криптовалютах и портфелях, ограничивая возможности инвесторов и аналитиков в оценке рисков и доходности;

- необходимость ручной обработки данных и подготовки аналитических отчетов техническим специалистом, что увеличивает вероятность ошибок и задержек;

- отсутствие автоматизированной системы распределения ролей и прав доступа, что создает риски несанкционированного доступа к конфиденциальной информации;

- отсутствие средств для автоматического построения графиков и диаграмм, затрудняющих визуальную оценку эффективности портфелей.

Для решения этих проблем предлагается разработка программного средства, предназначенного для автоматизации формирования и управления инвестиционными портфелями криптовалют на основе модели Марковица. Система будет ориентирована на две категории пользователей: инвесторов и аналитиков, выполняющих операции с портфелями, и технических специалистов, обеспечивающих техническую поддержку. В зависимости от роли предоставляется соответствующий доступ к функционалу, что обеспечивает как гибкость, так и безопасность работы с данными. Внедрение данного программного средства обещает повысить эффективность управления инвестициями, минимизировать время на ручные операции и снизить нагрузку

на технических специалистов, подготавливая основу для детального анализа требований в последующих разделах.

Программное средство формирования инвестиционного портфеля криптовалют на основе модели Марковица предназначено для автоматизации процессов создания, анализа и оптимизации портфелей, а также обеспечения безопасного управления доступом к данным. Разработка реализуется в виде клиент-серверного приложения на языке Java, с использованием оконного интерфейса (JavaFX) для клиентской части и многопоточной архитектуры сервера. Система включает серверное приложение, клиентский интерфейс и базу данных под управлением СУБД PostgreSQL, обеспечивая надежное хранение и обработку информации. Взаимодействие между пользователями разных ролей (инвесторы/аналитики, технические специалисты) будет организовано через централизованную систему авторизации и ролевого доступа, что станет предметом более детального рассмотрения в спецификации требований.

Программное средство формирования инвестиционного портфеля криптовалют с модулем аналитики создается с учетом следующих целей:

- автоматизация процессов формирования и оптимизации портфелей криптовалют с использованием модели Марковица для повышения точности расчетов;
- обеспечение удобного доступа инвесторов и аналитиков к данным о портфелях, аналитике и истории цен, что станет основой для их взаимодействия с системой;
- упрощение взаимодействия между инвесторами, аналитиками и техническими специалистами, минимизируя зависимость от ручного управления;
- предоставление инструментов для анализа рисков и доходности портфелей, которые будут интегрированы в функционал системы;
- обеспечение гибкого управления доступом к информации в зависимости от роли пользователя, что будет детализировано в модели прав доступа.

Для достижения этих целей необходимо разработать программное средство, решающее следующие задачи:

- автоматизация расчета рисков и доходности портфелей с учетом корреляции между криптовалютами, что будет основано на математическом моделировании;
- предоставление пользователям доступа к данным портфелей, аналитическим показателям и истории цен через интуитивный интерфейс;
- упрощение подачи запросов на обновление данных или корректировку портфелей, что улучшит оперативность работы;
- фильтрация и сортировка данных о криптовалютах и портфелях по ключевым параметрам, что облегчит анализ;

- формирование аналитических отчетов и визуализаций (графики, диаграммы) для оценки эффективности портфелей, что будет реализовано в аналитическом модуле.

Анализ текущего состояния бизнес-процессов управления портфелями криптовалют выявил следующие недостатки, которые требуют устранения для создания эффективной системы:

- большое количество операций выполняется вручную, что увеличивает затраты времени и ресурсов, подчеркивая необходимость автоматизации;

- отсутствие единой централизованной системы, обеспечивающей взаимодействие между инвесторами, аналитиками и техническими специалистами, что затрудняет координацию;

- невозможность пользователей самостоятельно получать доступ к данным о портфелях и аналитике, что зависит от вмешательства технических специалистов;

- отсутствие автоматизированной системы подачи и обработки запросов на обновление данных, что замедляет процесс управления;

- нехватка инструментов анализа и визуализации данных о рисках и доходности, ограничивая возможности пользователей;

- сложность управления доступом к данным в зависимости от роли пользователя, что создает риски для безопасности.

Таким образом, выявленные бизнес-проблемы и анализ текущего состояния процессов подтвердили необходимость разработки программного средства для автоматизации управления криптовалютными портфелями. Разработка системы, ориентированной на роли пользователей и нацеленной на реализацию аналитических и управленческих функций, позволит устранить существующие недостатки, повысить производительность и упростить взаимодействие всех участников инвестиционного процесса.

3 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

3.1 Архитектурные решения и технологии реализации программного средства

Выбор архитектурных решений и технологий реализации программного средства является определяющим этапом разработки, поскольку от него напрямую зависят производительность, масштабируемость, безопасность и удобство использования системы. Для системы управления портфелями криптовалют на основе модели Марковица, которая оперирует сложными математическими расчетами, большими объемами финансовых данных и требует интуитивного интерфейса для инвесторов и аналитиков, необходимо обеспечить надежную и эффективную техническую основу. Правильно подобранные технологии позволяют автоматизировать процессы формирования портфелей, минимизировать риски ошибок и обеспечить оперативный доступ к данным, что особенно важно в условиях высокой волатильности криптовалютного рынка. В данном разделе описываются архитектурные решения, используемые технологии, а также обоснование выбора инструментов и паттернов проектирования, которые обеспечивают успешную реализацию проекта.

Для реализации программного средства формирования инвестиционного портфеля криптовалют планируется использование набора современных технологий, обеспечивающих высокую производительность, гибкость архитектуры и удобство пользовательского взаимодействия. В качестве инструмента для разработки графического интерфейса клиентской части приложения будет применяться JavaFX, обладающий широкими возможностями по созданию современных и интерактивных пользовательских интерфейсов. Это обеспечит комфортную работу инвесторов и аналитиков с системой, в том числе позволит реализовать визуализацию аналитических данных, таких как эффективная граница портфеля или динамика изменения цен криптовалют.

Для организации взаимодействия между серверной частью и системой управления базами данных будет использован Hibernate — объектно-реляционный маппер (ORM), упрощающий работу с данными за счёт преобразования объектов Java в соответствующие записи таблиц базы данных. Использование Hibernate позволит централизованно управлять операциями чтения, записи, обновления и удаления данных, а также повысит безопасность приложения за счёт использования параметризованных запросов, минимизирующих риск SQL-инъекций.

В качестве системы управления базами данных выбрана PostgreSQL, обладающая высокой надёжностью, масштабируемостью и поддержкой сложных SQL-запросов, что делает её оптимальным решением для хранения и

обработки данных, связанных с криптовалютными портфелями и пользовательской активностью.

Для построения графиков и диаграмм найдут применение библиотеки JFreeChart, которая обеспечит создание визуализаций, таких как эффективная граница портфеля и динамика цен криптовалют, интегрируясь с JavaFX и предоставляя гибкие инструменты для отображения линий, столбцов и точечных диаграмм, а также встроенные компоненты JavaFX Charts, включая LineChart и ScatterChart, которые будут использованы для отображения аналитических данных, например, графиков цен или зависимости риска и доходности.

Для получения актуальных и исторических данных о ценах криптовалют в режиме реального времени в системе будет использоваться Binance API, предоставляющий доступ к рыночной информации посредством REST- и WebSocket-интерфейсов. С помощью данного API будут получаться данные о ценах, объёмах торгов и других ключевых рыночных метриках по ведущим криптовалютам, таким как *Bitcoin*, *Ethereum* и другим. Полученная информация будет сохраняться в таблице базы данных, что обеспечит необходимую базу для расчёта параметров инвестиционного портфеля в соответствии с моделью Марковица.

Дополнительно в системе планируется использование API CoinGecko, который позволит получить расширенные исторические данные, не всегда доступные через Binance, включая редкие криптовалюты и долгосрочные ценовые ряды. Это обеспечит более полную и достоверную информацию для анализа рыночных трендов и расчета коэффициентов корреляции между активами, что критически важно при формировании сбалансированного портфеля. Интеграция с обоими API обеспечит устойчивость к отказам и повысит точность аналитики за счёт перекрёстной валидации данных из разных источников.

Для выполнения математических расчётов, необходимых при построении оптимального портфеля по модели Марковица, будет использована библиотека Apache Commons Math, предоставляющая инструменты для линейной алгебры, численной оптимизации и статистического анализа. Это позволит реализовать ключевые алгоритмы расчета ожидаемой доходности, дисперсии, ковариации и других параметров, определяющих структуру эффективного портфеля.

Обработка и преобразование данных, полученных в формате JSON от сторонних API, будет осуществляться с использованием библиотеки Gson, которая упростит сериализацию и десериализацию данных, обеспечивая преобразование входных потоков в объекты Java. Это существенно ускорит процесс интеграции с внешними сервисами и обеспечит удобную работу с полученной информацией внутри системы.

Для ведения журналов событий и отладки системы будет использоваться связка SLF4J и Logback, обеспечивающая гибкое и централизованное логирование. Это позволит отслеживать действия пользователей (например,

авторизацию, создание портфелей, выполнение расчетов), а также регистрировать ошибки и исключения, что упростит сопровождение и диагностику приложения.

Обработка временных данных, включая хранение и анализ исторических цен, будет реализована с применением Java Time API, предоставляющего мощные средства для работы с датами, временами и временными интервалами. Это особенно важно при построении временных рядов, расчёте доходности за различные периоды и синхронизации данных из внешних источников.

Система будет реализована в виде клиент-серверного приложения, архитектура которого предполагает чёткое разделение обязанностей между клиентской и серверной частями. Клиент отвечает за отображение интерфейса и взаимодействие с пользователем, а сервер — за выполнение бизнес-логики, обработку запросов, хранение и анализ данных. Такой подход обеспечивает лучшую масштабируемость, модульность и возможность интеграции с внешними источниками данных и сервисами.

Взаимодействие между клиентом и сервером будет осуществляться по сетевому протоколу TCP/IP, что гарантирует надёжную и устойчивую передачу данных в процессе работы приложения. Использование TCP (Transmission Control Protocol) позволит обеспечить целостность и последовательность передаваемых данных, что особенно важно при передаче финансовой информации и параметров инвестиционного анализа. Кроме того, данная архитектура облегчает развертывание системы в распределённых средах, включая облачные решения, а также упрощает обновление и сопровождение отдельных компонентов без необходимости полной переустановки системы у конечных пользователей.

Для модульности и расширяемости применены паттерны проектирования, адаптированные к задачам системы. Использован паттерн Singleton (Одиночка) для обеспечения единственного экземпляра класса ClientSocket, управляющего соединением с сервером, диаграмма классов которого представлена на рисунке 3.1. Код реализует инициализацию сокета и потоков ввода-вывода один раз, что предотвращает дублирование подключений и упрощает управление ресурсами.

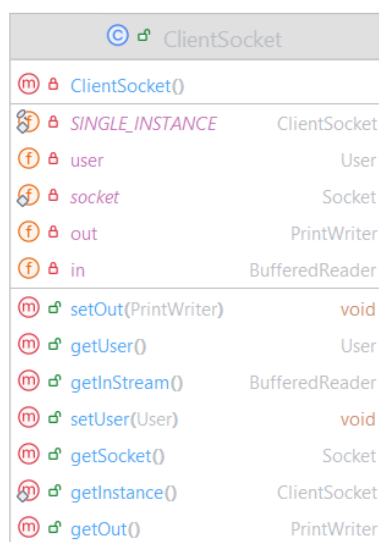


Рисунок 3.1 – Диаграмма классов паттерна «Singleton»

Паттерн Strategy (Стратегия) реализован для авторизации и регистрации, позволяя выбирать алгоритмы аутентификации в зависимости от контекста, обеспечивая гибкость и поддержку различных методов (рисунок 3.2). Эти паттерны повышают модульность и адаптивность системы.

