МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет вычислительной техники

Кафедра МО и ПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Приложение по работе с криптовалютой на демо-счете

Студент _____ Гумберидзе Артур Русланович
(подпись, дата) (ФИО полностью)

Руководитель _____ Балашова И.Ю.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ Такташкин Д.В
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Макарычев П.П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2022

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____П.П. Макарычев
«__» _____2022 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Гумберидзе Артур Русланович гр. 18ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Приложение по работе с криптовалютой на демо-счете

Тема утверждена приказом ПГУ № 327/О от "18" апреля 2022 г.

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Балашова И.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: приложение по работе с криптовалютой с возможностью торговли на демо-счете и проведения анализа ордера на покупку и статистики поведения криптовалюты.

2) Среда выполнения: операционная система Windows 10, Web-браузер

Языковые средства разработки: Java, JavaScript, UML.

Инструментальные средства: Intellij IDEA, MySQL Workbench.

Технологии разработки ООП, сервисно-ориентированный подход.

3) Основные функции приложения:

а) регистрация в приложении;

б) авторизация в приложении;

в) просмотр личных данных;

г) просмотр статистики изменения криптовалюты;

д) анализ открытия ордера на покупку;

е) покупка и продажа криптовалюты;

ж) просмотр истории торговли;

з) обнуление попытки;

и) изменение личных данных;

з) удаление, создание и редактирование данных пользователя администратором

5. Перечень подлежащих разработке вопросов:

1) *Анализ предметной области и постановка задачи;*
2) *Проектирование;*
3) *Реализация;*
4) *Контроль качества программного обеспечения;*
5) *План разработки бюджета;*
6) *Оформление пояснительной записки.*

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%		
2	Проектирование	25%		
3	Реализация	35%		
4	Контроль качества программного обеспечения	10%		
5	План разработки бюджета	5%		
6	Оформление пояснительной записки	15%		

Дата выдачи задания « » _____2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 88 листов, 34 рисунка, 16 таблиц, 11 использованных источников и 4 приложения.

КРИПТОВАЛЮТА, ДЕМО-СЧЕТ, ТРЕЙДИНГ, КЛИЕНТ, СЕРВЕР ПРИЛОЖЕНИЙ, СЕРВЕР БАЗЫ ДАННЫХ, UML, REST, MVC.

Объект исследования – приложения по работе с криптовалютой.

Цель выпускной квалификационной работы – разработать приложение по работе с криптовалютой с возможность торговать на демо-счете.

Технологии разработки – разработка сервера приложений проводилась на языке программирования Java с использованием фреймворка Spring и ORM Hibernate, клиентская часть разрабатывалась на языке Java с использованием инструмента JavaFX, для создания сервера БД использовалась MySQL. В качестве среды разработки использовался редактор программного кода Intellij IDEA и менеджер баз данных MySQL Workbench.

Среда исполнения – Windows 10, браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть приложения по работе с криптовалютой на демо-счете. Все требования к разработке приложения соблюдены.

Содержание

1 Анализ предметной области, требований и бизнес процессов приложения по работе с криптовалютой на демо-счете 8
1.1 Анализ предметной области 8
1.2 Сравнительный анализ продуктов для торговли на криптобирже 8
1.2.1 Binance App 9
1.2.2 Currency.com 10
1.2.3 Сайт bybit.com 11
1.2.4 Сравнительный анализ программных продуктов для торговли на криптобирже 13
1.3 Постановка задачи 18
1.4 Анализ требований 19
1.4.1 Функциональные требования 19
1.4.2 Нефункциональные требования 24
2 Проектирование приложения по работе с криптовалютой на демо-счете 25
2.1 Архитектура приложения 25
2.2 Проектирование структур данных 25
2.2.1 Логическая модель базы данных приложения 25
2.2.2 Физическая модель базы данных 27
2.3 Проектирование программных средств 28
2.3.1 Проектирование классов приложения торговли криптовалютой на демо-счете 28
2.3.2 Проектирование алгоритма торговли на демо-счете 31
2.3.3 Проектирование сценария торговли на демо-счете 33
2.3.4 Диаграмма кооперации приложения 34
2.4 Проектирование пользовательского интерфейса web-приложения 35
2.5 Проектирование клиентского приложения 39
3 Реализация программного обеспечения сервиса 40
4 Контроль качества программного обеспечения 43
4.1 Функциональное тестирование 43

4.2 Модульное тестирование 46
4.3 Метрики кода 48
4.4 Планирование разработки и оценка бюджета 50
4.4.1 Структурное планирование 50
4.4.2 Календарное планирование 53
Заключение 56
Список использованных источников 57
Приложение А. Глоссарий 59
Приложение Б. Листинг приложения 61
Приложение В. Результаты тестирования 79
Приложение Г. Диаграммы программных классов 83

Введение

На протяжении последних лет в мире наблюдается стремительное распространение криптовалют как средства осуществления электронных платежей. Цель создания криптовалют – обеспечение удобного платежного средства, потенциально привлекательного с инвестиционной точки зрения. По мнению финансовых экспертов «виртуальные валюты могут иметь долгосрочный потенциал, особенно в том случае, если инновации способствуют развитию более быстрой, надежной и эффективной платежной системы» [1].

Торговля криптовалютой чаще всего ведется на специализированных площадках, так называемых биржах. Некоторые криптовалютные биржи позволяют пользователям открывать демо-счета для сделок в тестовом режиме. Эта возможность позволяет начинающим трейдерам изучить работу торгового терминала без риска ошибиться и потерять свои средства. Заключая сделки при помощи виртуальных токенов, пользователь быстро ознакомится с финансовыми инструментами, функционалом площадки. Часто демо-счет используют опытные трейдеры для тестирования новых стратегий торговли или алгоритмов для торговых ботов. Таким образом, разработка приложения, позволяющего трейдерам приобрети навыки инвестирования в криптовалюту и обучиться трейдингу, является актуальной задачей.

Для осуществления поставленной цели были выделены следующие задачи, которые необходимо решить:

– проведение анализа предметной области и существующих приложений аналогичного назначения;

– постановка задачи на разработку приложения для торговли криптовалютой на демо-счете;

– определение требований к разрабатываемому приложению;

– выполнение функционального моделирования системы;

– проектирование приложения торговли криптовалютой на демо-счете;

– реализация приложения;

– тестирование приложения.

Целью выпускной квалификационной работы является разработка приложения по работе с криптовалютой на демо-счете.

1 Анализ предметной области, требований и бизнес процессов приложения по работе с криптовалютой на демо-счете

1.1 Анализ предметной области

Сервисы по работе с криптовалютой является востребованными уже на протяжении многих лет. Криптовалюта как один из видов валюты предоставляет инвестору больший выбор для работы с финансами, а соответственно открывается больше возможностей повысить свою прибыль. В отличие от электронного кошелька, в основе криптовалюты лежит сложный криптографический алгоритм. Принцип работы цифровых монет построен на открытом протоколе передачи данных. Все действия, включая выпуск новых денег и обработку транзакций, выполняются коллективно участниками сети.

Необходимо уметь правильно вести торговлю для того, чтобы уменьшить риски потерять свой капитал. С этой целью некоторые сервисы предлагают трейдеру пройти обучение на демо-счете.

Демонстрационный счет – это виртуальный депозит, предоставляемый брокерской компанией клиентам для торговли. Все операции на нем осуществляются только на виртуальные средства, при этом котировки и сам процесс трейдинга – идентичен реальному рынку. Торговля на демо-счете – это вариант потренироваться и отработать стратегию, индикатор, новый подход в торговле [2].

1.2 Сравнительный анализ продуктов для торговли на криптобирже

В настоящее время существует достаточно большое количество приложений, предоставляющие возможности торговли криптовалютой, просмотра статистики изменений валюты и анализа открытия ордера на покупку. При этом, учитывая, что количество желающих научиться торговать и зарабатывать деньги в данной сфере деятельности растет с каждым годом, необходимо обеспечить новых трейдеров программным обеспечением, которое позволит пользователю научиться базовым навыкам торговли, не

рискуя своим капиталом. Именно поэтому торговля на демо-счете является одной из самых важных функций приложений для работы с криптовалютой.

Существующие приложения предоставляют функции в усложненном интерфейсе и с множеством различных анализаторов, которые могут показаться сложными новичкам. При этом не каждая криптобиржа позволяет практиковаться торговле на виртуальном счете без риска потерять свой капитал.

Тем не менее, существующие приложения обладают набором базовых функций, которыми удовлетворяют потребности современного пользователя. Самыми популярными продуктами-аналогами торговли криптовалютой являются: «Binance App», «Currency.com», «ByBit.com».

Все перечисленные приложения объединяет следующий набор функционала:

– покупка и продажа криптовалюты;

– просмотр статистики изменения валюты за заданный период времени;

– просмотр истории торговли.

Для обзора было подобрано три наиболее популярных продукта-аналога, которые будут рассмотрены далее.

1.2.1 Binance App

Binance – одна из самых популярных криптовалютных бирж в мире на данный момент. Ею можно пользоваться не только с помощью веб-браузера и компьютера, но и с помощью Android приложения. В приложении есть все те же функции, что и на компьютере. Приложение торговли на бирже Binance написано под платформу Windows «Binance App». Компания-разработчик предоставляет бесплатный продукт, который устанавливается на ПК с операционной системой Windows и телефоны с операционной системой IOS, Android [3]. Интерфейс приложения изображен на рисунке 1.

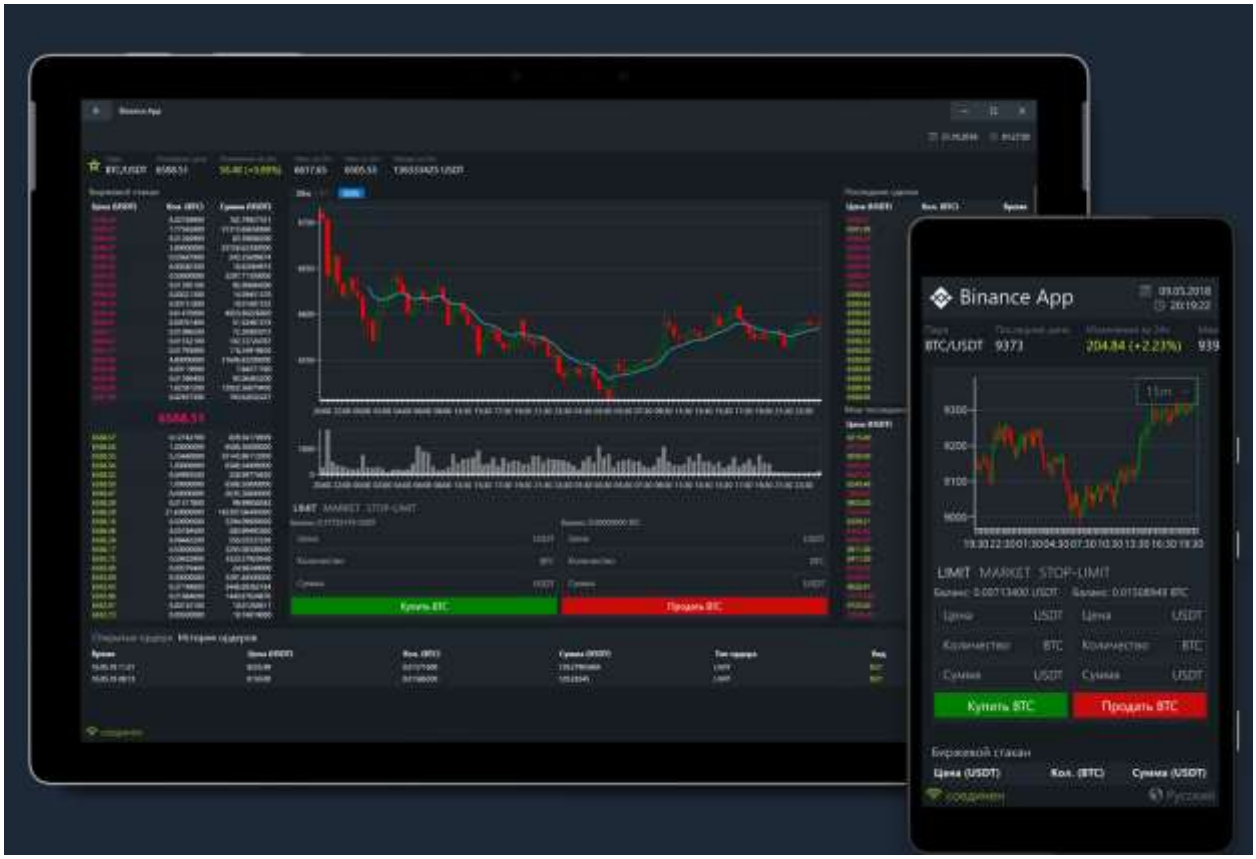

Рисунок 1 – Интерфейс приложения Binance App

Преимущества приложения:

– большой выбор валют, что обеспечивает высокий уровень ликвидности и привлекает к площадке пользователей редких альткоинов;

– высокая скорость работы, техническая возможность ежесекундно обрабатывать тысячи транзакций.

Основные недостатки приложения:

– усложненный интерфейс;

– только реальная торговля;

– отсутствие анализа для определенных видов торговли.

1.2.2 Currency.com

Currency.com (ООО «Карренси Ком Бел») зарегистрирована в республике Беларусь и является резидентом минского «Парка высоких технологий» [4]. Currency.com стала первой криптовалютной биржей, начавшей работу в Беларуси. Торги на бирже стартовали в январе 2019 года. Интерфейс приложения currency.com представлен на рисунке 2.

Рисунок 2 – Интерфейс приложения currency.com

Преимущества приложения:

– простой и понятный пользовательский интерфейс;

– мобильное приложение для устройств на базе операционной системы Андроид и iOS;

– большой выбор ордеров, в том числе рыночные, лимитные, возможность выставления стоп-лосс и тейк-профит.

Недостатки приложения:

– только пять криптовалют в листинге;

– непростой процесс верификации;

– негативные отзывы, связанные с зависанием платформы.