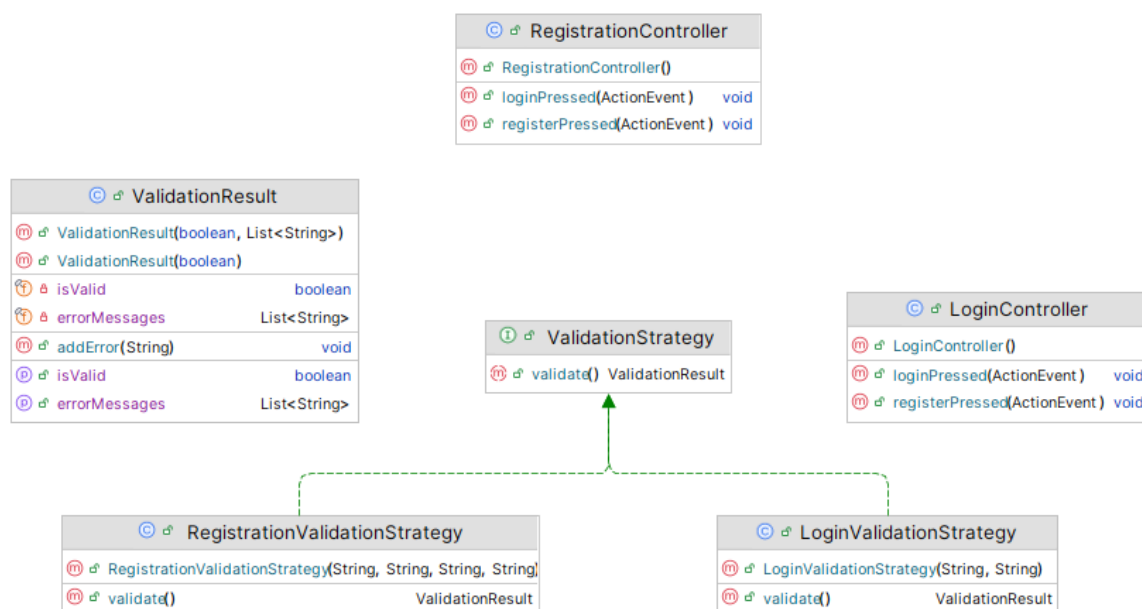


Рисунок 3.2 – Диаграмма классов паттерна «Strategy»

Таким образом, выбор архитектурных решений и технологий разработки программного средства формирования инвестиционного портфеля криптовалют обоснован спецификой задачи и требованиями к

производительности, надёжности и масштабируемости системы. Использование JavaFX для реализации графического интерфейса обеспечит удобную и наглядную работу пользователей с аналитическими инструментами. Внедрение Hibernate и PostgreSQL позволит эффективно управлять данными и обеспечит безопасность взаимодействия с базой данных. Интеграция с внешними API, такими как Binance и CoinGecko, обеспечит систему достоверной и актуальной рыночной информацией. Применение специализированных библиотек — Apache Commons Math, Gson, SLF4J и Java Time API — обеспечит точность математических расчётов, удобство обработки данных и возможность отладки на всех этапах жизненного цикла программы. В совокупности данные решения формируют прочную техническую основу, способную обеспечить корректную реализацию модели Марковица и адаптироваться к изменяющимся условиям криптовалютного рынка.

3.2 Проектирование и разработка пользовательского интерфейса

Пользовательский интерфейс представляет собой ключевой компонент программного обеспечения, определяющий удобство и эффективность взаимодействия пользователя с системой, что напрямую влияет на её практическую ценность. Он включает совокупность визуальных и функциональных элементов, обеспечивающих доступ к функциональности приложения и отображающих информационную модель предметной области. В условиях высокой сложности вычислений и необходимости визуализации аналитических данных, характерных для системы управления портфелями криптовалют на основе модели Марковица, особенно важно будет спроектировать интерфейс таким образом, чтобы он был не только функциональным, но и интуитивно понятным. Пользователи, включая инвесторов и аналитиков, должны будут легко формировать портфели, отслеживать ключевые метрики (доходность, риск, структура активов) и визуализировать результаты моделирования и анализа. Качество интерфейса станет играть решающую роль в восприятии системы конечными пользователями и обеспечит её практическую применимость в инвестиционной сфере [7].

Разработка интерфейса для программного средства формирования инвестиционного портфеля криптовалют будет вестись с акцентом на ясную навигацию, адаптивность и доступность всех функций как для опытных инвесторов, так и для пользователей без специальной подготовки. Интерфейс станет отражать сложную информационную модель системы, интегрируя данные из базы данных, такие как портфели, аналитические показатели и история цен, и предоставит пользователям инструменты для их эффективного анализа.

Для реализации визуальной части будет использован фреймворк JavaFX, который обеспечит широкий набор компонентов для построения современных графических интерфейсов и поддержит гибкое взаимодействие с данными. Визуальные элементы, такие как таблицы с данными криптовалют, графики зависимости риска и доходности, диаграммы распределения активов и кнопки для управления портфелем, будут распределены по вкладкам с логической структурой. Это создаст целостную и понятную пользовательскую среду, где каждый элемент интерфейса будет соответствовать конкретной задаче пользователя, упрощая доступ к функциям, таким как создание портфеля или просмотр аналитики.

На рисунке 3.3 представлен дизайн формы авторизации. Она включает поля для ввода логина и пароля, кнопки "Войти" и ссылку "Регистрация", которая перенаправляет на форму регистрации. Форма также отображает

уведомления об ошибках (например, "Неверный логин или пароль") и поддерживает выбор роли пользователя (инвестор, аналитик, администратор) после успешного входа, что соответствует требованиям ролевой модели системы

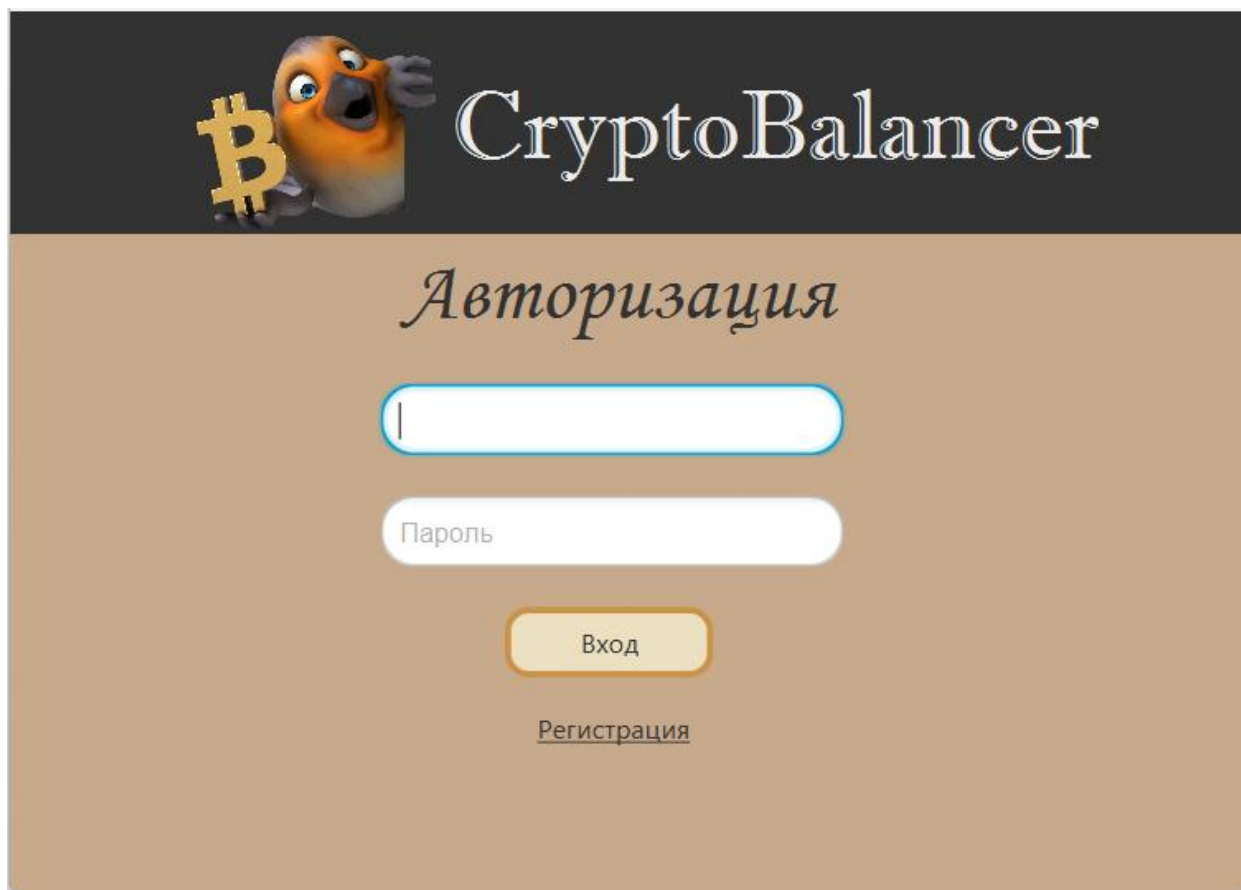
The image shows a login form for a system called "CryptoBalancer". At the top left, there is a cartoon bird holding a large Bitcoin symbol. To its right, the text "CryptoBalancer" is displayed in a stylized, serif font. Below this header, the word "Авторизация" (Authorization) is written in a cursive script. The form consists of two input fields: the first is empty, and the second is labeled "Пароль" (Password). Below the password field is a yellow button with the text "Вход" (Login). At the bottom, there is a link labeled "Регистрация" (Registration) in a blue, underlined font.

Рисунок 3.3 – Дизайн формы авторизации

На рисунке 3.4 изображён дизайн формы для управления портфелями и их аналитики, которая предоставляет пользователю удобные функции для работы со всеми аспектами формирования портфелей. Форма включает список с данными о текущих портфелях и кнопки для удаления, создания новых и формирования отчетов портфелей, а также область для отображения данных портфеля. Кнопка для просмотра анализа позволяет открыть диалоговое окно с аналитикой портфеля.