1.2.3 Сайт bybit.com

Официальный сайт биржи криптовалютных деривативов bybit.com был запущен в 2018 году. Это площадка, на которой торгуют фьючерсными контрактами на популярные цифровые монеты. Платформа ByBit – это биржа для маржинальной торговли криптовалютой. Это означает, что здесь можно заключать сделки с кредитным плечом. В качестве обеспечения торговых операций можно использовать относительно небольшой процент от первоначального размера транзакции.

Торговая платформа ByBit – это относительно неплохой вариант для начинающих трейдеров и опытных инвесторов. Она отличается интуитивно понятным и простым интерфейсом. В верхнем меню есть удобное переключение между кошельками, а также система управления аккаунтами. Можно также переключаться между фьючерсными контрактами Bitcoin и Ethereum [5].

Интерфейс сайта bybit.com представлен на рисунке 3.

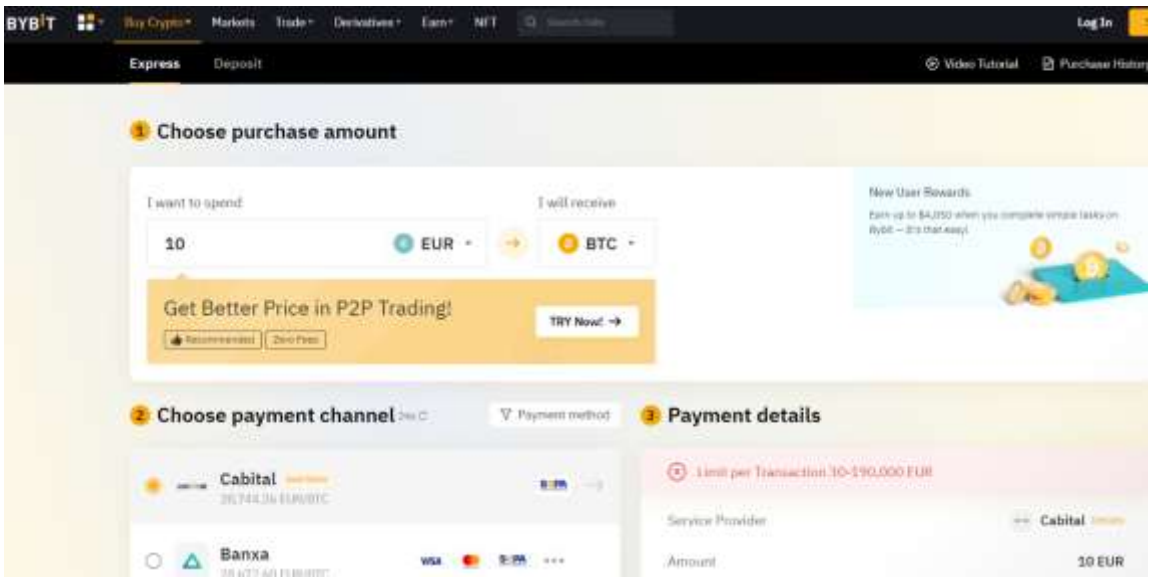

Рисунок 3 – Интерфейс bybit.com

Преимущества криптобиржи:

– простой и понятный интерфейс;

– высокая скорость исполнения торговых ордеров.

Недостатки криптобиржи:

– в списке доступных активов очень ограниченный выбор цифровых монет;

– нет мобильного приложения или десктопной программы для торговли;

– нет скидок на комиссии.

В связи с вышесказанным прослеживается необходимость разработки приложения, позволяющего вести торговлю криптовалютой на демо-счете, не рискуя своими настоящими деньгами, изучать поведение каждой из представленной криптовалюты, отрабатывать свою стратегию торговли для

извлечения большей прибыли. Разрабатываемое приложение, в отличие от большинства существующих программных продуктов, должно поддерживать торговлю на виртуальном счете, анализ открытия ордера на покупку и обнуление попытки при неудачной торговле.

1.2.4 Сравнительный анализ программных продуктов для торговли на криптобирже

Рассмотрев подробно каждый из аналогов, можно выделить главные элементы, которые должны находиться в каждой криптобирже:

- возможность торговать на демо-счете;
- статистика изменения валюты;
- окно для регистрации нового пользователя;
- статистика торгов каждого пользователя;
- анализатор покупки ордеров.

Возможность торговать на демо-счете позволит начинающему трейдеру отработать свои стратегии и попробовать торговлю на практике, не рискуя своим капиталом. Опытным трейдерам эта функция будет полезна для отработки новой стратегии. Имеется возможность пополнять демо-счет и обнулять его.

Статистика изменения валюты – график, на котором отображаются изменения определенного вида криптовалюты в различные промежутки времени. Так же для удобства считается волатильность каждого вида валют и предоставляется список актуальных курсов каждой валюты.

Окно для регистрации нового пользователя содержит поля для ввода персональных данных нового добавляемого в приложение пользователя.

Статистика торгов каждого пользователя – таблица с историей покупки и продажи ордеров, подсчитанным профитом от всех торгов и анализом прибыльности непосредственно самой стратегии.

Анализатор покупки ордеров – страница с полями для ввода данных о своем капитале, стоимости и количестве ордеров. На основе введенных данных и данных, взятых из статистики, определенной криптовалюты,

рассчитывает насколько выгодно будет торговать и стоит ли открывать ордер на покупку у брокера.

Мобильные и десктопные приложения позволяют вести торги из любого места, не открывая браузер. Это облегчает систему торговли для трейдера и повышает соответственно концентрацию на более важных деталях и индикаторах, которые так же должны предоставлять мобильные и десктопные приложения.

Среди критериев, по которым была проведена сравнительная характеристика аналогов, были выделены следующие:

- удобство интерфейса;
- безопасность;
- наличие демо-счета;
- анализатор ордеров;
- производительность.

Каждый из критериев является обязательным для обеспечения комфортной и более точной торговли. Рассмотрим подробно каждый из приведенных выше критериев.

Удобство интерфейса показывает, насколько эффективно конечный пользователь будет пользоваться основными функциями, предоставляемыми приложением, и насколько быстро пользователь будет ориентироваться в программе. Оценка критерия от 0 до 5.

Расшифровка оценок критерия представлена в таблице 1.

Таблица 1 – Расшифровка оценок критерия «Удобство интерфейса»

Оценка	Описание
0	Интерфейс не реализован.
1	Интерфейс максимально неудобен для использования, его логические элементы очень трудно воспринимать, и они расположены в порядке, максимально затрудняющем взаимодействие с ними.
2	Удобство интерфейса на низком уровне, имеются небольшие затруднения при взаимодействии с логическими элементами их трудно воспринимать.

Продолжение таблицы 1

3	Удобство интерфейса на среднем уровне, элементы взаимодействия расположены в логичной связке, при этом общее понимание полной функциональности приложения затруднено.
4	Интерфейс удобен для использования. Логические связи не нарушены, весь функционал доступен пользователю, при этом незначительные недостатки утомляют при долгом взаимодействии с приложением.
5	Интерфейс удобен для использования без каких-либо недостатков.

Безопасность показывает, какой уровень проверки доступа необходимо пройти идентифицируемому сотруднику, чтобы осуществить вход в приложение. Оценка критерия от 0 до 5. Расшифровка оценок критерия приведена в таблице 2.

Таблица 2 – Расшифровка оценок критерия «Безопасность»

Оценка	Описание
0	Все персональные данные пользователя в общем доступе
1	Персональные данные доступны только пользователю личного кабинета. При этом вход в личный кабинет не имеет никаких проверок, кроме ввода пароля и логина
2	Персональные данные доступны только пользователю личного кабинета. При этом вход в личный кабинет имеет защиту в виде капчи только от различных генераторов паролей, и соответственно вход осуществляется путем ввода пароля и логина
3	Персональные данные доступны только пользователю личного кабинета. При этом вход в личный кабинет имеет защиту в виде капчи от различных генераторов паролей, контрольного слова и восстановления данных по номеру телефона. Вход осуществляется по паролю и логину.
4	Персональные данные доступны только пользователю личного кабинета. При этом вход в личный кабинет имеет защиту в виде капчи от различных генераторов паролей, контрольного слова, восстановления данных по номеру телефона и двухэтапной верификации. Вход осуществляется вводов пароля, логина и кода, поступившего на номер телефона пользователя.

Продолжение таблицы 2

Оценка	Описание
5	Персональные данные доступны только пользователю личного кабинета. При этом вход в личный кабинет имеет защиту в виде капчи от различных генераторов паролей, контрольного слова, восстановления данных по номеру телефона и двухэтапной верификации. Вход осуществляется путем ввода пароля, логина, кода, поступившего на номер телефона пользователя и биометрии.

Наличие демо-счета показывает наличие возможности торговли на демо-счете. Оценка критерия от 0 до 1. Расшифровка оценок критерия приведена в таблице 3.

Таблица 3 – Расшифровка оценок критерия «Наличие демо-счета»

Оценка	Описание
0	Функция демо-счета не реализована
1	Функция демо-счета реализована

Производительность показывает, насколько большое количество транзакций позволяет приложение проводить пользователями. Оценка критерия от 0 до 2. Расшифровка оценок критерия представлена в таблице 4.

Таблица 4 – Расшифровка оценок критерия «Производительность»

Оценка	Описание
0	Приложение позволяет совершать одну транзакцию одному пользователю в определенный момент времени
1	Приложение позволяет совершать большое количество транзакций пользователям, при этом приложение начинает тормозить.
2	Приложение позволяет совершать большое количество транзакций пользователю без каких-либо проблем.

Анализатор ордеров показывает возможность анализа покупки ордеров с различными входными параметрами. Оценка критерия от 0 до 1. Расшифровка оценок представлена в таблице 5.

Таблица 5 – Расшифровка оценок критерия «Анализатор ордеров»

Оценка	Описание
0	Не имеется возможности анализа ордеров
1	Реализована возможность анализа ордеров

Наличие мобильных и десктопных приложений показывает возможность вести торги с телефона или ПК. Оценка критерия от 0 до 2. Расшифровка оценок приведена в таблице 6.

Таблица 6 – Расшифровка оценок критерия «Наличие мобильных и десктопных приложений»

Оценка	Описание
0	Не имеется мобильного или десктопного приложений
1	Имеется либо мобильное, либо десктопное приложение
2	Имеется и мобильное и десктопное приложеия

Результаты сравнения программ-аналогов по представленным выше характеристикам приведены в таблице 7.

Таблица 7 – Сравнительная характеристика продуктов-аналогов

Критерий	Binance App	Currency.com	ByBit
Удобство интерфейса	3	4	4
Безопасность	5	5	3
Наличие демо-счета	0	0	0
Анализатор ордеров	1	1	1
Производительность	2	2	2
Наличие мобильных и десктопных приложений	2	1	0
Оценка приложения по среднему арифмитическому	2.15	2.15	1.65

В результате сравнения программ-аналогов можно сказать, что многие рассмотренные продукты имеют различные преимущества друг перед другом,

при этом ни одна из программ не содержит в себе полный функционал, необходимый каждому трейдеру. Каждый из представленных программных продуктов не имеет демо-счета и только криптобиржа Binance предоставила доступ пользователям к мобильным и десктопным приложениям. С точки зрения безопасности каждая из рассмотренных программ, кроме ByBit, соответствует требованиям конфиденциальности и защищенной авторизации пользователя.

Рассмотренные программы показывают хорошую производительность и позволяют производить большое число транзакций в секунду. По итоговой оценке, можно сделать выводы, что приложения Binance App и Currency больше подходят трейдерам для проведения торгов.

Разрабатываемое приложение должно обеспечить людям удобный доступ к просмотру статистики поведения курса криптовалюты за различные временные отрезки или выбору необходимой из предоставленных криптовалют. Благодаря этому пользователь сможет анализировать рынок, прогнозировать поведение валюты и оценивать риски. Также пользователю будет полезен анализатор, помогающий принимать решение относительно покупки криптовалюты в определенном количестве и при определенном личном капитале.

В приложении также должна быть реализована возможность торговать криптовалютой на демо-счете. Это позволит пользователю изучить систему торговли на собственном опыте, не рискуя собственными деньгами. Каждая покупка и продажа будут сохранены в статистику, которая будет отображать профит стратегии каждого пользователя индивидуально.

Если профит слишком низкий или отрицательный, программа, анализируя статистику покупки и продаж, будет подсказывать какой стратегии лучше придерживаться пользователю.

1.3 Постановка задачи

В результате анализа предметной области было принято решение разработать десктопное приложение под операционную систему Windows 10.

Основная задача приложения торговли криптовалютой на демо-счете состоит в том, чтобы предоставить пользователю возможность покупать и продавать валюту через виртуальный кошелек, просматривать свою историю трейдинга и статистику изменения каждой криптовалюты.

Так как программа направлена на пользователей, желающих впервые познакомиться с торговлей криптовалютой и на тех трейдеров, кто хочет отработать свою стратегию, интерфейс должен быть простым для понимания.

Для надежного хранения и обработки информации необходимо разработать базу данных (БД), содержащую информацию о пользователях, криптовалюте, историю торговли пользователя и историю изменения криптовалюты. Приложение должно обладать следующим функционалом:

- регистрация и авторизация пользователей;
- просмотр личных данных пользователя;
- просмотр истории торговли;
- просмотр статистики изменения валюты за заданный промежуток времени;
- покупка и продажа криптовалюты;
- обнуление попытки;
- анализ открытия ордера на покупку.

В процессе проектирования и разработки необходимо предусмотреть модуль автоматического или автоматизированного добавления актуальных данных в таблицу истории криптовалюты.

1.4 Анализ требований

1.4.1 Функциональные требования

Для отражения функциональных требований к разрабатываемому мобильному приложению была спроектирована модель в форме диаграммы вариантов использования в нотации UML. Диаграмма прецедентов является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Диаграмма вариантов использования представлена на рисунке 4.

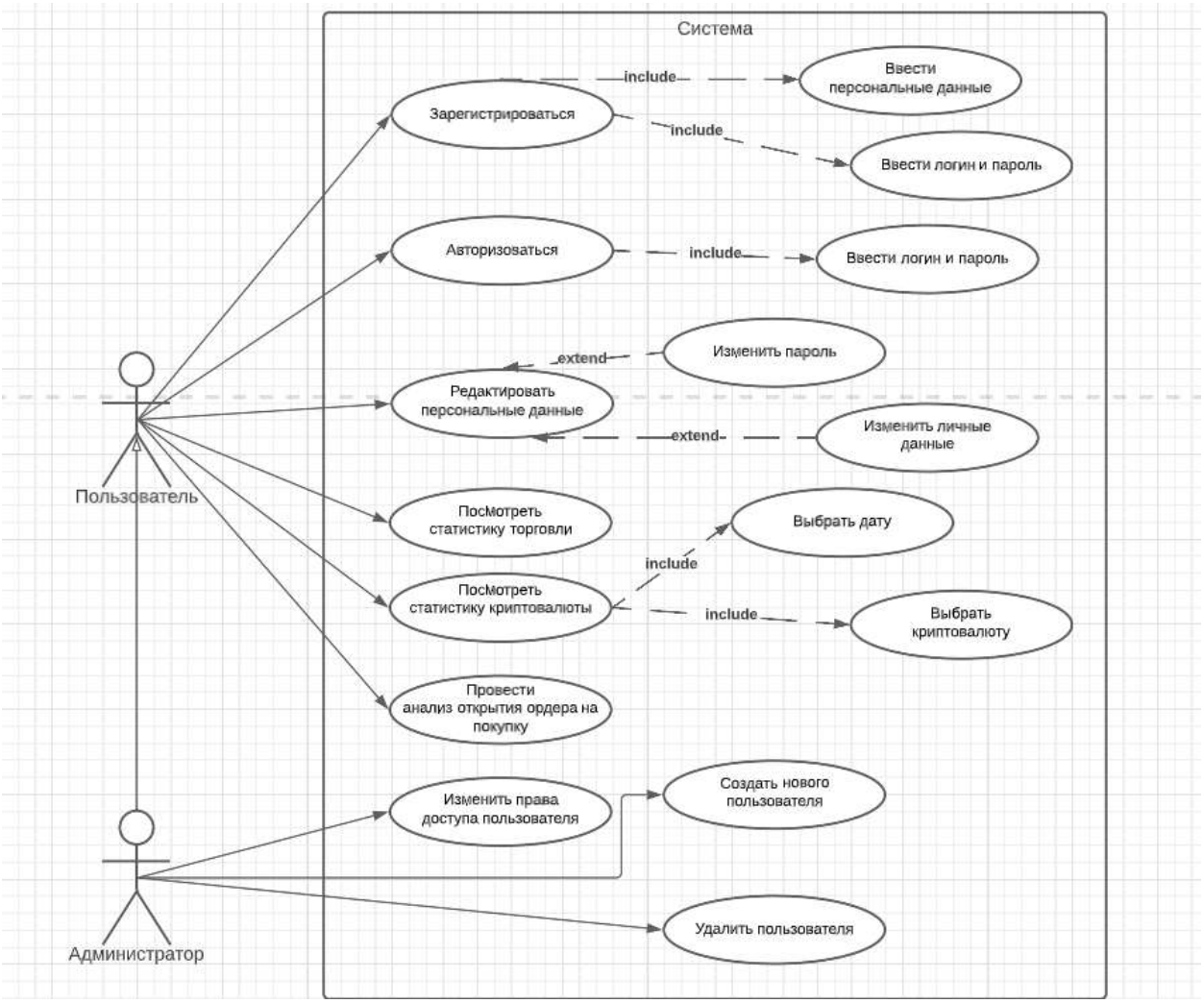

Рисунок 4 – Диаграмма вариантов использования

Для отображения связи между действующими лицами и вариантами использования на диаграмме используются отношения ассоциации, включения, расширения и обобщения [6].

Приложение должно обеспечивать выполнение следующих функций:

– регистрация;
– авторизация;
– редактирование персональных данных;
– просмотр статистики торговли;
– просмотр статистики криптовалюты;
– проведение анализа открытия ордера на покупку;
– изменение прав доступа пользователя;
– создание нового пользователя;
– удаление пользователя.

Спецификация прецедента «Зарегистрироваться» представлена на рисунке 5.

Зарегистрироваться
Id: 1
Описание: Пользователь регистрируется в системе
Главные актеры: Пользователь, система
Второстепенные актеры: Нет
Предусловие: Нет
Основной поток: 1. Если пользователь вводит некорректные данные 1.1. Система выдает предупреждение на той строке, которая введена неверно 2. Иначе, если пользователь с данной почтой или логином уже существует в системе 2.1. Система выдает предупреждение, что такой пользователь уже существует 3. Иначе 3.1. Пользователь добавляется в систему
Постусловие: Пользователь добавлен в БД системы

Рисунок 5 – Спецификация прецедента «Зарегистрироваться»

Прецедент «Зарегистрироваться» – описывает способ создания аккаунта пользователя.

На рисунке 6 представлена спецификация прецедента «Авторизоваться».

Авторизоваться
Id: 2
Описание: Пользователь авторизуется в системе
Главные актеры: Пользователь, система
Второстепенные актеры: Нет
Предусловие: Нет
Основной поток: 1. Если пользователь вводит некорректные логин или пароль 1.1. Система выдает предупреждение «Bad credential» 2. Иначе, если пользователь с данной почтой или логином существует в системе 2.1. Пользователь входит в личный кабинет
Постусловие: Пользователь авторизован в системе

Рисунок 6 – Спецификация прецедента «Авторизация»

Прецедент «Авторизация» – описывает способ авторизации пользователя в системе.

На рисунке 7 представлена спецификация прецедента «Редактировать персональные данные».

Редактировать персональные данные
Id: 3
Описание: Пользователь меняет персональные данные
Главные актеры: Пользователь, система
Второстепенные актеры: Нет
Предусловие: Нет
Основной поток: 1. Пользователь вводит новые данные 2. Изменения сохраняются в БД
Постусловие: Информация пользователя изменена в БД системы

Рисунок 7 – Спецификация прецедента «Редактировать персональные данные»

Прецедент «Редактировать персональные данные» описывает способ редактирования данных пользователя.

На рисунке 8 представлена спецификация прецедента «Посмотреть статистику торговли».

Посмотреть статистику торговли
Id: 4
Описание: Пользователь заходит в личный кабинет под своими данными и открывает страницу торговли
Главные актеры: Пользователь, система
Второстепенные актеры: Нет
Предусловие: Нет
Основной поток: 1. Пользователь входит в личный кабинет 2. Пользователь переходит на страницу торговли 3. Пользователю выдается таблица истории торговли
Постусловие: Информация пользователя изменена в БД системы

Рисунок 8 – Спецификация прецедента «Посмотреть статистику торговли»

Прецедент «Посмотреть статистику торговли» описывает способ просмотра истории торговли пользователя за весь период трейдинга.

На рисунке 9 представлена спецификация прецедента «Посмотреть статистику криптовалюты».

Посмотреть статистику криптовалюты
Id: 5
Описание: Пользователь открывает страницу криптовалюты и смотрит на график
Главные актеры: Пользователь, система
Второстепенные актеры: Нет
Предусловие: Нет
Основной поток: 1. Пользователь открывает страницу криптовалюты 2. Пользователь выбирает период с какого по какое число отображать изменения криптовалюты 3. Пользователь выбирает вид криптовалюты 4. Отображается график статистики выбранной криптовалюты
Постусловие: Статистический график криптовалюты построен

Рисунок 9 – Спецификация прецедента «Посмотреть статистику криптовалюты»

Прецедент «Посмотреть статистику криптовалюты» описывает способ просмотра статистики криптовалюты.

На рисунке 10 представлена спецификация прецедента «Провести анализ открытия ордера на покупку».

Провести анализ открытия ордера на покупку
Id: 5
Описание: Пользователь открывает страницу анализа и проводит анализ
Главные актеры: Пользователь, система
Второстепенные актеры: Нет
Предусловие: Нет
Основной поток: 1. Пользователь открывает страницу анализа 2. Пользователь выбирает период с какого по какое число рассчитывать волатильность криптовалюты 3. Пользователь выбирает вид криптовалюты 4. Пользователь вводит свой капитал 5. Система выдает результат анализа на основе введенных данных
Постусловие: Система выдала результат анализа

Рисунок 10 – Спецификация прецедента «Провести анализ открытия ордера на покупку»

Прецедент «Провести анализ открытия ордера на покупку» описывает способ проведения анализа открытия ордера на покупку.

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования были представлены их спецификации, которые позволяют отобразить последовательность действий пользователя и системы при выполнении определенных функций.

1.4.2 Нефункциональные требования

Серверная часть приложения должна быть выполнена на языке Java с использованием веб-фреймворка Spring MVC и фреймворка Spring Security.

Spring Web MVC Framework предлагает архитектуру модель-представление-контроллер и предлагает компоненты, которые помогают вам быть гибкими и создавать слабо связанные веб-приложения. Шаблон MVC позволяет разделять различные аспекты приложения, предлагая слабую связь между этими элементами.

Spring Security – это фреймворк, который сфокусирован на обеспечение как аутентификации, так и авторизации в Java-приложениях. Как и все Spring проекты, настоящая сила Spring Security в том, что он может быть легко дополнен нужным функционалом.

Язык Java был выбран так, как он больше подходит для Web-разработки и имеет платформу JavaFX, которая обеспечивает простое создание GUI-приложений. В качестве СУБД будет использоваться MySQL. Для работы с БД необходимо использовать ORM Hibernate. Клиентская часть приложения будет реализована на языке JavaScript с использованием фреймворка JQuery. При разработке необходимо использовать редактор исходного кода Intellij IDEA.

2 Проектирование приложения по работе с криптовалютой на демо-счете

2.1 Архитектура приложения

Архитектура приложения торговли криптовалютой на демо-счете основана на REST архитектуре и шаблоне проектирования MVC. MVC архитектура является наиболее распространенной при проектировании web-приложений, так как разделения на слои Model, View, Controller естественно ложится на структуру подобных сервисов. На рисунке 11 представлена схема взаимодействия приложений с REST API, построенного по MVC архитектуре.

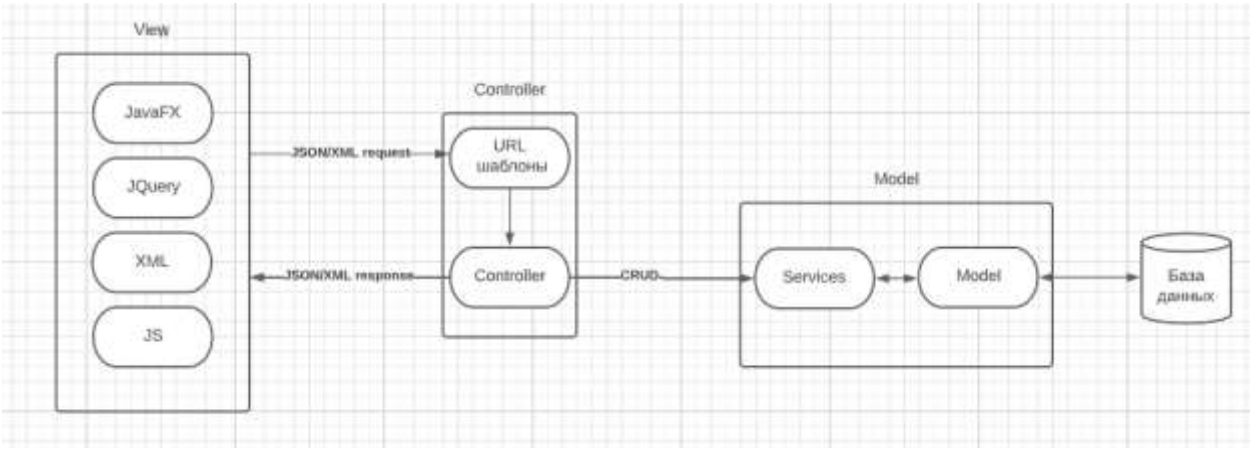

Рисунок 11 – Схема взаимодействий приложений с API в MVC архитектуре

Из данной модели видно, что данный сервис основан на трехуровневой модели, состоящей из клиента, сервера приложений и сервера базы данных.

Клиентом в данной системе является браузер и приложение, написанное под ОС Windows 10.

2.2 Проектирование структур данных

2.2.1 Логическая модель базы данных приложения

На основе проведенного анализа предметной области и функциональных требований была построена логическая модель базы данных с использованием нотации «сущность-связь». Логическая модель отражает суть выбранной предметной области и выделяет основные сущности, атрибуты и связи в системе.

В начале необходимо выделить список всех основных сущностей данной предметной области. Данный список представлен в таблице 8.

Таблица 8 – Описание сущностей предметной области

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Пользователь	Списочный состав пользователей, зарегистрированных в системе	Стержневая	100	10% в неделю
2	Роли	Список ролей с их разрешениями	Вспомогательная	2	Очень редко
3	Роли пользователей	Список ролей и принадлежащих к ним пользователей	Вспомогательная	100	10% в неделю
4	Криптовалюта	Списочная информация о каждой криптовалюте, записанной в систему	Стержневая	10	Очень редко
5	История криптовалюты	Список изменений каждого вида представленной валюты	Стержневая	10000	24 записи в день
6	История торговли	Список данных о покупке и продаже определенного вида криптовалюты пользователем	Стержневая	100	5% в день

На основании данных сущностей необходимо построить логическую модель базы данных. Логическая модель данных является начальным прототипом будущей базы данных. Она строится в терминах информационных единиц, но без привязки к конкретной СУБД. Для логической модели данных характерно то, что, выполняя все основные требования, предъявляемые СУБД, не поддерживается ориентация на конкретную СУБД, что реализуется в физической модели данных.