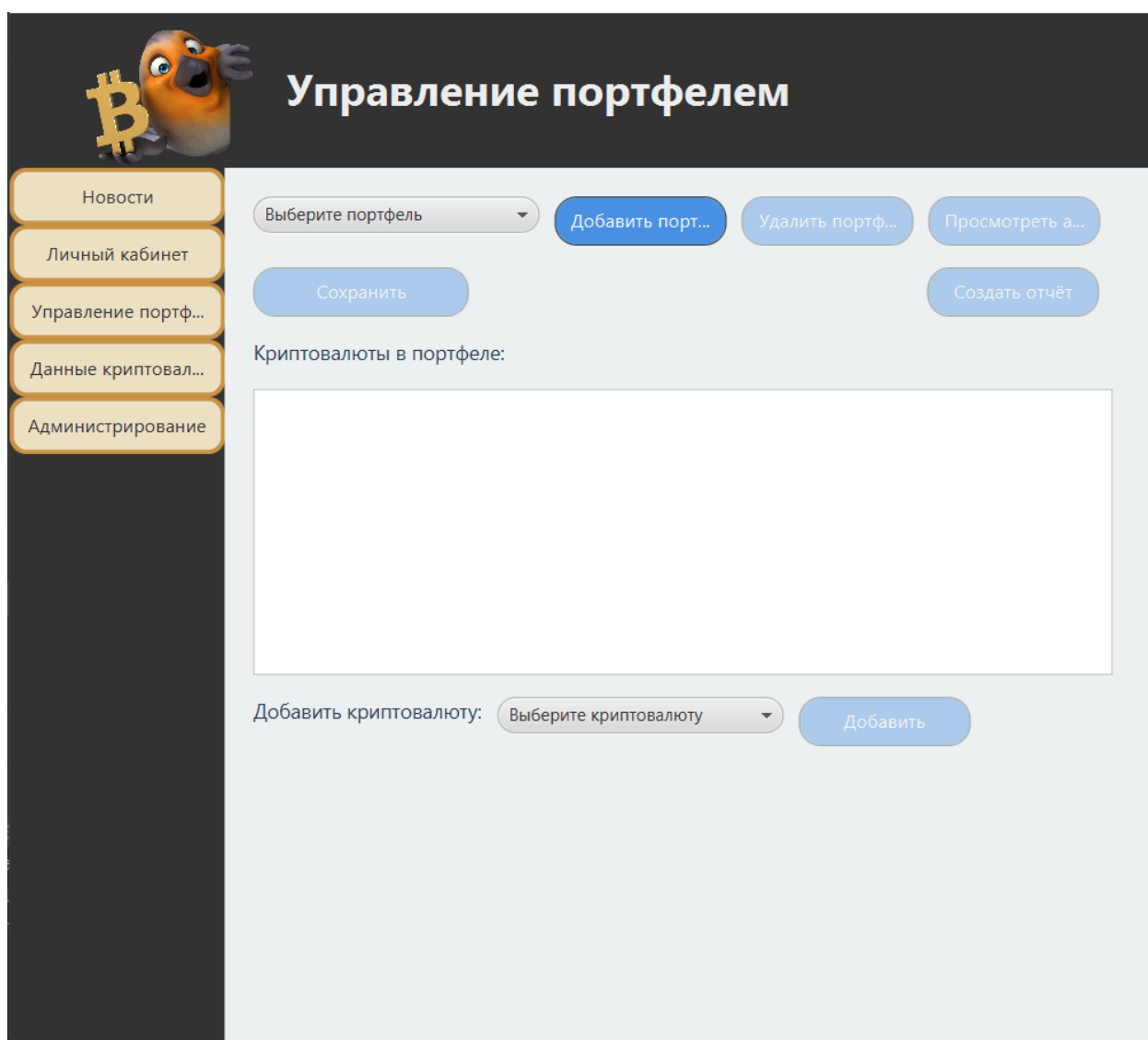


Рисунок 3.4 – Дизайн формы управления портфелями

Разработка пользовательского интерфейса для программного средства станет важным шагом в обеспечении его практической ценности, предоставляя интуитивно понятный и функциональный доступ к данным и аналитике. Использование JavaFX в сочетании с логической структурой вкладок и визуальными элементами обеспечит удобство работы для всех категорий пользователей, включая инвесторов, аналитиков и администраторов. Разработанный интерфейс охватывает все ключевые функции системы: авторизацию, регистрацию, управление портфелями, аналитику, мониторинг цен и просмотр новостей, подготавливая основу для эффективного управления портфелями криптовалют на основе модели Марковица.

3.3 Разработка модели данных

Информационная модель является ключевым элементом проектирования базы данных, обеспечивая структурированное представление данных и их взаимосвязей в рамках системы управления портфелями криптовалют на основе модели Марковица. Она служит инструментом для определения основных компонентов системы, их атрибутов и отношений, что позволяет оптимизировать процессы хранения, обработки и анализа информации. Информационная модель может быть представлена в виде ER-диаграммы (Entity-Relationship Diagram), которая наглядно отображает сущности, их атрибуты и связи между ними, упрощая разработку и поддержку базы данных. Актуальность данной модели обусловлена её применением в инвестиционной сфере, где точность и эффективность управления портфелями криптовалют имеют решающее значение для инвесторов и аналитиков [8].

ER-диаграмма представляет собой графическую модель, которая описывает структуру данных системы и включает в себя 3 основных компонента:

- сущности – объекты реального мира или концепции, которые имеют независимое существование и могут быть однозначно идентифицированы. Каждая сущность представлена в виде прямоугольника на диаграмме;
- атрибуты – характеристики или свойства, описывающие сущности;
- связи – ассоциации между сущностями, показывающие, как они взаимодействуют друг с другом.

Эти компоненты объединены в единую структуру, отражающую логику работы системы и обеспечивающую её целостность. Визуализация информационной модели представлена на рисунке 3.5.

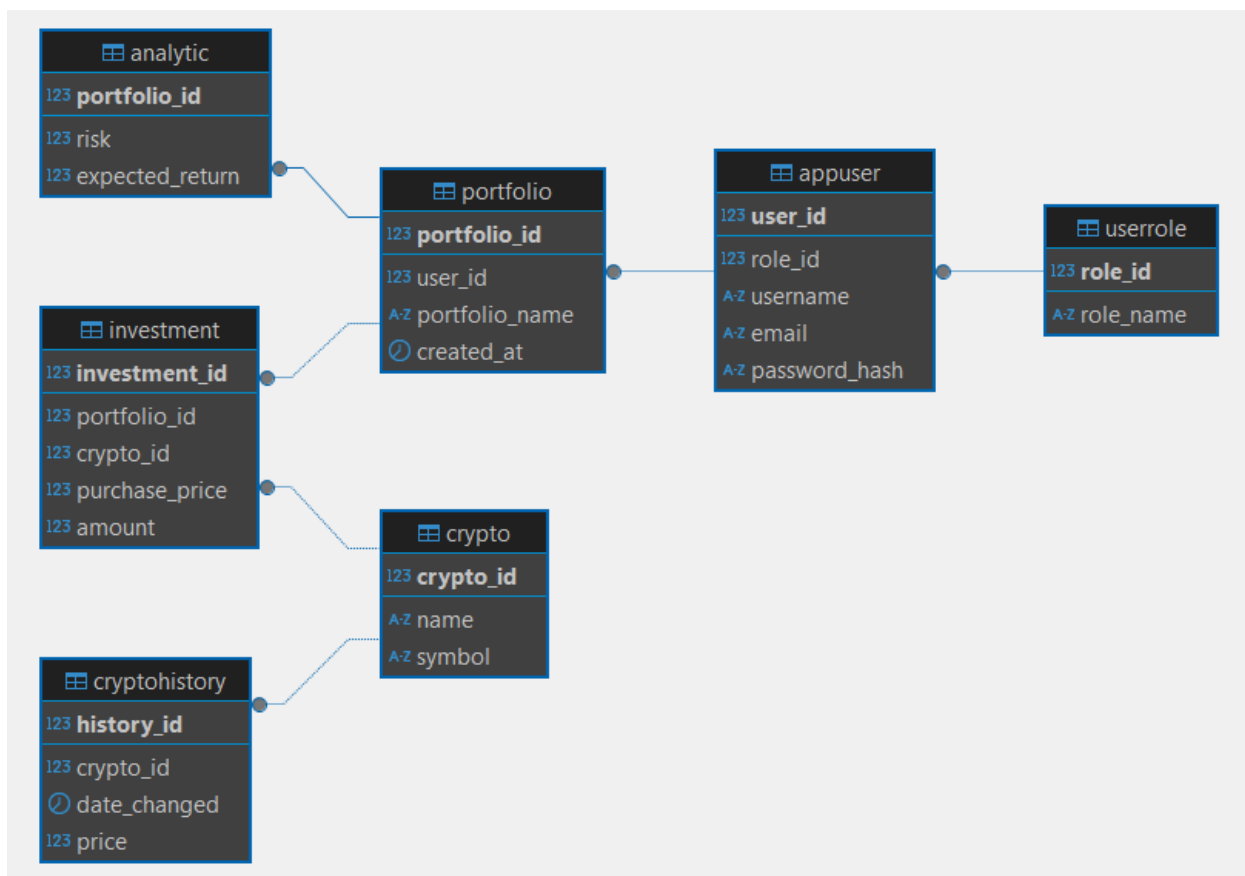


Рисунок 3.5 – Модель базы данных

Представленная ER-диаграмма соответствует третьей нормальной форме (3NF), что обеспечивает минимизацию избыточности данных и упрощение их обновления. Рассмотрим соответствие модели каждой из нормальных форм:

1 Первая нормальная форма (1NF): все атрибуты атомарны, то есть не содержат повторяющихся групп или множественных значений. Например, поле `username` в таблице `appuser` хранит одно значение для каждого пользователя, а инвестиции в портфеле вынесены в отдельную таблицу `investment` с уникальными записями.

2 Вторая нормальная форма (2NF): все неключевые атрибуты полностью зависят от первичного ключа. Например, в таблице `investment` атрибуты `purchase_price` и `amount` зависят только от `investment_id`, а связи реализованы через внешние ключи, исключая частичные зависимости.

3 Третья нормальная форма (3NF): отсутствуют транзитивные зависимости. Например, в таблице `appuser` атрибуты `username` и `email` зависят только от `user_id`, а не от других атрибутов, таких как `role_id`.

Описание каждой сущности, включая их назначение и роль в системе управления портфелями криптовалют, представлено в таблице 1. Эта таблица предоставляет детализированную информацию о каждом компоненте базы данных, подчёркивая их функциональное предназначение и значимость для эффективного управления инвестициями в криптовалюты.

Таблица 1 – Сущности модели проектируемой базы данных

Имя	Определение	Определение записи пользователей или программного средства
Role (Роль)	Сущность, представляющая роли пользователей в системе для разграничения доступа.	Определяет права доступа пользователей
User (Пользователь)	Сущность, содержащая данные о зарегистрированных пользователях системы.	Хранит учетные данные и роли пользователей для управления доступом.
Portfolio (Портфель)	Сущность, представляющая инвестиционный портфель пользователя.	Описывает портфели, созданные пользователями, для управления активами.
Analytic (Аналитика)	Сущность, содержащая аналитические данные портфеля (риск и доходность).	Служит для хранения результатов анализа портфелей по модели Марковица.
Crypto (Криптовалюта)	Сущность, представляющая доступные для инвестиций криптовалюты.	Обеспечивает справочник криптовалют для выбора активов в портфелях.
Investment (Инвестиция)	Сущность, описывающая конкретные инвестиции в криптовалюты в портфеле.	Фиксирует данные о покупках криптовалют пользователями в рамках портфеля.
CryptoHistory(История криптовалют)	Сущность, хранящая историю изменений цен криптовалют.	Используется для анализа динамики цен и оптимизации портфелей.

Таблица 2 содержит описание атрибутов каждой сущности. Для каждого атрибута приведено краткое описание его назначения и роли в рамках соответствующей сущности, что упрощает интерпретацию данных. Эти описания необходимы для точного проектирования базы данных и её дальнейшего использования в системе управления портфелями.