Для создания модели использовался язык IDEF1X с использованием CASE-средства Visual Paradigm. На рисунке 12 изображена построенная логическая модель сервиса.

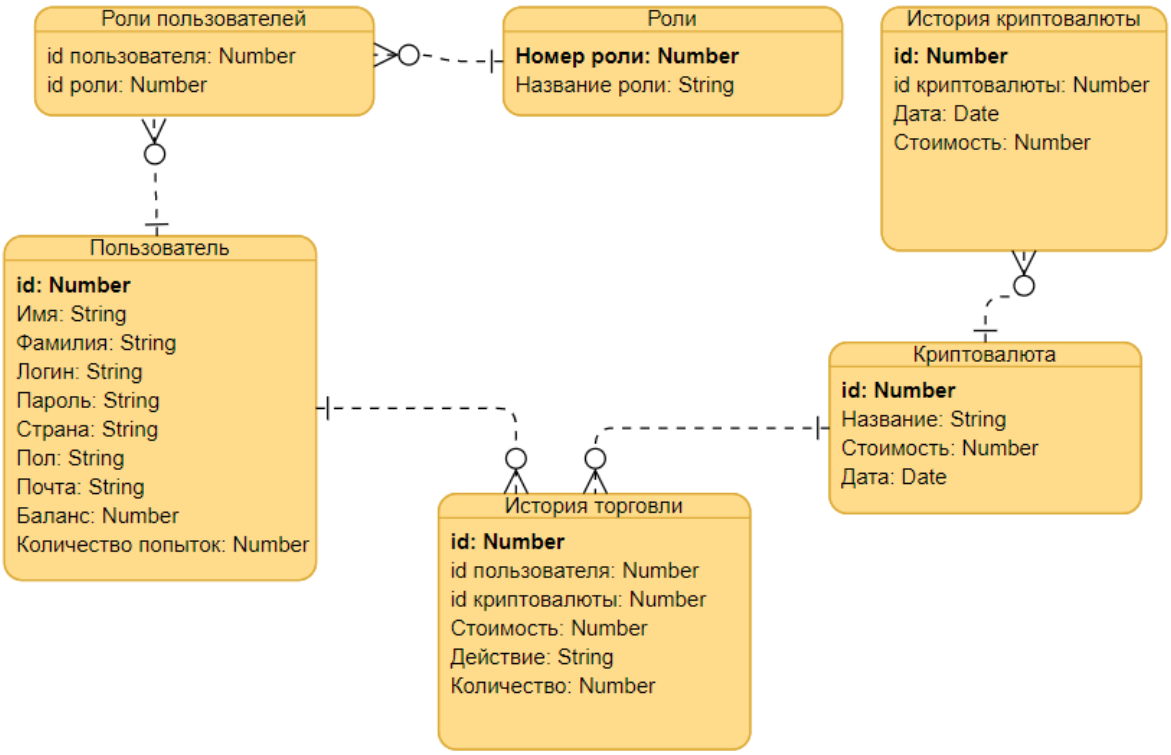

Рисунок 12 – Логическая схема БД

От данного уровня модели БД нужно перейти к физическому.

2.2.2 Физическая модель базы данных

Физическая модель – логическая модель базы данных, выраженная в терминах языка описания конкретной СУБД. Физические модели баз данных определяют способы размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне. Если физическая модель данных реализована средствами реляционной СУБД, то отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся столбцами таблиц, для ключевых атрибутов создаются уникальные индексы, домены преображаются в типы данных, принятые в конкретной СУБД. В качестве СУБД будет использован MySQL.

На основании логической модели была разработана физическая модель базы данных. Физическая модель БД в нотации IDEF1X представлена на рисунке 13.

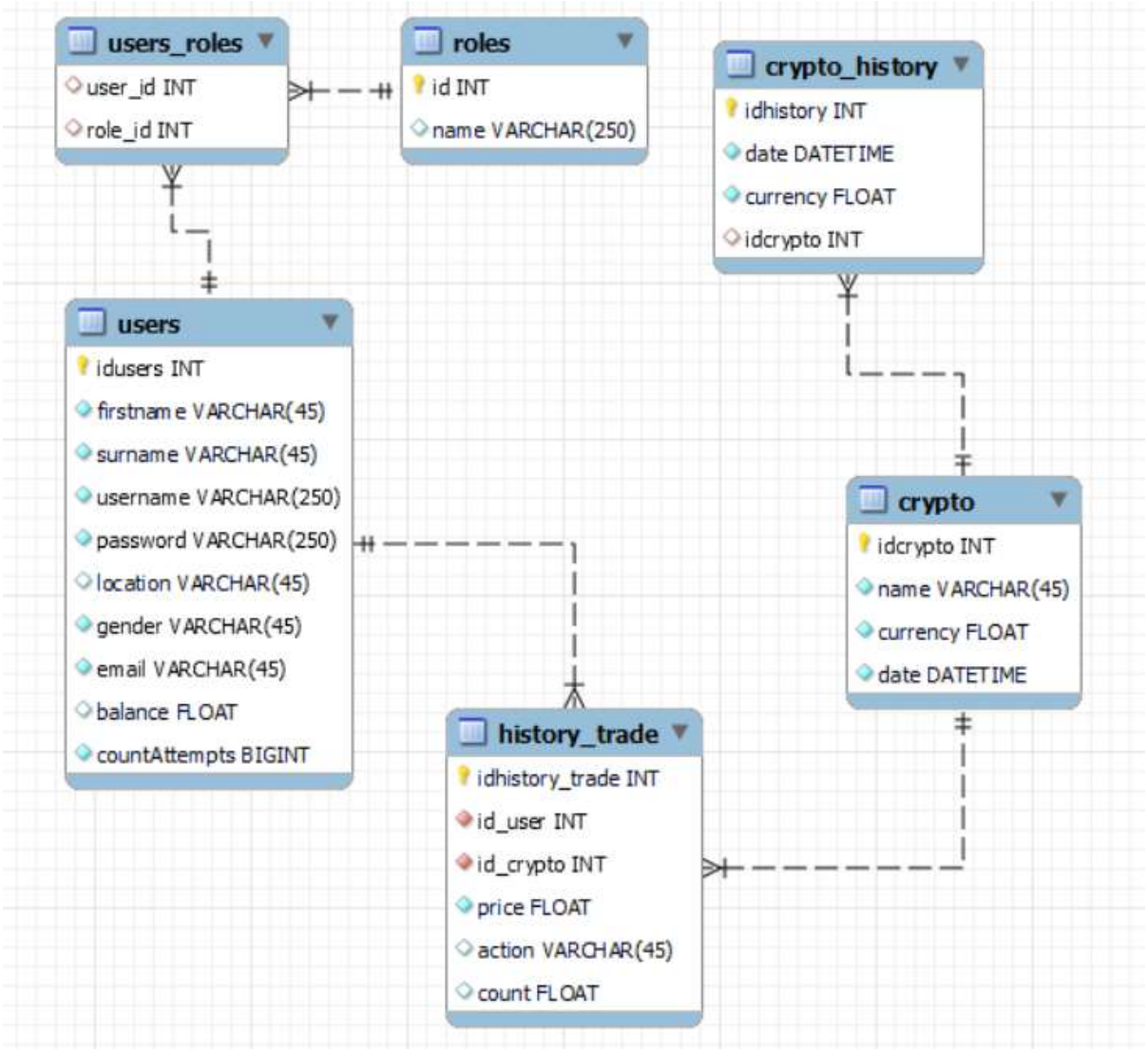

Рисунок 13 – Физическая схема БД

В процессе создания физической модели было принято решение использовать для создания модели БД миграции, а для описания БД ORM Hibernate. ORM основана на объектном подходе – представлении таблиц БД в виде классов. Поэтому построенную физическая модель необходимо описать в терминах ORM [7].

2.3 Проектирование программных средств

2.3.1 Проектирование классов приложения торговли криптовалютой на демо-счете

Согласно модели предметной области, были спроектированы диаграммы программных классов двух модулей, на которые разбита программа. Первый модуль отвечает за авторизацию, регистрацию

пользователя, изменение своих данных. Так же пользователям с уровнем доступа «Администратор» доступно удаление пользователей, просмотр всех пользователей, редактирование информации, создание нового пользователя и изменение его прав доступа. Диаграммы классов первого и второго модулей программы изображены в приложении Г.

Краткое описание представленных на диаграмме классов приведено в таблице 9.

Таблица 9 – Описание классов 1 модуля

№	Класс	Описание
1.	User	Класс сущности User, реализующий таблицу пользователей в БД.
2.	DataBaseHandler	Класс для подключения и отправки запросов к БД.
3.	Configs	Вспомогательный класс для класса DataBaseHandler, описывающий конфигурационные параметры подключения к БД.
4.	UserController	Класс-контроллер для пользователя, реализующий отправку GET и POST запросов по ссылкам.
5.	Const	Класс, содержащий константы программы.
6.	Role	Класс сущности Role, реализующий таблицу ролей в БД.
7.	UserService	Класс-сервис, реализующий интерфейс UserRepository и необходимый для CRUD-операций над пользователями.
8.	AdminController	Класс-контроллер для администратора, реализующий отправку GET и POST запросов по ссылкам.
9.	ThreadData	Класс-нить, запускаемый параллельно с запуском программы. Реализует обновление данных криптовалюты в БД.
10.	RoleRepostiory	Интерфейс, содержащий CRUD-операции для БД.
11.	UserRepository	Интерфейс, содержащий CRUD-операции для БД.
12.	RestUserController	Класс, реализующий Rest-контроллеры.
13.	Crypto	Класс, описывающий таблицу с данными криптовалют.
14.	ParserSite	Класс, отвечающий за подключение и парсинг к сторонней странице с актуальными данными стоимости криптовалюты.

Продолжение таблицы 9

№	Класс	Описание
15.	WebSecurityConfig	Данный класс используется для конфигурации защиты для приложения.
16.	UserDetailServiceImpl	Класс, реализующий интерфейс UserDetailsService. Интерфейс UserDetailsService используется для извлечения данных, связанных с пользователем. У него есть один метод с именем loadUserByUsername (), который можно переопределить, чтобы настроить процесс поиска пользователя.
17.	RoleService	Класс-сервис, реализующий интерфейс RoleRepository и необходимый для CRUD-операций над ролью.
18.	ProcessingArrayForParserSite	Вспомогательный класс для процесса парсинга информации со стороннего сайта.
19.	SuccessUserHandler	Класс, реализующий интерфейс AuthenticationSuccessHandler, описывающий правила авторизации для пользователей с разными уровнями доступа.
20.	PasswordEncoderConfig	Класс, реализующий BCryptPasswordEncoder() для кодирования и сравнения хешей паролей.
21.	MvcConfig	Конфигурационный класс, использующий форму JavaBean.
22.	SpringBootSecurityDemoApplication	Точка входа в программу.

В первом модуле классы можно разделить на две группы: классы, реализующие Spring Security и классы, реализующие парсинг страницы.

Краткое описание представленных на диаграмме классов приведено в таблице 10.

Таблица 10 – Описание классов 2 модуля

№	Класс	Описание
1.	Boss	Класс главной формы, в которой отображаются другие окна при нажатии на соответствующие кнопки.
2.	Trading	Класс окна торговля. Окно открывается при нажатии соответствующей кнопки в окне главной формы. Состоит из двух таблиц и данных о балансе и количестве попыток пользователя.
3.	Analyze	Класс окна анализ. Состоит из нескольких полей ввода и кнопок предварительного анализа и расчета.
4.	CryptoWindow	Класс окна криптовалюта. Содержит актуальные данные стоимости криптовалют и график с историей изменения стоимости валюты за заданный период времени.

Продолжение таблицы 10

№	Класс	Описание
5.	Const	Класс, содержащий константы программы.
6.	Configs	Класс, содержащий конфигурационные данные для подключения к БД.
7.	DataBaseHandler	Класс для подключения и отправки запросов к БД.
8.	Crypto	Класс, описывающий таблицу с данными криптовалют.
9.	Main	Точка входа в программу.
10.	MainWindow	Класс пользовательского окна личных данных. Содержит персональную информацию о пользователе.
11.	User	Класс сущности User, реализующий таблицу пользователей в БД.
12.	Controller	Класс окна авторизации. Проверят на наличие пользователя в БД по id и имени.
13.	BalanceAllCryptoUser	Класс, содержащий информацию о балансе криптовалют всех пользователей.
14.	CryptoHistoryUser	Класс, содержащий информацию об истории криптовалюты.

Условно классы в данном модуле приложения можно разделить на две группы: классы, описывающие бизнес-логику и отображение каждой страницы, и вспомогательные классы.

2.3.2 Проектирование алгоритма торговли на демо-счете

С помощью диаграммы деятельности можно разбить некоторую длительную операцию на составные части. При этом отображается последовательность данных операций, где каждое переход от одной операции срабатывает только при завершении операции в предыдущем состоянии.

Одной из важнейших функций является торговля на демо-счете. Данная функция является основной для обучения трейдингу. Вся история торговли сохраняется в базу данных, что позволяет отслеживать свои действия по каждой криптовалюте.

Пользователь перед ведением торговли должен авторизоваться и получить доступ к своему виртуальному кошельку и балансу криптовалют. После авторизации пользователь должен перейти на страницу трейдинга и ввести данные о том, что именно он хочет сделать. Изначально система

проверит введенные данные и не позволит пользователю потратить больше денег, чем есть в кошельке или продать больше криптовалюты, чем имеется на балансе. Диаграмма деятельности торговли на демо-счете представлена на рисунке 14.

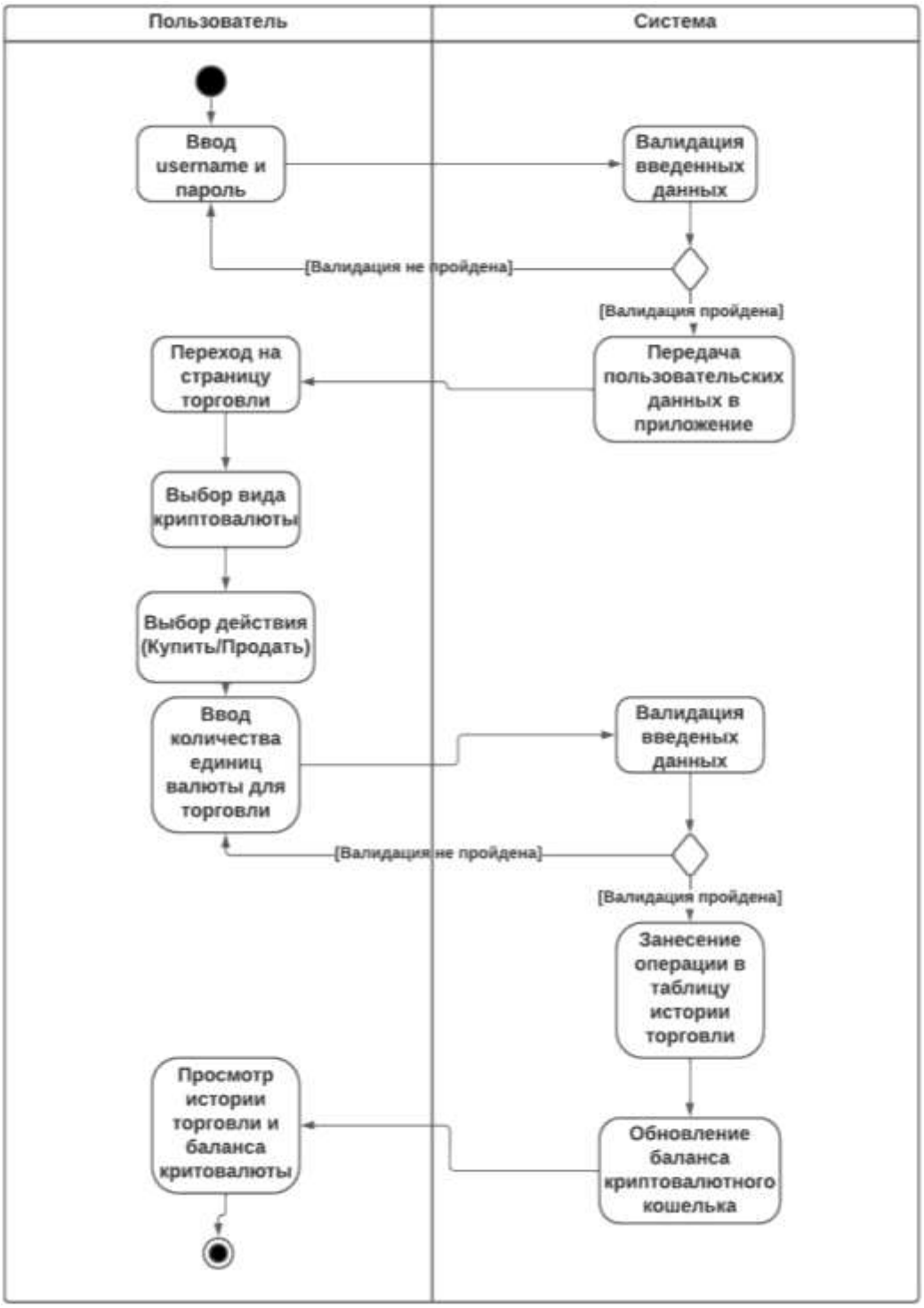

Рисунок 14 – Диаграмма деятельности «Торговля на демо-счете»

После валидации введенных данных торговли криптовалютой запись торговли будет передана БД.

2.3.3 Проектирование сценария торговли на демо-счете

Диаграмма последовательности описывает коммуникационный аспект между основными объектами системы в некотором сценарии [8]. На данной диаграмме описан процесс торговли на демо-счете (рисунок 15).

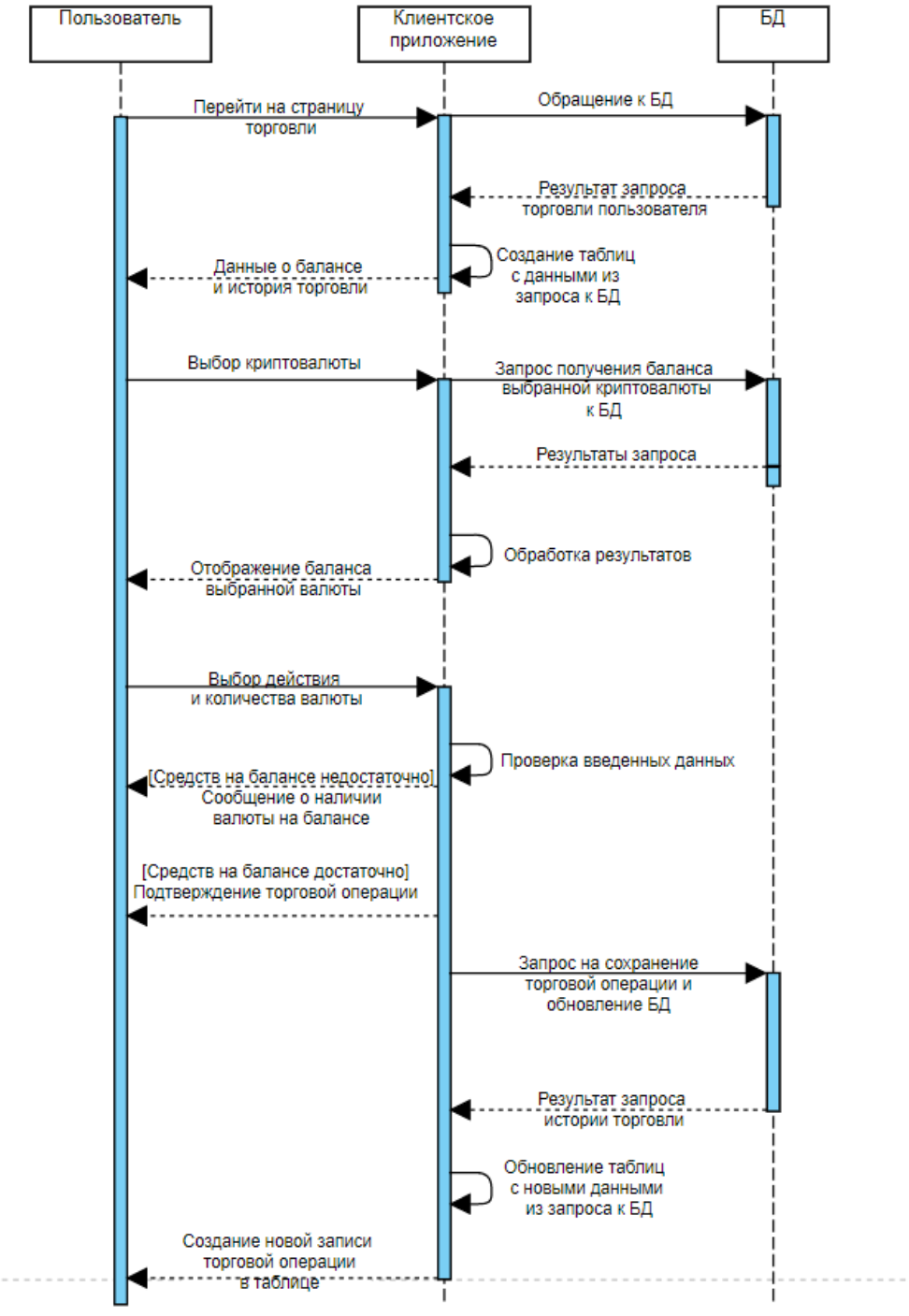

Рисунок 15 – Диаграмма последовательности процесса торговли на демо-счете

Эта диаграмма включает три основных объекта: пользователь, клиентское приложение и сервер БД. Первым действием пользователь

переходит на страницу торговли и видит данные, предоставленные базой данных.

После этого пользователь выбирает необходимое действие торговли (купить/продать), вид криптовалюты и количество валюты для совершения торговой операции. Клиентское приложение посылает запрос к базе данных, определяя баланс выбранного вида криптовалюты.

Как только пользователь совершает торговую операцию, она сохраняется в базу данных и отображается в таблице истории торговли.

2.3.4 Диаграмма кооперации приложения

Диаграмма кооперации описывает последовательность взаимодействия объектов и их структурные аспекты отношений. Объекты представлены в виде прямоугольников.

Динамические связи изображаются в виде линий, где стрелками указывается направление потоков сообщений между объектами, а порядковый номер устанавливает очередность взаимодействия.

Для отображения последовательности обмена сообщениями в процессе покупки билета была построена диаграмма кооперации, представленная на рисунке 16.

Рисунок 16 – Диаграмма кооперации торговли на демо-счете

Диаграмма описывает процесс ведения торговли на демо-счете в приложении. Она включает в себя три объекта: пользователь, приложение и сервер БД, которые обмениваются сообщениями.

Объект «Пользователь» выбирает вид валюты, вид торговой операции (купить/продать) и количество валютной единицы. Совершает торговую операцию и получает обновленную таблицу с добавленной записью операции.

Объект «Приложение» получает от пользователя данные о торговой операции, совершает расчет и отправляет запрос БД. После подтверждения наличия достаточного количества средств на балансе от сервера БД, приложение формирует и отображает обновленную таблицу.

Объект «Сервер БД» осуществляет проверку баланса пользователя.

2.4 Проектирование пользовательского интерфейса web-приложения

В результате разработки пользовательского интерфейса для клиентского приложения была разработана иерархия диалоговых окон. На рисунке 17 изображена иерархия диалоговых окон.

Рисунок 17 – Иерархия диалоговых окон

Для каждого диалогового окна, используемого в клиентском приложении, была спроектирована модель графического интерфейса с использованием встроенных средств среды разработки JavaFx. Для создания web-интерфейса был использован язык разметки HTML и язык скриптов JavaScript.

Для каждой модели интерфейса приведена подробная спецификация с пояснением каждого элемента интерфейса.

Модель, изображенная на рисунке 18, является главным и отображает все персональные данные пользователя.

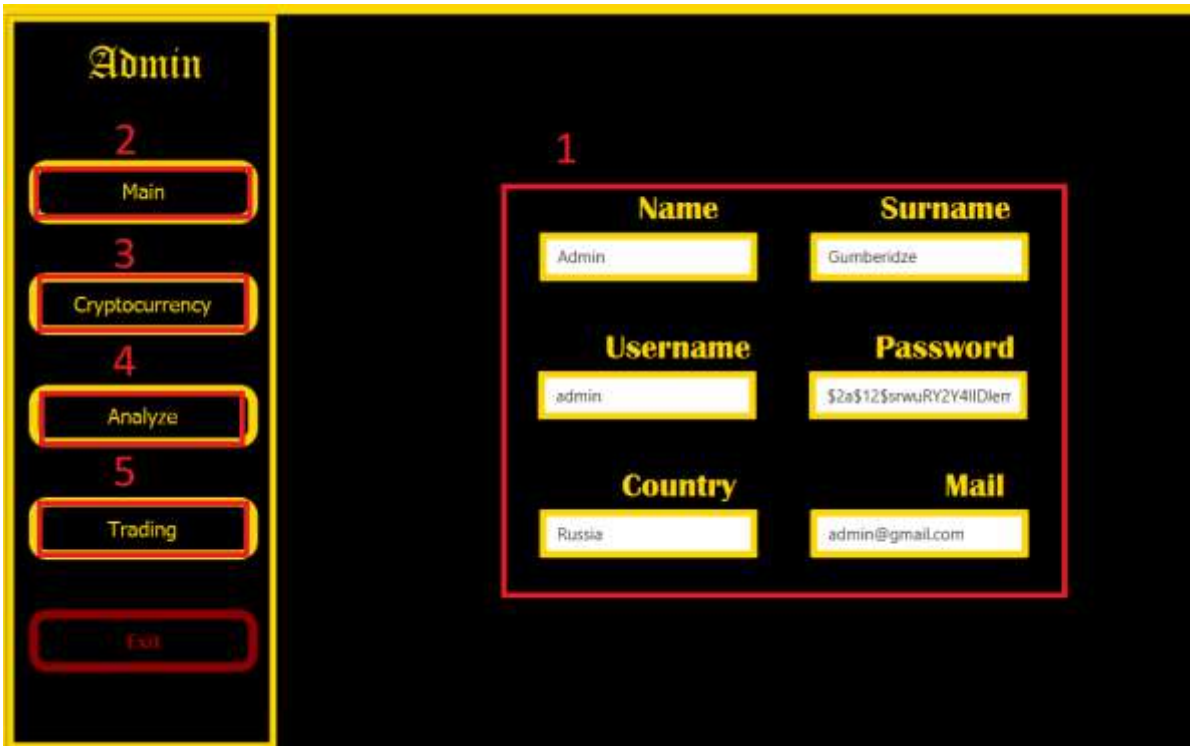

Рисунок 18 – Модель графического интерфейса (главное окно программы)

Описание элементов интерфейса:

1 – главное окно, содержащее данные пользователя из таблицы users;

2 – переход на главную страницу;

3 – переход на страницу актуальных данных и статистики криптовалюты, содержит данные из таблицы crypto;

4 – переход на страницу анализа криптовалюты, содержит данные из таблицы crypto и crypto_history;

5 – переход на страницу трейдинга и истории торговли пользователя, содержит данные из таблицы crypto и history_trade.

Изменение пользовательских данных происходит путем взаимодействия с web-интерфейсом приложения, который позволяет администратору менять роль пользователя, удалять и создавать новых участников.

Модель, изображенная на рисунке 19 отображает статистические данные криптовалюты.

Рисунок 19 – Модель графического интерфейса (Статистика и актуальные данные)

Описание элементов интерфейса:

1 – таблица с актуальными данными криптовалюты;

2 – график, отображающий статистику изменений криптовалюты;

3 – рассчитанный рейтинг изменения криптовалюты.