Таблица 2 – Описание сущностей базы данных

Таблица	Назначение атрибута	Наименование поля	Тип данных	Ключ
Role (Роль)	Идентификатор роли	role_id	serial	PK
	Название роли	role_name	nvarchar(50)	
User (Пользователь)	Идентификатор пользователя	user_id	serial	PK
	Идентификатор роли	role_id	int	FK, связывает с таблицей Role, кардинальность связи – N:1
	Имя пользователя	username	nvarchar(50)	
	Электронная почта	email	nvarchar(50)	
	Хешированный пароль	password_hash	nvarchar(60)	
Analytic (Аналитика)	Идентификатор портфеля	portfolio_id	int	PK, FK, связывает с таблицей Portfolio, кардинальность связи – 1:1
	Риск портфеля	risk	numeric (10,6)	
	Ожидаемая доходность	expected_return	numeric (10,6)	
Portfolio (Портфель)	Идентификатор портфеля	portfolio_id	int	PK
	Код пользователя	user_id	int	FK, связь с таблицей User, кардинальность связи – N:1
	Имя портфеля	portfolio_name	nvarchar(50)	
	Дата создания	created_at	timestamp	
Crypto (Криптовалюта)	Идентификатор криптовалюты	crypto_id	int	PK
	Название криптовалюты	crypto_name	nvarchar(50)	
	Символ	symbol	nvarchar(10)	
CryptoHistory ()	Идентификатор записи	crypto_hystory_id	serial	PK
	Идентификатор криптовалюты	crypto_id	int	FK, связь с таблицей Crypto, кардинальность связи – N:1
	Дата изменения	date_changed	date	
	Цена	price	numeric (20,8)	

Продолжение таблицы 2

Таблица	Назначение атрибута	Наименование поля	Тип данных	Ключ
Investment (Инвестиция)	Идентификатор инвестиции	investment_id	int	PK
	Идентификатор портфеля	portfolio_id	int	FK, связь с таблицей Portfolio, кардинальность связи – N:1
	Идентификатор криптовалюты	crypto_id	int	FK, связь с таблицей Crypto, кардинальность связи – N:1
	Цена покупки	purchase_price		
	Количество	amount		

Таким образом разработанная информационная модель базы данных для системы управления портфелями криптовалют, основанной на модели Марковица, представляет собой тщательно продуманную структуру, воплощённую в ER-диаграмме с чётко определёнными компонентами и связями, опирающимися на авторитетное руководство. Анализ продемонстрировал полное соответствие модели третьей нормальной форме (3NF), что гарантирует отсутствие избыточности данных и упрощает их модификацию, создавая прочную основу для дальнейшего развития системы. Описания компонентов и атрибутов, представленные в таблицах 1 и 2, служат не только ориентиром для реализации программного обеспечения, но и мощным инструментом для обеспечения целостности и эффективности работы всей системы, обещающей стать надёжным решением для инвесторов и аналитиков.

3.4 Описание статических и динамических аспектов поведения программных объектов

UML (Unified Modeling Language) – графический язык для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению. UML предоставляет стандартизированные нотации и синтаксис для описания различных аспектов программной системы, таких как ее структура, поведение и взаимодействие между ее компонентами. Использование UML-моделей позволяет разработчикам лучше понимать и описывать архитектуру программного обеспечения, представлять ее в виде графических диаграмм и обеспечивать более наглядное взаимодействие между членами команды.

Для системы управления портфелями криптовалют на основе модели Марковица диаграмма классов информационной модели отображает структуру данных, хранящихся в базе данных. Она иллюстрирует ключевые сущности, такие как инвесторы, портфели, криптоактивы и инвестиции, а также их взаимосвязи, обеспечивая ясное представление о том, как данные организованы и хранятся. Это позволяет разработчикам эффективно проектировать и поддерживать систему, обеспечивая её соответствие требованиям предметной области. На рисунке 3.7 представлена диаграмма классов информационной модели.

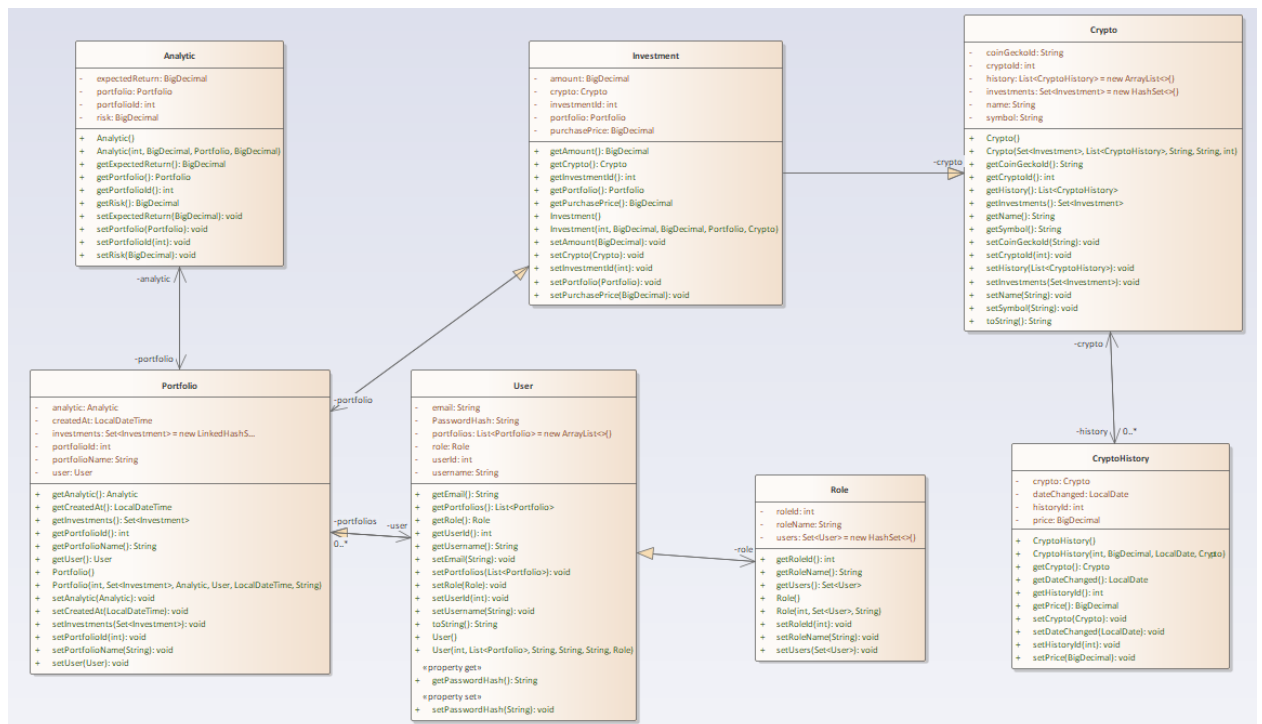


Рисунок 3.6 – Диаграмма классов информационной модели

В рамках клиентской части приложения диаграмма классов контроллеров отражает организацию компонентов, управляющих взаимодействием между пользовательским интерфейсом и серверной логикой. Она включает основные классы, отвечающие за обработку действий пользователей, таких как авторизация, управление портфелями, анализ данных и работа с новостями, а также связи между ними, что упрощает понимание архитектуры системы и её дальнейшее развитие. На рисунке 3.8 изображена диаграмма классов контроллеров.

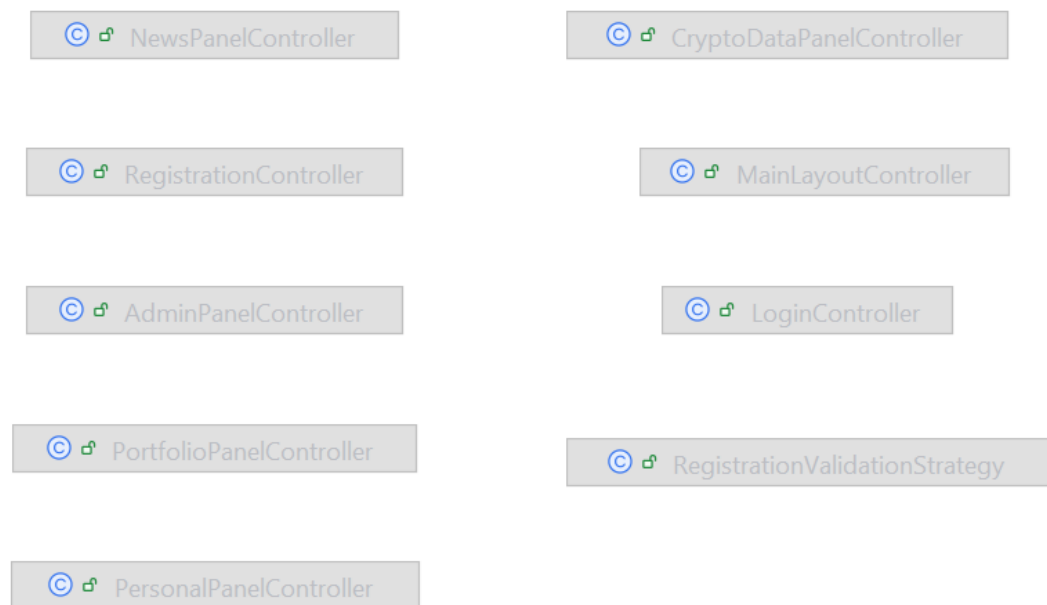


Рисунок 3.7 – Диаграмма классов контроллеров

Моделирование поведения системы осуществляется с помощью диаграммы состояний, которая описывает возможные состояния объектов и переходы между ними в зависимости от событий. В контексте данной системы она помогает понять, как портфель проходит через этапы создания, оптимизации и анализа, а также как пользователь взаимодействует с системой в различных состояниях, таких как авторизованный или неавторизованный доступ. На рисунке 3.8 представлена диаграмма состояний.