Модель на рисунке 20 отображает анализатор покупки ордера.

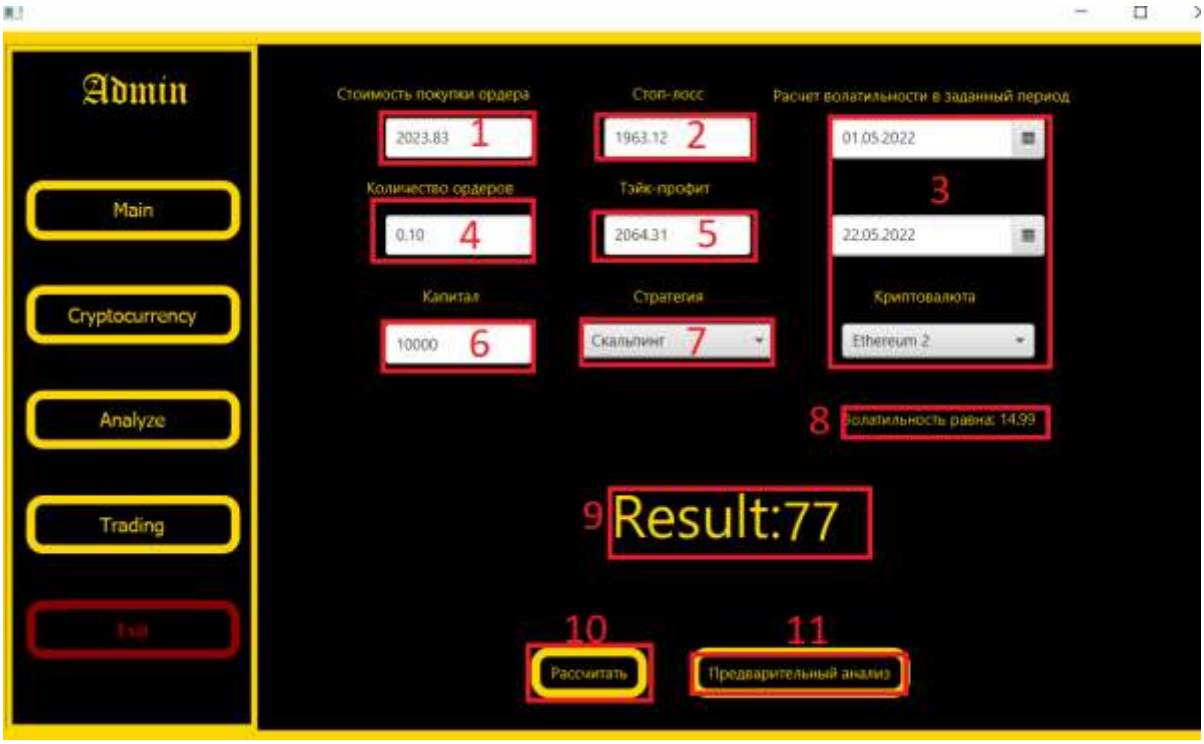

Рисунок 20 – Модель графического интерфейса (Анализ ордера)

Описание элементов интерфейса:

1 – текстовое поле, в котором отображается актуальная стоимость криптовалюты;

2 – текстовое поле, в котором отображается рассчитанный стоп-лосс;

3 – блок выбора даты для расчета волатильности в заданный период;

4 – текстовое поле, в котором отображается количество ордеров, которые допустимо купить;

5 – текстовое поле, в котором отображается рекомендуемый тейк-профит;

6 – текстовое поле, отображающее капитал;

7 – список выбора стратегии торговли;

8 – рассчитанная волатильность криптовалюты, учитывая историю ее изменений;

9 – результат расчета прибыльности;

10 – кнопка для расчета прибыльности сделки;

11 – кнопка для расчета полей «Стоп-лосс», «Тейк-профит», «Количество ордеров».

Рисунок 21 отображает историю торговли пользователя.

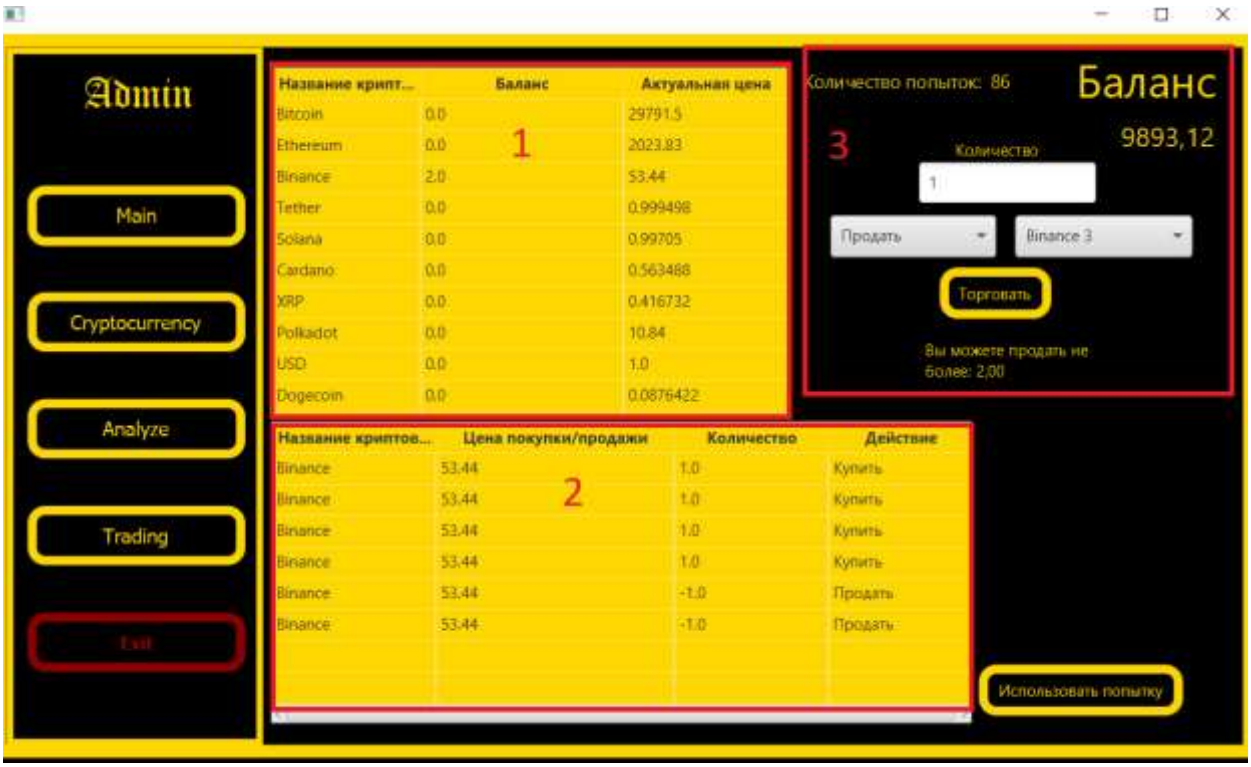

Рисунок 21 – Модель графического интерфейса (История трейдинга)

Описание элементов интерфейса:

1 – таблица с балансом каждой криптовалюты пользователя;

2 – таблица с историей торговли пользователя;

3 – блок, отвечающий за покупку и продажу криптовалюты.

На рисунке 22 изображен web-интерфейс приложения.

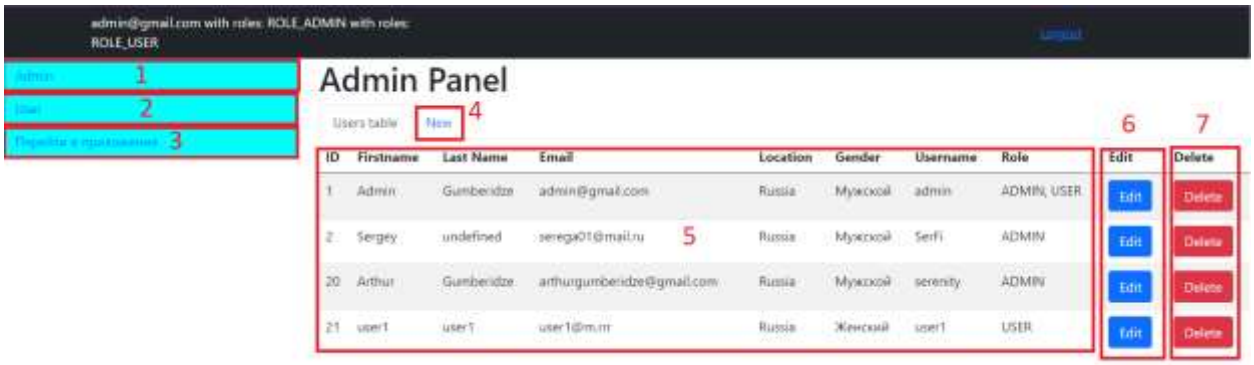

Рисунок 22 – Модель графического интерфейса (Панель администратора)

Описание элементов интерфейса:

1 – кнопка перехода на администраторскую панель;

2 – кнопка перехода на страницу личной информации;

3 – кнопка для открытия приложения с данными авторизованного пользователя;

4 – кнопка создания нового пользователя;

5 – таблица с пользователями, зарегистрированными в системе;

6 – кнопки для редактирования данных о пользователе;

7 – кнопки для удаления пользователя.

2.5 Проектирование клиентского приложения

Для написания клиентского приложения были выбраны фреймворк Spring и платформа JavaFX. Spring приложение упрощает работу с объектами, он автоматически создает их и связывает используя принцип инверсии контроля (IOC).

JavaFX позволяет создавать приложения с богатой насыщенной графикой благодаря использованию аппаратного ускорения графики и возможностей GPU.

3 Реализация программного обеспечения сервиса

Программные средства приложения для торговли криптовалютой на демо-счете реализовано на языке Java, с помощью платформы JavaFX для создания приложения GUI. Диаграмма компонентов, из которых состоит программное обеспечение клиентского приложения приведена на рисунке 23.

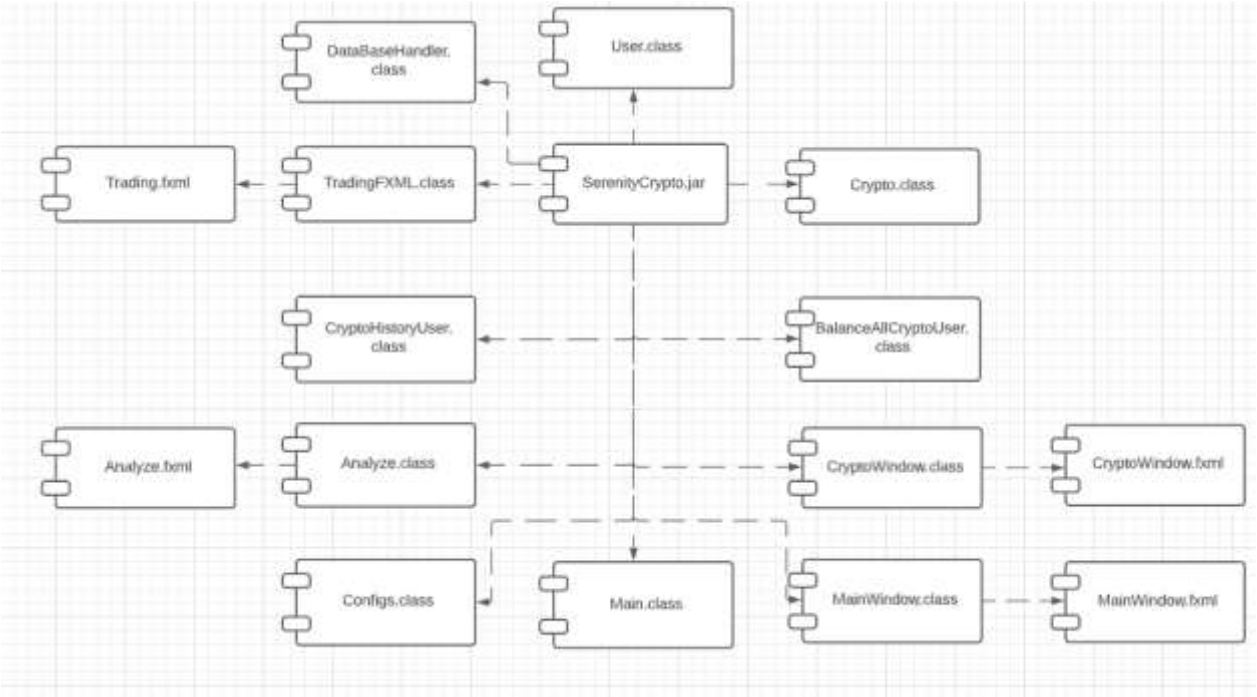

Рисунок 23 – Диаграмма компонентов

Рассмотрим каждый компонент:

– SerenityCrypto.jar – исполняемый файл программы, полученный в ходе сборки проекта;

– User.class – содержит данные о пользователе и реализацию класса User;

– DataBaseHandler.class – содержит логику взаимодействия с базой данных и реализацию класса DataBaseHandler;

– TradingFXML.class – содержит логику взаимодействия для окна «Трейдинг»;

– TradingFXML.fxml – содержит разметку для окна «Трейдинг»;

– MainWindow.class – содержит логику взаимодействия для окна «Главная страница»;

– MainWindow.fxml – содержит разметку для окна «Главная страница»;

– CryptoWindow.class – содержит логику взаимодействия для окна «Криптовалюта»;

– CryptoWindow.fxml – содержит разметку для окна «Криптовалюта»;

– Analyze.class – содержит логику взаимодействия для окна «Анализ»;

– Analyze.fxml – содержит разметку для окна «Анализ»;

– Crypto.class – содержит данные о криптовалюте и реализацию класса Crypto;

– CryptoHistoryUser.class – содержит данные об истории торговли каждой валютой пользователя и реализацию класса CryptoHistoryUser;

– BalanceAllCryptoUser.class – содержит данные о балансе каждой криптовалюты пользователя и реализацию класса BalanceAllCryptoUser;

– Configs.class – содержит данные для подключения к базе данных и реализацию класса Configs;

– Main.class – главный класс программы и реализация класса Main.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. Диаграмма развертывания является единой для системы в целом и всецело отражает особенности ее реализации [9].

Диаграмма развертывания приложения для торговли криптовалютой на демо-счете представлена на рисунке 24.

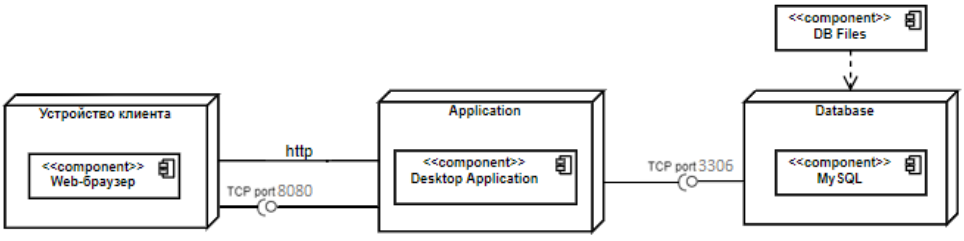

Рисунок 24 – Диаграмма развертывания

Архитектура состоит из трех основных составляющих: устройства клиента на платформе Windows, приложения, скомпилированного в jar файл и запускаемого из web-браузера и базы данных MySQL. С помощью web-браузера пользователь осуществляет взаимодействие непосредственно с приложением для торговли криптовалютой на демо-счете. Сервер MySQL позволяют принимать запросы пользователя, отправлять запросы на БД, получать ответ и возвращать его пользователю.

4 Контроль качества программного обеспечения

4.1 Функциональное тестирование

Функциональное тестирование методом черного ящика – это тестирование функциональных возможностей ПО, в условиях, когда внутренняя структура и реализация неизвестна.

Произведем функциональное тестирование создания нового пользователя. Пользователю обязательно должна быть присвоена роль. Почта обязательно должна содержать символ «@» и состоять из ограниченного символьного набора. Также пользователю присваивается свой псевдоним, который тоже должен пройти валидацию по длине. Не могут быть созданы пользователи под одним псевдонимом. При несоблюдении хотя бы одного из этих условий – пользователь не будет создан и зарегистрирован в системе. В таблице 11 представлены классы эквивалентности, отображающие требования к каждому входному условию [10].

Таблица 11 – Классы эквивалентности

Входные условия	Правильные классы эквивалентности	Неправильные классы эквивалентности
Длина псевдонима	>= 1 и <= 30	<1 или >30
Почта	Состоит из букв латинского алфавита, цифр, нижнего подчеркивания или тире и символа обязательного «@»	Содержит другие символы и не содержит символ «@»
Занятость псевдонима	Псевдоним свободен	Псевдоним занят
Роль	Присвоена роль пользователю	Роль не присвоена

Определим граничные условия. Длина псевдонима: $n \in [1, 30]$. Следовательно, длины 1 и 30 находятся на границах допустимых значений. Эти значения следует проверить. Также имеет смысл проверить ближайшие неверные значения 0 и 31.

Граничные условия для состава символов в названии почты размыты. Следует проверить работу программы на корректных последовательностях символов в названии почты, так и последовательности, содержащие некорректные символы.

Для признака занятости псевдонима существует только два возможных варианта. Либо псевдоним занят, либо свободен. Следует протестировать оба варианта.

Для проверки признака наличия роли у пользователя существует только два возможных варианта. Либо роль присвоена, либо нет. Следует протестировать оба варианта.

Для процесса создания нового пользователя была построена функциональная диаграмма, представленная на рисунке 25.

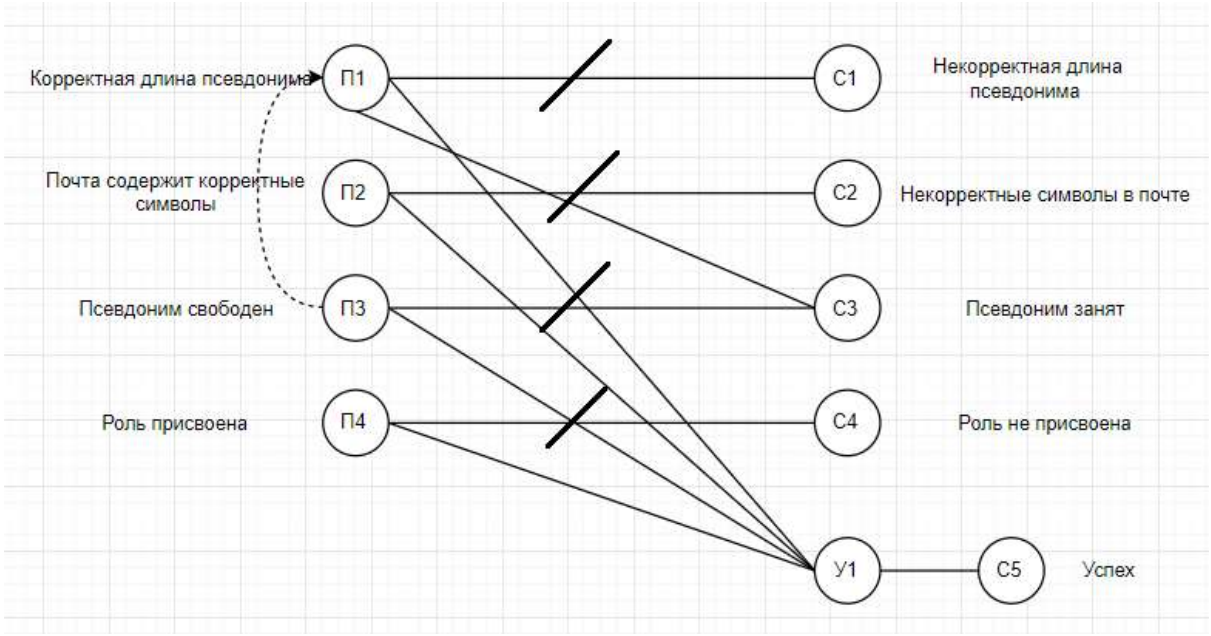

Рисунок 25 – Функциональная диаграмма

Причины:

– корректная длина псевдонима;

– почта содержит корректные символы;

– псевдоним свободен;

– роль присвоена.

Следствия:

– сообщение об ошибке «Неверная длина псевдонима»;

– сообщение об ошибке «Некорректные символы в названии почты»;

– сообщение об ошибке «Псевдоним занят»;

– сообщение об ошибке «Роль не присвоена»;

– успех.

На основании функциональной диаграммы была построена таблица решений. Таблица решений представлена в таблице 12.

Таблица 12 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12
П1	0	0	0	0	1	1	1	1	1	1	1	1
П2	0	0	1	1	0	0	0	0	1	1	1	1
П3	-	-	-	-	0	0	1	1	0	0	1	1
П4	0	1	0	1	0	1	0	1	0	1	0	1
С1	1	1	1	1	0	0	0	0	0	0	0	0
С2	1	1	0	0	1	1	1	1	0	0	0	0
С3	-	-	-	-	1	1	0	0	1	1	0	0
С4	1	0	1	0	1	0	1	0	1	0	1	0
С5	0	0	0	0	0	0	0	0	0	0	0	1

В таблице 13 представлен набор тестов, составленные на основе таблицы решений.

Таблица 13 – Набор тестов

№	Входные данные	Ожидаемый результат	Результат
1	Псевдоним:ItsTenSymbItsTenSymbItsTenSymb! Почта:artur Роль:	(С1, С2, С4)	Тест пройден (рисунок В.1)
2	Псевдоним: adminexample Почта:a!rthurs Роль: Admin	(С1, С2)	Тест пройден
3	Псевдоним:ItsTenSymbItsTenSymbItsTenSymb! Почта:admin@mail.rus Роль:	(С1, С4)	Тест пройден
4	Псевдоним:ItsTenSymbItsTenSymbItsTenSymb! Почта:arthur@mail.ru Роль:USER	(С1)	Тест пройден (рисунок В.2, В.3)
5	Псевдоним:admin Почта:admin Роль:	(С2, С3, С4)	Тест пройден
6	Псевдоним:admin Почта:adm Роль:USER	(С2, С3)	Тест пройден
7	Псевдоним:adminNew Почта:horror Роль:	(С2, С4)	Тест пройден (рисунок В.4, В.5)

Продолжение таблицы 13

№	Входные данные	Ожидаемый результат	Результат
8	Псевдоним:adminNews Почта:adminNewmail.com Роль:ADMIN	(C2)	Тест пройден
9	Псевдоним:admin Почта:admin@mail.ru Роль:	(C3, C4)	Тест пройден
10	Псевдоним:admin Почта:admin@mail.ru Роль:ADMIN	(C3)	Тест пройден
11	Псевдоним:adminExample Почта:ad@mail.ru Роль:	(C4)	Тест пройден
12	Псевдоним:user1 Почта:user1@mail.ru Роль:USER	(C5)	Тест пройден (рисунок В.6, В7)

Функциональное тестирование показало, что при создании нового пользователя программа работает правильно. Результаты тестирования представлены в приложении В.

4.2 Модульное тестирование

В результате разработки программы было проведено модульное тестирование. Модульные тесты были разработаны с помощью библиотеки JUnit. Исходный код модульных тестов приведен в приложении. План тестирования приведен в таблице 14.

Таблица 14 – План модульного тестирования

Название теста	Описание теста	Ожидаемый результат	Результат
stopProfitRate	Проверка правильности подсчета соотношения тэйк-профита к стоп-лоссу	Пройден	Пройден (рисунок 26)
containsMy	Проверка правильности поиска вхождения элемента в массив	Пройден	Пройден (рисунок 27)
Analyze	Проверка расчета итогового результата анализа прибыльности покупки ордера	Пройден	Пройден (рисунок 28)

На рисунке 26 представлен результат модульного теста stopProfitRate.

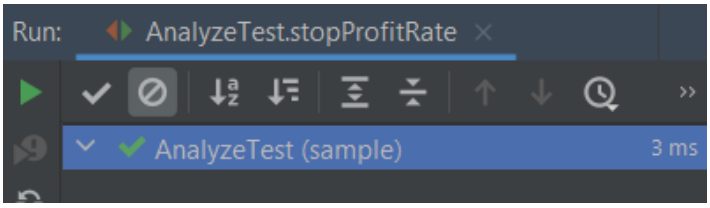

Рисунок 26 – Результат модульного тестирования

На рисунке 27 представлен результат модульного теста containsMy.

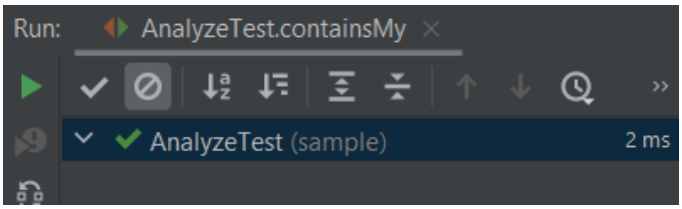

Рисунок 27 – Результат модульного тестирования

На рисунке 28 представлен результат модульного теста containsMy.

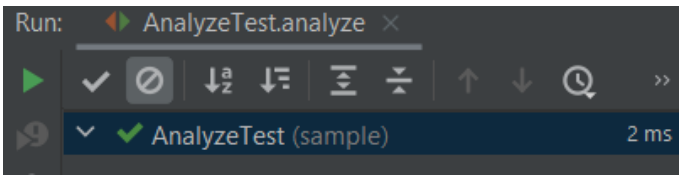

Рисунок 28 – Результат модульного тестирования

Исходный код проекта модульных тестов представлен на листинге 1.

Листинг 1 – Исходный код модульных тестов.

```
package sample;
import org.junit.Assert;
import org.junit.Test;

public class AnalyzeTest {
    @org.junit.Test
    public void stopProfitRate(){
        String a = Analyze.stopProfitRate(100,120,80);
        Assert.assertEquals("1.0",a);
    }
    @org.junit.Test
    public void containsMy(){
        String[] a = new String[]{"Bitcoin", "Ethereum", "Solana", "Binance",
                "Tether", "Cardano", "XRP", "Polkadot", "USD", "Dogecoin"};
        String result = ParserSite.containsMy(a,"USD");
        Assert.assertEquals(result,"USD");


    }
```

```
    @Test
    public void analyze(){
        Analyze an = new Analyze();
        int result = an.analyze("Скальпинг",20,0.7,2);
        Assert.assertEquals(result,99);
    }
}
```

Модульное тестирование основных компонентов классов Analyze и ParserSite реализовано и успешно пройдено без каких-либо ошибок.

4.3 Метрики кода

В качестве первой вычисленной метрики будут вычислены количественные метрики. Данный тип метрик был вычислен с использование утилиты MetricsReload, которая подсчитывает количество пустых строк, количество комментариев, число строк, содержащих исходный код для функций (классов, файлов) [11]. Результаты представлены на рисунке 29.

file type	LOC	NCLOC
.gitignore (GitIgn	2	2
Java	1 629	1 543
Manifest	2	2
XML	367	367
Total	**2 000**	**1 914**
Average	500,00	478,50

Рисунок 29 – Подсчитанная метрика количества строк кода

Также, в данном проекте были рассчитаны соотношения между абстрактностью и устойчивостью.

Для того, чтобы вычислить данные соотношения, сгруппируем основные программные классы в компоненты (пакеты). Получится три компонента: Sample, Forma, Database. Диаграмма компонентов (пакетов) приведена на рисунке 30.