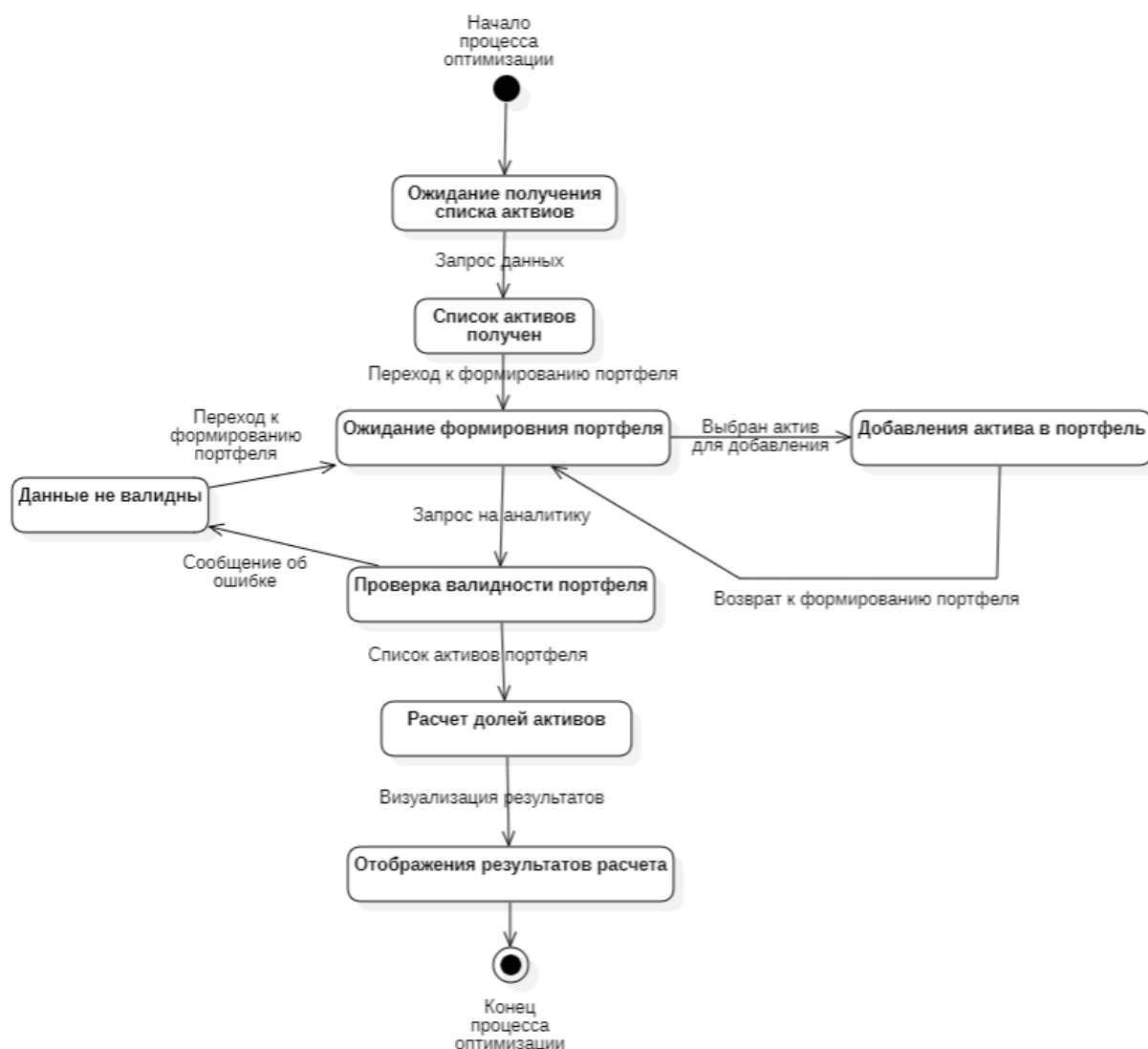


Рисунок 3.8 – Диаграмма состояний формирования оптимального инвестиционного портфеля

Анализ взаимодействия объектов системы проводится с использованием диаграммы последовательности, которая отображает процесс обмена сообщениями между объектами в определённом сценарии, например, при создании портфеля. Она показывает, как пользователь через интерфейс инициирует запрос, который передаётся на сервер для выполнения расчётов по модели Марковица, и как результаты возвращаются обратно для отображения. На рисунке 3.9 изображена диаграмма последовательности.

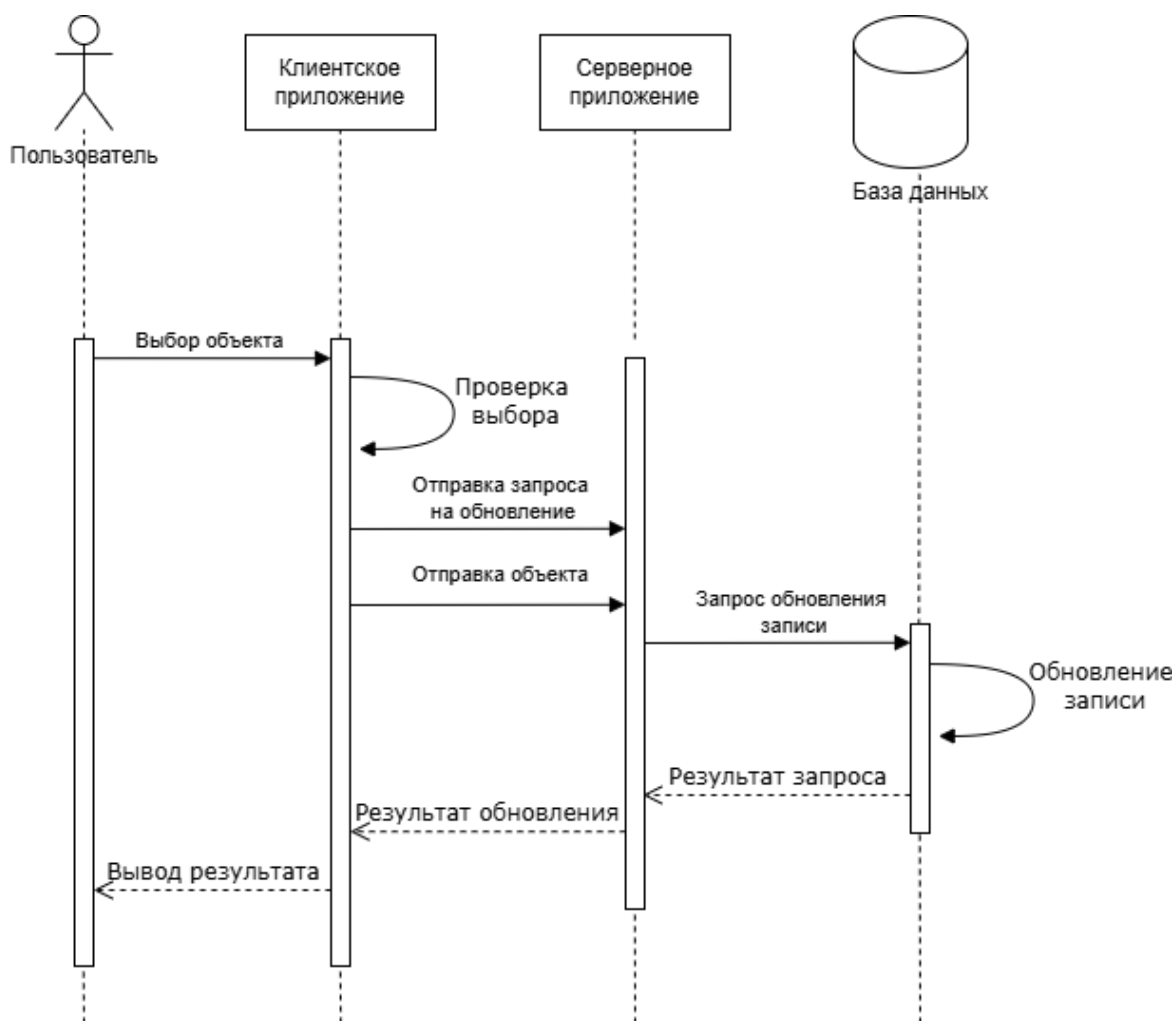


Рисунок 3.9 – UML-диаграмма последовательности

Структура системы на уровне модулей описывается с помощью диаграммы компонентов, которая иллюстрирует организацию программных модулей и зависимости между ними. В данном проекте она включает клиентскую часть с интерфейсом, серверную логику, базу данных и внешние API, показывая, как эти компоненты взаимодействуют для обеспечения функциональности системы. На рисунке 3.10 представлена диаграмма компонентов.

Рисунок 3.10 – UML-диаграмма компонентов

Физическая архитектура системы представлена с помощью диаграммы развёртывания, которая отображает конфигурацию узлов и размещённых на

них компонентов. Она иллюстрирует, как клиентское приложение, сервер и база данных располагаются на различных устройствах, взаимодействуя через сеть, что помогает понять требования к аппаратному обеспечению и сетевой инфраструктуре на этапе проектирования. На рисунке 3.11 изображена диаграмма развёртывания.

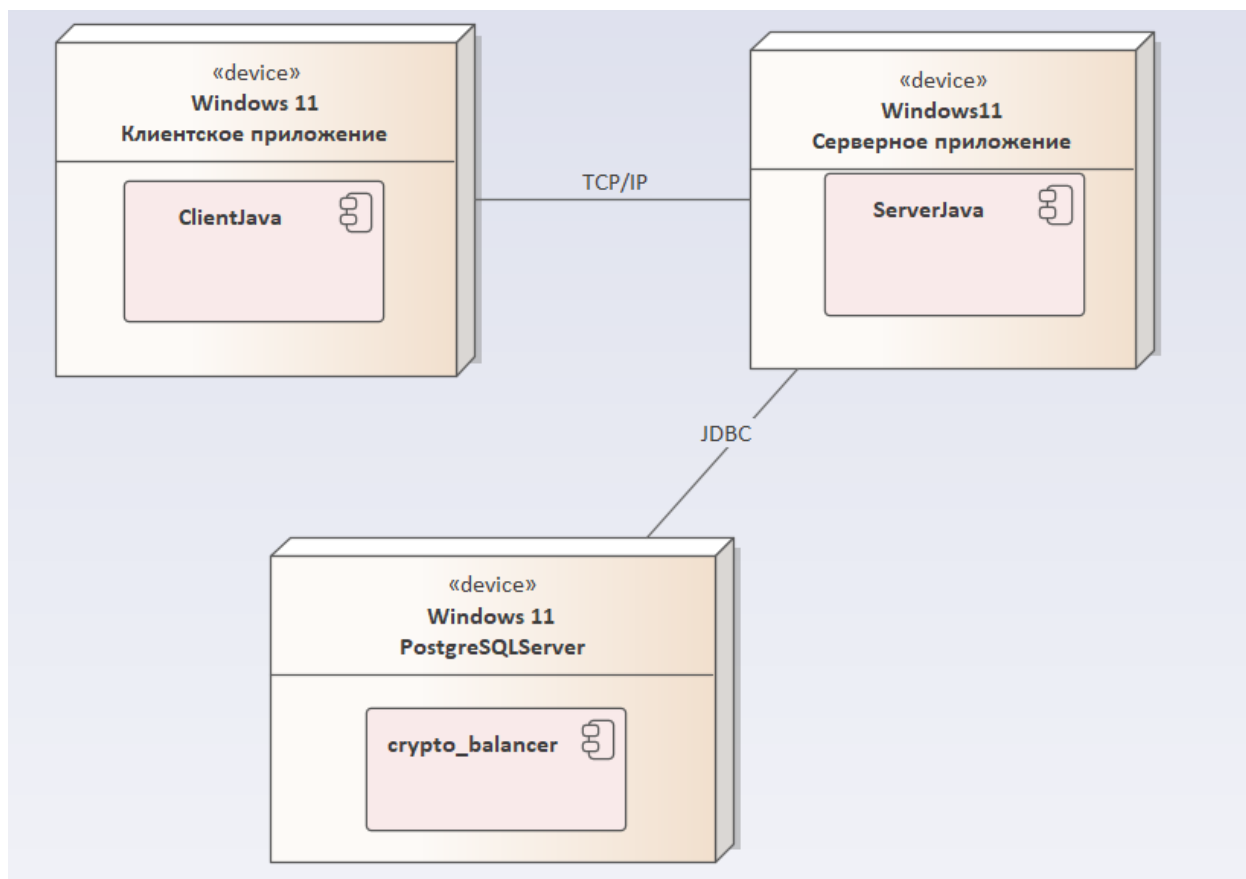


Рисунок 3.11 – UML-диаграмма развёртывания

Разработка UML-моделей для системы управления портфелями криптовалют позволила создать целостное представление её архитектуры и поведения. Эти модели обеспечили чёткое понимание структуры данных, организации клиентской части, динамики взаимодействия компонентов и физического развёртывания, что упростило процесс проектирования и создало прочную основу для дальнейшей реализации и тестирования программного средства.

3.5 Разработка и описание алгоритмов, реализующих бизнес-логику программного средства

Алгоритм – это упорядоченная совокупность точных и понятных исполнителю команд, задающих порядок и содержание действий, которые он должен выполнить для нахождения решения любой задачи из рассматриваемого класса задач. Хорошо разработанные алгоритмы позволяют программистам создавать эффективный и оптимизированный код, обеспечивая быстрое и точное выполнение задач. Они служат основой для проектирования программного обеспечения, включая такие аспекты, как управление данными, обработка запросов и взаимодействие с пользователем.

Схема алгоритма представляет собой графическое изображение алгоритма, использующее стандартные символы и соединительные линии для иллюстрации последовательности шагов и операций, необходимых для решения задачи. Такие схемы облегчают визуальное восприятие структуры алгоритма, способствуя его анализу, модификации и передаче знаний другим разработчикам. Каждый элемент схемы обозначает определённый тип операции: ввод данных (овалы), вычисления или процессы (прямоугольники), условные переходы (ромбы) и вывод результатов (другие фигуры), что делает их универсальным инструментом для проектирования.

На рисунке 3.12 представлена схема алгоритма удаления пользователя. Эта функция доступна исключительно администраторам системы. Процесс начинается с отображения списка пользователей в табличной форме на панели администрирования, откуда администратор выбирает объект для удаления. Далее система проверяет валидность выбранных данных, чтобы убедиться, что пользователь действительно выбран и его возможно удалить. При успешной валидации клиент отправляет запрос на сервер через сетевое соединение, где сервер выполняет удаление записи из базы данных. После обработки запроса сервер возвращает ответ клиенту, который затем отображается на экране в виде уведомления об успехе или ошибке. Такой подход обеспечивает контроль доступа и согласованность данных между клиентом и сервером.

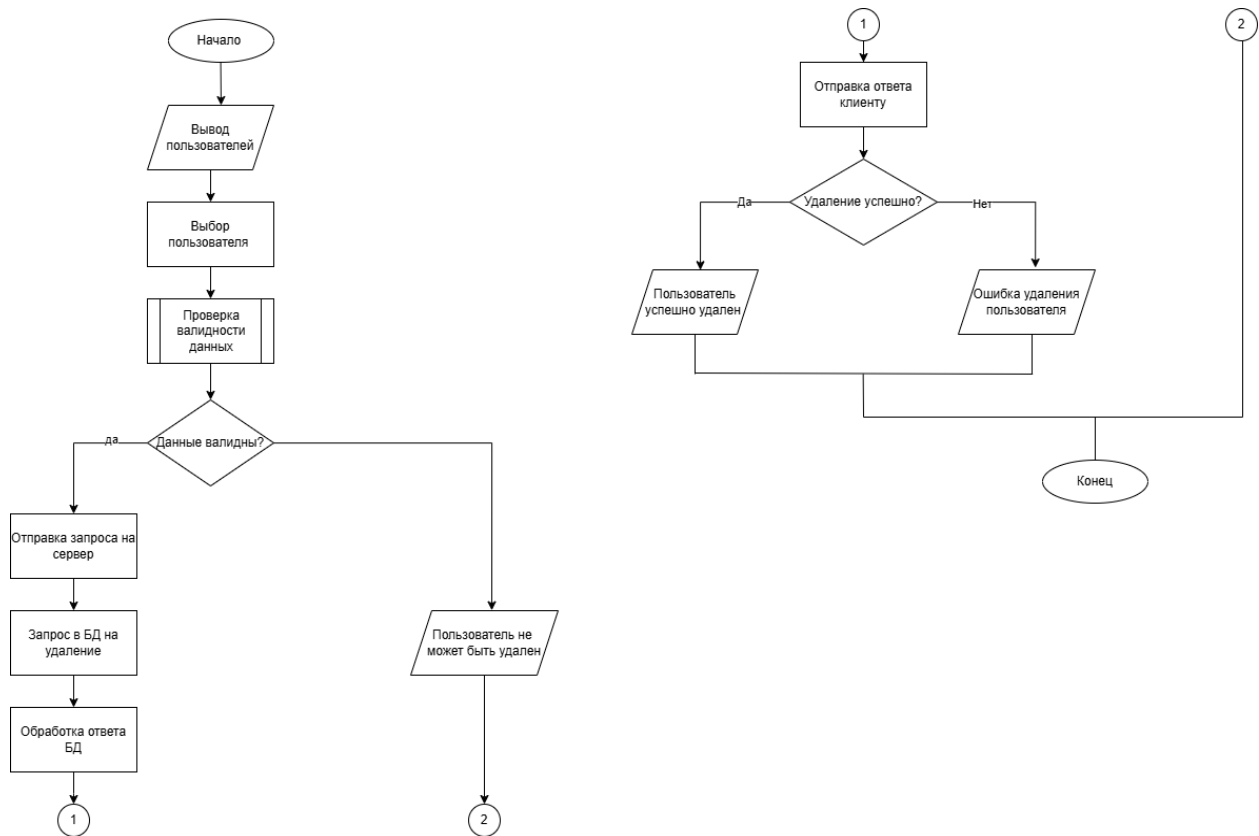


Рисунок 3.12 – Схема алгоритма удаления пользователя

На рисунке 3.13 представлена схема алгоритма оптимизации портфеля, доступная как обычным пользователям, так и администраторам. Процесс начинается с получения данных портфеля и их отображения на экране в панели управления портфелем. Пользователь выбирает необходимый набор данных, который затем проходит валидацию на стороне клиента для проверки корректности выбора. При успешной валидации клиент отправляет запрос на сервер через сетевое. Сервер выполняет функцию оптимизации портфеля, и сохраняет результаты оптимизации в базу данных. После обработки запроса сервер возвращает ответ клиенту, содержащий статус операции и, при необходимости, обновлённые данные портфеля. Клиентская часть обрабатывает ответ, отображая результат операции на экране через уведомление и обновляет интерфейс, если оптимизация прошла успешно. Такой подход обеспечивает гибкость, согласованность данных и информативность для пользователя.

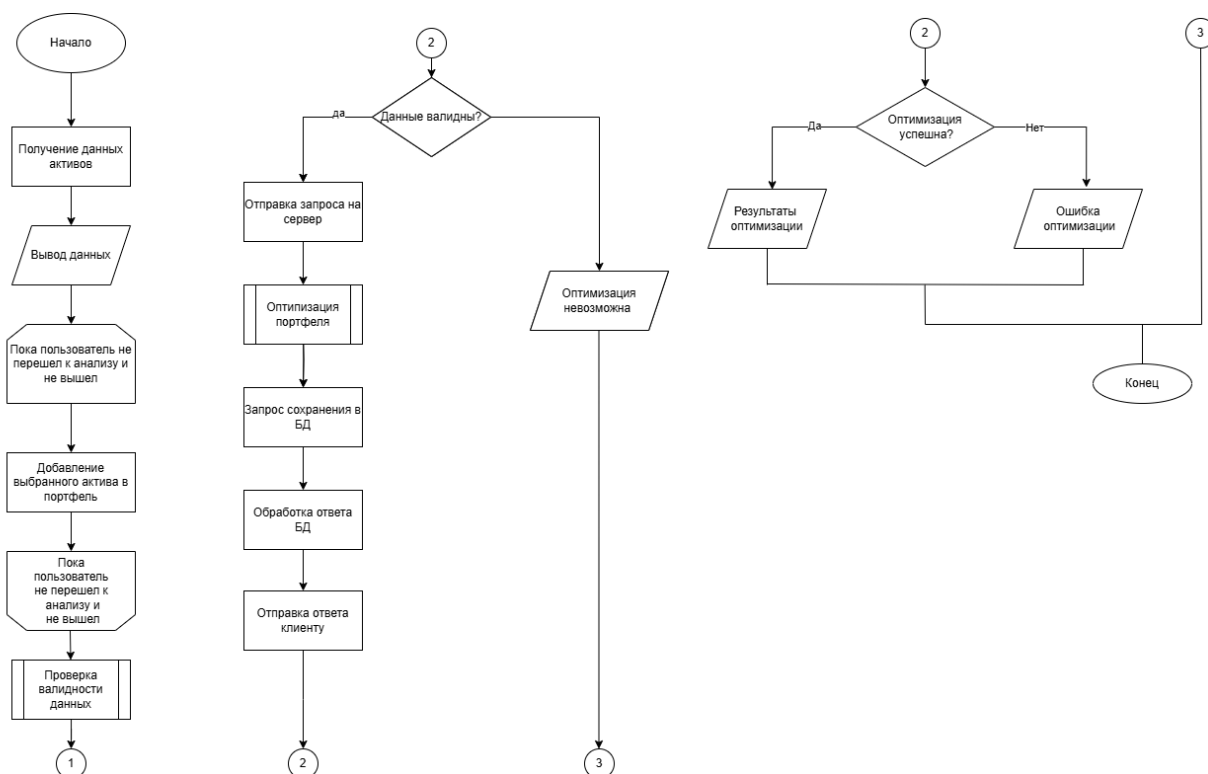


Рисунок 3.13 – Схема алгоритма оптимизации портфеля

Использование схем алгоритмов способствует улучшению документации проекта. Графическое представление процессов, дополненное текстовыми описаниями, делает документацию более структурированной и доступной для понимания. Это особенно важно для долгосрочных проектов, где поддержка и модернизация программы могут осуществляться спустя значительное время после её первоначальной разработки. Схемы становятся своего рода "дорожной картой", которая помогает ориентироваться в логике приложения и вносить изменения с минимальными рисками.

Таким образом, разработка схем алгоритмов является неотъемлемой частью процесса создания программного обеспечения, обеспечивая его высокое качество, надёжность и удобство сопровождения. В контексте курсового проекта схемы алгоритмов не только упростили проектирование ключевых функций, таких как управление пользователями и портфелями, но и заложили основу для дальнейшего масштабирования приложения. Их применение подтверждает, что хорошо продуманные алгоритмы и их визуализация являются залогом успешной реализации сложных систем, способных эффективно решать поставленные задачи и удовлетворять потребности пользователей.

3.6 Механизмы обеспечения информационной безопасности

Информационная безопасность является неотъемлемой частью функционирования системы управления портфелями криптовалют. Для защиты данных пользователей и обеспечения устойчивости к внешним и внутренним угрозам в системе реализован ряд базовых, но эффективных мер, направленных на предотвращение несанкционированного доступа, утечек информации и нарушения целостности данных:

1 Пользователи системы имеют разграниченные полномочия в зависимости от своей роли. Такой подход позволяет ограничить круг доступных функций и данных для каждого типа пользователей, что минимизирует риски злоупотребления полномочиями и доступа к конфиденциальной информации;

2 При регистрации и работе с системой обеспечивается уникальность пользовательских идентификаторов и корректность вводимых данных. Это способствует предотвращению дублирования аккаунтов и повышает надёжность системы аутентификации;

3 Для повышения стойкости учетных записей к взлому применяются базовые требования к сложности паролей. Это снижает вероятность успешного подбора пароля и защищает от несанкционированного доступа;

4 Доступ к внутренним данным системы осуществляется только через серверную сторону программы, что исключает возможность прямого взаимодействия с критически важными компонентами, такими как база данных. Это помогает предотвратить возможные атаки и вмешательства со стороны пользователей;

5 Пароли пользователей обрабатываются с помощью алгоритмов хэширования, что позволяет сохранить их в закрытом виде, даже при несанкционированном доступе к базе данных.

В совокупности эти меры обеспечивают надёжную защиту информации в системе, соответствуя современным стандартам безопасности и требованиям к разработке программных продуктов, обрабатывающих конфиденциальные данные.

4 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование является важным этапом процесса разработки программного обеспечения. Оно необходимо для проверки соответствия реализованной функциональности требованиям, обнаружения возможных ошибок и обеспечения надежности системы. При написании кода программы необходимо предусмотреть обработку исключительных ситуаций, таких как:

- введенные пользователем данные не соответствуют формату поля (например, неправильный формат электронной почты);
- неверно введен логин или пароль при входе в личный кабинет;
- логин уже существует при регистрации нового пользователя;
- введенные пользователем данные нелогичны (например, отрицательное значение доходности портфеля);
- подтверждение удаления портфеля или пользователя;
- ничего не найдено по результатам поиска криптовалюты или пользователя;
- запрет удаления администратором собственного аккаунта;
- подтверждение пароля при регистрации и смене пароля.

В ходе тестирования была проверена обработка ввода данных в поле электронной почты. Пользователь ввёл некорректный формат, и программа обнаружила ошибку, отобразив сообщение об ошибке (рисунок 4.1).

Рисунок 4.1 – Ввод неправильного формата электронной почты

Также тестировался вход в личный кабинет с неверным логином или паролем. Программа выявила несоответствие и уведомила пользователя сообщением (рисунок 4.2).

Рисунок 4.2 – Ввод неверного логина или пароля

Проверка регистрации нового пользователя показала, что при попытке создать аккаунт с существующим логином программа выдаёт уведомление, изображенное на рисунке 4.3.

Рисунок 4.3 – Попытка создать аккаунт с существующим логином

Для предотвращения ввода нелогичных данных было протестировано добавление портфеля с отрицательной доходностью. Программа обнаружила ошибку и отобразила сообщение (рисунок 4.4).

Рисунок 4.4 – Попытка добавить портфель с отрицательной доходностью

Чтобы исключить случайное выполнение необратимых действий, в программе реализовано подтверждение удаления. Например, при удалении портфеля или пользователя появляется диалоговое окно с запросом (рисунок 4.5).

Рисунок 4.5 – Попытка выполнить удаление

Поиск криптовалюты или пользователя также был протестирован. При отсутствии совпадений программа выводит соответствующее сообщение (рисунок 4.6).

Рисунок 4.6 – Результат поиска криптовалюты и пользователя

Для защиты от ошибочных действий администратора реализован запрет на удаление собственного аккаунта. При попытке выполнить такое действие программа выдаёт уведомление (рисунок 4.7).

Рисунок 4.7 – Попытка администратора удалить собственную учетную запись

Наконец, при регистрации нового пользователя или смене пароля добавлена проверка подтверждения пароля. Если пароли не совпадают, программа уведомляет пользователя (рисунок 4.8).

Рисунок 4.8 – Проверка совпадения паролей при регистрации или смене пароля

Для повышения безопасности и предотвращения некорректных действий в программе реализована динамическая доступность кнопок. Кнопки, связанные с действиями, которые недоступны при текущем наборе данных, теряют способность быть нажатыми (рисунок 4.9).

Рисунок 4.9 – Динамическая доступность кнопок в зависимости от текущего набора данных

Таким образом, клиентская часть программы CryptoBalancer предусматривает комплексную обработку исключительных ситуаций, что существенно снижает риск ошибок и обеспечивает стабильную работу приложения. Это достигается благодаря встроенным проверкам корректности данных, которые позволяют оперативно выявлять некорректные значения, такие как неверный формат электронной почты или отрицательная доходность портфеля, предотвращая сбои и сохраняя целостность системы.

При выполнении важных операций, таких как удаление портфеля, пользователя или внесение изменений в базу данных, предусмотрены механизмы подтверждения действий. Эти меры, реализованные через диалоговые окна с запросом на согласие, минимизируют вероятность случайных ошибок со стороны пользователей, обеспечивая дополнительный уровень защиты данных. Кроме того, для случаев критических ошибок или некорректного взаимодействия с сервером и базой данных используются блоки обработки исключений. Такой подход предотвращает аварийное завершение программы и позволяет продолжить работу даже при возникновении сбоев.

Пользователям предоставляются информативные и понятные сообщения об ошибках, сопровождаемые рекомендациями по их устранению, что делает интерфейс приложения более интуитивным и удобным. Например, уведомления о дублировании логина или отсутствии результатов поиска помогают пользователю быстро понять причину проблемы и принять корректное решение.

Динамическая доступность кнопок, зависящая от текущего состояния данных, добавляет ещё один слой безопасности, делая недоступными действия, которые нельзя выполнить (например, сохранение портфеля без выбора криптовалют).

В итоге, благодаря тщательной обработке исключительных ситуаций, курсовой проект не только удовлетворяет строгим требованиям надёжности и стабильности, но и обеспечивает комфортную и безопасную работу для пользователей, минимизируя вероятность ошибок и некорректного поведения системы.

5 РУКОВОДСТВО ПО УСТАНОВКЕ (РАЗВЕРТЫВАНИЮ) И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

5.1 Руководство по установке (развертыванию) программного средства

5.2 Руководство пользователя.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были изучены принципы формирования инвестиционного портфеля криптовалют и разработано клиент-серверное приложение на основе модели Марковица, а также рассмотрены различные варианты реализации данной задачи. Было установлено, что автоматизация расчётов позволяет уменьшить трудоёмкость управления портфелями, минимизировать ошибки и повысить эффективность принятия инвестиционных решений. Пользовательские функции приложения обеспечивают удобную работу с данными, включая создание портфелей, анализ рисков и доходности, а также визуализацию результатов. Также были предусмотрены исключительные ситуации и их обработка с использованием механизмов безопасности.

Созданные пользовательские функции позволяют эффективно управлять портфелями, проводить анализ и получать рекомендации на основе актуальных данных. Функциональная схема работы программы представлена в виде алгоритмов отдельных функций. Приложение обладает понятным графическим интерфейсом, что позволяет пользователям с минимальными навыками легко использовать систему для управления активами и просмотра аналитики.

Для понимания логики работы программы были разработаны модели бизнес-процессов предметной области в нотациях IDEF0 и UML, а также проектные решения на основе архитектурных подходов. Программное средство экономно использует ресурсы системы, не перегружает процессор и обеспечивает ввод корректных данных благодаря интеграции с внешними API и безопасным механизмам авторизации.

Цель, заявленная во введении этой работы (разработка клиент-серверного приложения, автоматизирующего формирование оптимального инвестиционного портфеля криптовалют с использованием модели Марковица), была успешно достигнута.

Как итог данной работы можно назвать функциональное приложение для управления криптовалютными портфелями, включающее элементы многопоточной архитектуры, аналитический модуль и интеграцию с API.

Несомненно, в дальнейшем система может быть доработана оптимизацией и добавлением новых функций, таких как прогнозирование рыночных трендов и расширение аналитических инструментов. Такие улучшения позволят системе не только соответствовать текущим требованиям, но и адаптироваться к будущим изменениям на криптовалютном рынке.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Модель Марковица и финансовый портфель – Empirix [Электронный ресурс]. – Режим доступа: <https://empirix.ru/optimizacziya-portfelya-po-markoviczu/>.

[2] Portfolio Performance. (2023). Open Source Portfolio Management Tool [Электронный ресурс]. – Режим доступа: <https://www.portfolio-performance.info>.

[3] Обзор CoinStats: Сервис Агрегации Портфелей [Электронный ресурс]. – Режим доступа: <https://ru.bitdegree.org/crypto/obzor-coinstats>.

4 Обзор Cryptocompare — инструмент для контроля (учёта) своего инвестиционного портфеля [Электронный ресурс]. – Режим доступа: <https://happycoin.club/41066-2/>.

[5] Автоматизация процессов — средства и этапы автоматизации [Электронный ресурс]. – Режим доступа: <https://proresult.by/sredstva-i-etapy-avtomatizatsii/>.

[6] Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Addison-Wesley, 2003.

[7] Galitz, W. O. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques. 3rd ed. Wiley, 2007.

[8] Coronel, C., Morris, S. Database Systems: Design, Implementation, and Management. 13th ed. Cengage Learning, 2019.

ПРИЛОЖЕНИЕ А
(обязательное)

Отчет о проверке на заимствование в системе «Антиплагиат»

Рисунок А.1 – Результат проверки на заимствования

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг кода алгоритмов, реализующих основную бизнес-логику

```
private void loadCryptoData(String cryptoName) {
    try {
        ClientSocket.getInstance().getOut().println(gson.toJson(new
Request(RequestType.GET_CRYPTO_DATA, cryptoName)));
        ClientSocket.getInstance().getOut().flush();
        String responseJson =
ClientSocket.getInstance().getInStream().readLine();
        Response response = gson.fromJson(responseJson, Response.class);
        if (response.getResponseStatus() == ResponseStatus.OK) {
            Type historyType = new
TypeToken<List<CryptoHistory>>().getType();
            List<CryptoHistory> historyList =
gson.fromJson(response.getResponseData(), historyType);
            if (historyList != null && !historyList.isEmpty()) {
                updateChartAndInfoWithHistory(cryptoName, historyList);
            } else {
                AlertUtil.showError("Ошибка", "История для криптовалюты не
найдена", "");
            }
        } else {
            AlertUtil.showError("Ошибка", "Не удалось загрузить данные
криптовалюты", response.getResponseMessage());
        }
    } catch (IOException e) {
        e.printStackTrace();
        AlertUtil.showError("Ошибка", "Не удалось загрузить данные
криптовалюты", e.getMessage());
    }
}

private void loadPortfolioData(String portfolioName){
    try {
        ClientSocket.getInstance().getOut().println(gson.toJson(new
Request(RequestType.GET_PORTFOLIO_DATA, gson.toJson(portfolioName))));
        ClientSocket.getInstance().getOut().flush();
        String responseMessage =
ClientSocket.getInstance().getInStream().readLine();
        Response response = gson.fromJson(responseMessage, Response.class);
        if (response.getResponseStatus() == ResponseStatus.OK) {
            currentPortfolio = gson.fromJson(response.getResponseData(),
Portfolio.class);
            if (currentPortfolio != null) {
                updatePortfolioUI(currentPortfolio);
            } else {
                AlertUtil.showError("Ошибка", "Данные портфеля пусты", "");
                cryptoPortfolioComboBox.setItems(cryptoNames);
            }
        } else {
            AlertUtil.showError("Ошибка", "Не удалось загрузить данные
портфеля", response.getResponseMessage());
            cryptoPortfolioComboBox.setItems(cryptoNames);
        }
    } catch (IOException e) {
        e.printStackTrace();
        AlertUtil.showError("Ошибка", "Не удалось загрузить данные портфеля",
```

Продолжение приложения Б

```
e.getMessage());
    cryptoPortfolioComboBox.setItems(cryptoNames);
}
}

public class ClientSocket {
    private static final ClientSocket SINGLE_INSTANCE = new ClientSocket();
    private ClientSocket() {
        try {
            socket = new Socket("localhost", 5555 );
            in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            out = new PrintWriter(socket.getOutputStream());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static ClientSocket getInstance() {
        return SINGLE_INSTANCE;
    }
    private User user;
    private static Socket socket;
    private BufferedReader in;
    private PrintWriter out;
    public Socket getSocket() {
        return socket;
    }
    public BufferedReader getInStream() {
        return in;
    }
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }
    public PrintWriter getOut() {
        return out;
    }
    public void setOut(PrintWriter out) {
        this.out = out;
    }
}

public class Request {
    private RequestType requestType;
    private String requestMessage;
    public Request() {
    }
    public Request(RequestType requestType){
        this.requestType = requestType;
    }
    public Request(RequestType requestType, String requestMessage) {
        this.requestType = requestType;
        this.requestMessage = requestMessage;
    }
}
```

Продолжение приложения Б

```
public String getRequestMessage() {
    return requestMessage;
}
public RequestType getRequestType() {
    return requestType;
}
public void setRequestMessage(String requestMessage) {
    this.requestMessage = requestMessage;
}
public void setRequestType(RequestType requestType) {
    this.requestType = requestType;
}
}

public class Response {
    private ResponseStatus responseStatus;
    private String responseMessage;
    private String responseData;
    public Response(ResponseStatus responseStatus, String
responseMessage,String responseData) {
        this.responseStatus = responseStatus;
        this.responseMessage = responseMessage;
        this.responseData = responseData;
    }
    public Response(){
    }
    public ResponseStatus getResponseStatus() {
        return responseStatus;
    }
    public String getResponseData() {
        return responseData;
    }
    public String getResponseMessage() {
        return responseMessage;
    }
    public void setResponseStatus(ResponseStatus responseStatus) {
        this.responseStatus = responseStatus;
    }
    public void setResponseData(String responseData) {
        this.responseData = responseData;
    }
    public void setResponseMessage(String responseMessage) {
        this.responseMessage = responseMessage;
    }
}
```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг SQL-скрипта

```
-- Таблица userrole (Role)
CREATE TABLE crypto_balancer.userrole (
    role_id SERIAL PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL
);

-- Таблица appuser (User)
CREATE TABLE crypto_balancer.appuser (
    user_id SERIAL PRIMARY KEY,
    role_id INT NOT NULL,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(60) NOT NULL, -- Увеличено до 60 для поддержки BCrypt
    FOREIGN KEY (role_id) REFERENCES crypto_balancer.userrole(role_id) ON
DELETE RESTRICT ON UPDATE CASCADE
);

-- Таблица portfolio (Portfolio)
CREATE TABLE crypto_balancer.portfolio (
    portfolio_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    portfolio_name VARCHAR(50) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES crypto_balancer.appuser(user_id) ON DELETE
CASCADE ON UPDATE CASCADE
);

-- Таблица analytic (Analytic)
CREATE TABLE crypto_balancer.analytic (
    portfolio_id INT PRIMARY KEY,
    risk NUMERIC(10, 6) NOT NULL,
    expected_return NUMERIC(10, 6) NOT NULL,
    FOREIGN KEY (portfolio_id) REFERENCES
crypto_balancer.portfolio(portfolio_id) ON DELETE CASCADE ON UPDATE CASCADE
);

-- Таблица crypto (Crypto)
CREATE TABLE crypto_balancer.crypto (
    crypto_id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    symbol VARCHAR(10) NOT NULL UNIQUE
);

-- Таблица cryptohistory (CryptoHistory)
CREATE TABLE crypto_balancer.cryptohistory (
    history_id SERIAL PRIMARY KEY,
    crypto_id INT NOT NULL,
    date_changed DATE NOT NULL,
    price NUMERIC(20, 8) NOT NULL,
    FOREIGN KEY (crypto_id) REFERENCES crypto_balancer.crypto(crypto_id) ON
DELETE CASCADE ON UPDATE CASCADE
);

-- Таблица investment (Investment)
CREATE TABLE crypto_balancer.investment (
```

Продолжение приложения В

```
investment_id SERIAL PRIMARY KEY,
portfolio_id INT NOT NULL,
crypto_id INT NOT NULL,
purchase_price NUMERIC(20, 8) NOT NULL,
amount NUMERIC(20, 8) NOT NULL,
FOREIGN KEY (portfolio_id) REFERENCES
crypto_balancer.portfolio(portfolio_id) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (crypto_id) REFERENCES crypto_balancer.crypto(crypto_id) ON
DELETE RESTRICT ON UPDATE CASCADE
);

-- Индексы для оптимизации запросов
CREATE INDEX idx_appuser_role_id ON crypto_balancer.appuser(role_id);
CREATE INDEX idx_portfolio_user_id ON crypto_balancer.portfolio(user_id);
CREATE INDEX idx_cryptohistory_crypto_id ON
crypto_balancer.cryptohistory(crypto_id);
CREATE INDEX idx_investment_portfolio_id ON
crypto_balancer.investment(portfolio_id);
CREATE INDEX idx_investment_crypto_id ON
crypto_balancer.investment(crypto_id);
```

ПРИЛОЖЕНИЕ Г (обязательное)

Схемы алгоритмов

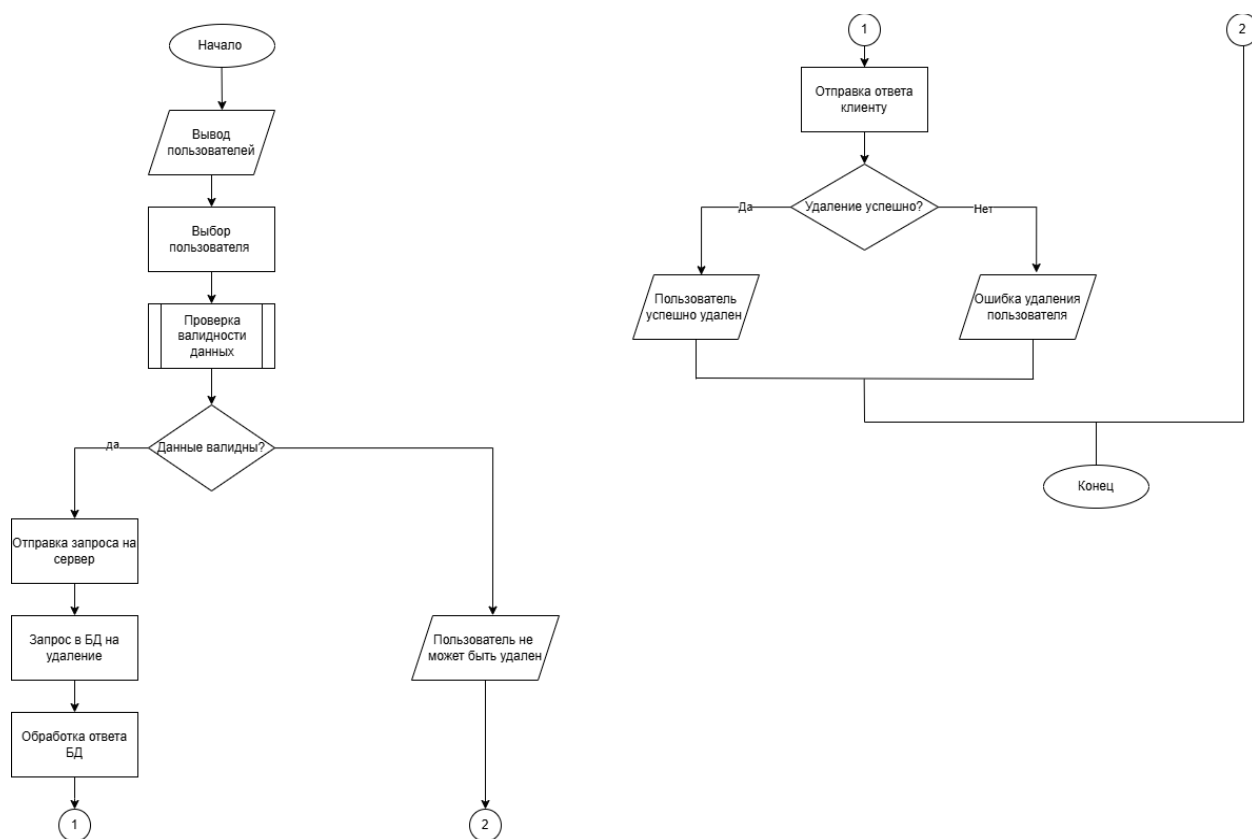


Рисунок Г.1 – Схема алгоритма удаления пользователя

Продолжение приложения Г

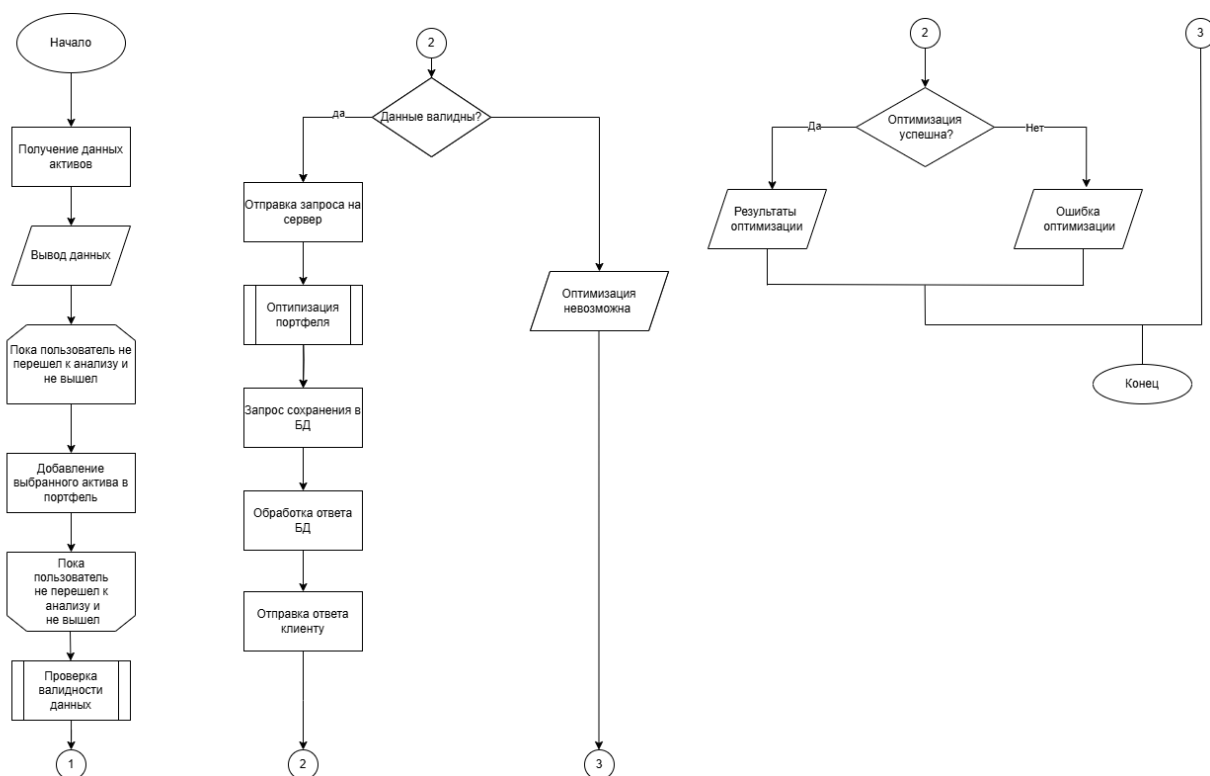


Рисунок Г.2 – Схема алгоритма оптимизации портфеля