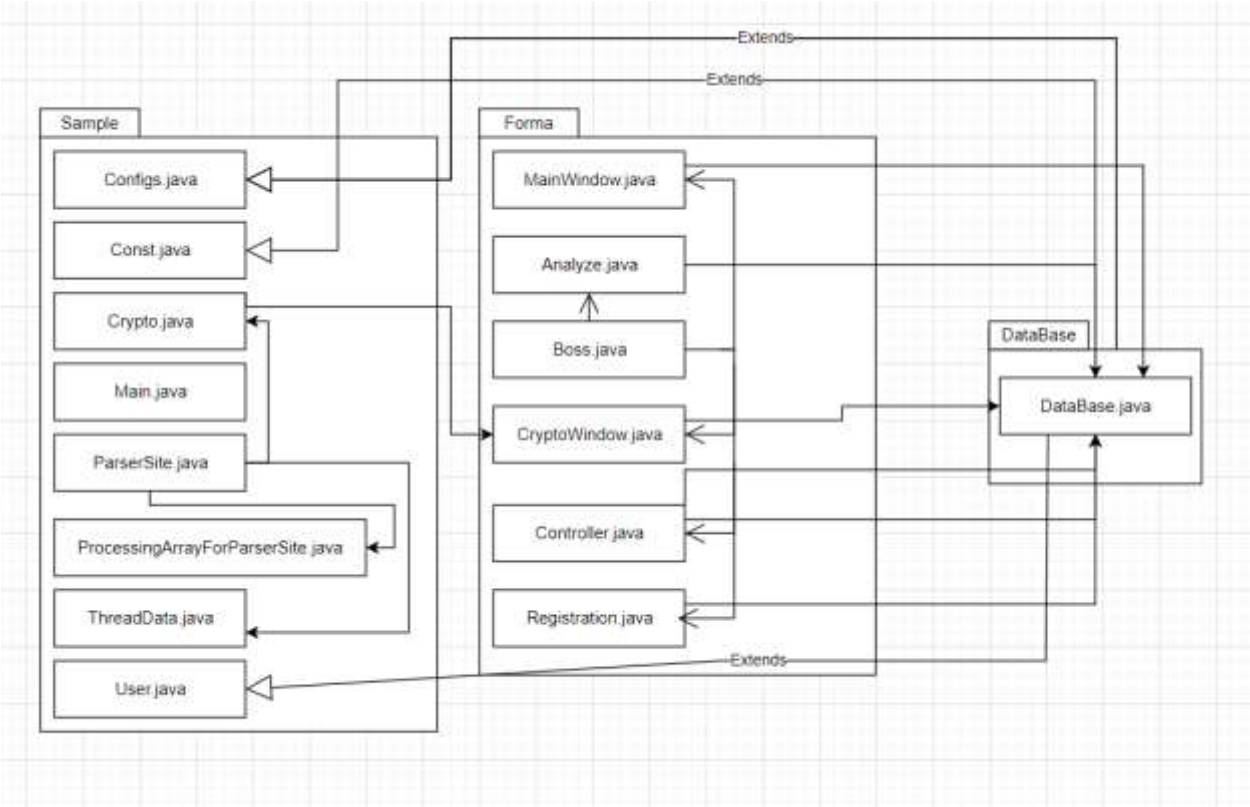

Рисунок 30 – Диаграмма компонентов (пакетов)

Абстрактность вычисляется по следующей формуле:

$$A = \frac{Na}{Nc},$$

где Nc – количество классов в компоненте;

Na – количество абстрактных классов в компоненте.

Была рассчитана абстрактность для следующих пакетов:

– для пакета Forma значение A будет равно 0;

– для пакета Sample значение A будет равно 0;

– для пакета Database значение A будет равно 0.

Из полученных результатов следует, что в компонентах отсутствуют абстрактные классы.

Устойчивость рассчитывается по следующей формуле:

$$I = \frac{Ce}{(Ca + Ce)},$$

где Ca – входящие связи;

Ce – исходящие связи.

Была рассчитана устойчивость для следующих пакетов:

– для пакета Forma значение I будет приблизительно равно 0.83;

– для пакета Sample значение I будет равно 0.25;

– для пакета Database значение I будет равно 0.375.

По полученным результатам можно сказать, что наиболее устойчивым будет компонент Sample. Компонент Forma будет самым неустойчивым.

На основе вычисленных метрик абстрактности и устойчивости для каждого компонента приложения было найдено нормированное расстояние от главной последовательности:

$$D' = |A + I - 1|,$$

где A – абстрактность;

I – устойчивость.

Было рассчитано расстояние от главной последовательности для следующих пакетов:

– для пакета Database расстояние будет равно 0.625;

– для пакета Sample расстояние будет равно 0.75;

– для пакета Forma расстояние будет равно 0.17.

Компоненты Database и Sample находятся близко к главной последовательности, а наиболее удаленным является компонент Forma.

Остальные компоненты лежат достаточно близко к главной последовательности. Это значит, что отношение абстрактности к устойчивости этих компонент является правильным.

4.4 Планирование разработки и оценка бюджета

4.4.1 Структурное планирование

Основной целью структурного планирования является описание состава и взаимосвязи технологических операций, которые требуются для реализации проекта. Также на этом этапе необходимо определить или предварительно оценить продолжительности работ.

Структурное планирование можно разделить на следующие этапы:

– разбиение проекта на совокупность отдельных работ, выполнение которых необходимо для реализации проекта;

– оценка временных характеристик работ и ресурсов.

Результатом структурного планирования является сетевой график – ориентированный граф, вершинами которого являются работы проекта, а дугами временные взаимосвязи.

Он должен удовлетворять следующим свойствам:

– каждой работе соответствует только одна вершина;

– ни одна работа не может быть начата, пока не закончатся все непосредственно предшествующие ей работы;

– ни одна работа, которая непосредственно следует за некоторой работой, не может начаться до момента ее окончания;

– начало и конец проекта (вехи) обозначены работами с нулевой продолжительностью.

Для оценки стоимости и трудоемкости проекта необходимо провести правильное планирование. Для этого были определены задачи и ресурсы, которые необходимы для создания приложения.

В таблице 15 представлены задачи, которые представляют этапы создания проекта.

Таблица 15 – Перечень задач проекта

№	Название	Вид задачи	Предшественник	Длительность
1	Начало реализации проекта	Веха	-	-
2	Анализ требований и постановка Задачи	Фаза	-	-
3	Анализ программных Аналогов	Задача	1	2
4	Постановка задачи	Задача	3	1
5	Анализ функциональных Требований	Задача	3	2
6	Выбор средств разработки	Задача	4;5	2
7	Анализ требований завершен	Веха	6	-
8	Проектирование приложений	Фаза	2	-
9	Функциональное моделирование	Задача	7	3
10	Проектирование структур Данных	Задача	9	2

Продолжение таблицы 15

№	Название	Вид задачи	Предшественник	Длительность
11	Проектирование программных Средств	Задача	10	4
12	Проектирование пользовательского интерфейса	Задача	10	2
13	Проектирование завершено	Веха	11;12	-
14	Реализация	Фаза	8	-
15	Реализация структур данных	Задача	13	3
16	Реализация пользовательского Интерфейса	Задача	15	2
17	Реализация программных Средств	Задача	16	15
18	Реализация завершена	Веха	17	-
19	Отладка и тестирование	Фаза	14	-
20	Отладка серверной части	Задача	18	3
21	Рефакторинг	Задача	21	1
22	Анализ качества кода	Задача	22	2
23	Отладка и тестирование Завершена	Веха	23	-
24	Конец проекта	Веха	23	-

Из таблицы, представленной выше видно, что для реализации проекта необходимо выполнить 24 задачи.

Необходимые ресурсы представлены в таблице 16.

Таблица 16 – Ресурсы

№	Название ресурса	Тип ресурса	Затраты		Начисление
			Стандартная ставка	Ставка сверхур.	
1	Программист	Трудовой	150 000 р./мес.	700 р./ч.	Пропорциональное
2	Тестировщик	Трудовой	60 000 р./мес.	370 р./ч.	Пропорциональное
3	Компьютер	Материальный	90 000р.	-	Пропорциональное
4	Интернет	Затраты	900р	-	Пропорциональное

В таблице, представленной выше, определены зарплаты программистов с учетом ставки сверхурочных, материальные затраты, такие как покупка компьютера и месячная плата за интернет.

4.4.2 Календарное планирование

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project. Проект был начат 13 апреля 2022 года.

При планировании разработки было выделено 24 задачи, выделены и назначены ресурсы для каждой задачи и необходимое количество рабочих дней для ее полного завершения.

Каждая большая задача разбита на множество подзадач, которые определяют итоговые сроки выполнения большой задачи, учитывая нерабочие дни (выходные, праздники).

Ресурсы, необходимые для выполнения проекта были взяты из таблицы 16.

Отношения задач были сформированы по данным из таблицы 15.

Список задач и их описание представлено на рисунке 31.

		Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1		Начало реализации пр	0 дней	Ср 13.04.22	Ср 13.04.22	3	Компьютер[0 штука]
2		**Анализ требований**	**6 дней**	**Ср 13.04.22**	**Ср 20.04.22**	**8**	
3		Сравнительный анал	2 дней	Ср 13.04.22	Чт 14.04.22	4;5	Интернет;Программ
4		Постановка задачи	1 день	Пт 15.04.22	Пт 15.04.22	6	Программист[50%]
5		Анализ функциональ	2 дней	Пт 15.04.22	Пн 18.04.22	6	Программист[50%]
6		Анализ средств разр	2 дней	Вт 19.04.22	Ср 20.04.22	7	Программист
7		Анализ требований	0 дней	Ср 20.04.22	Ср 20.04.22	9	
8		**Проектирование приложения**	**9 дней**	**Чт 21.04.22**	**Вт 03.05.22**	**14**	
9		Функциональное мо	3 дней	Чт 21.04.22	Пн 25.04.22	10	Программист
10		Проектирование стр	2 дней	Вт 26.04.22	Ср 27.04.22	11;12	Программист
11		Проектирование про	4 дней	Чт 28.04.22	Вт 03.05.22	13	Программист[50%]
12		Проектирование пол	2 дней	Чт 28.04.22	Пт 29.04.22	13	Программист[50%]
13		Проектирование зав	0 дней	Вт 03.05.22	Вт 03.05.22	15	
14		**Реализация**	**20 дней**	**Ср 04.05.22**	**Ср 01.06.22**	**19**	
15		Реализация структур	3 дней	Ср 04.05.22	Пт 06.05.22	16	Программист
16		Реализация пользов	2 дней	Пн 09.05.22	Вт 10.05.22	17	Программист
17		Реализация REST API	15 дней	Ср 11.05.22	Вт 31.05.22	18	Интернет;Программ
18		Реализация заверше	0 дней	Ср 01.06.22	Ср 01.06.22	20	
19		**Отладка и тестирован**	**6 дней**	**Ср 01.06.22**	**Чт 09.06.22**		
20		Отладка серверной ч	3 дней	Ср 01.06.22	Пт 03.06.22	21	Тестировщик[70%]
21		Рефакторинг	2 дней	Пн 06.06.22	Вт 07.06.22	22	Программист;Тестир
22		Анализ качества код	1 день	Ср 08.06.22	Ср 08.06.22	23	Тестировщик
23		Отладка и тестирова	0 дней	Чт 09.06.22	Чт 09.06.22	24	
24		Конец проекта	0 дней	Чт 09.06.22	Чт 09.06.22		
25							

Рисунок 31 – Список задач проекта

В результате выделения задач и ресурсов была построена диаграмма Ганта (рисунок 32). На ней представлены продолжительность задач и взаимосвязи их друг с другом.

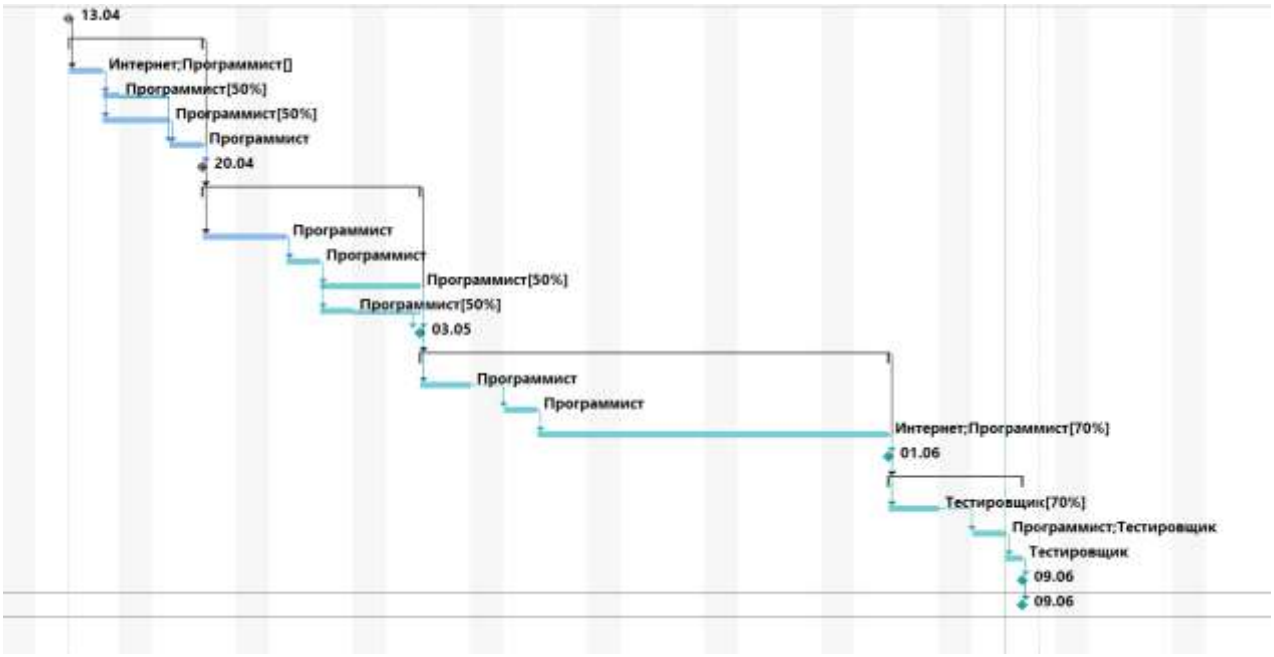

Рисунок 32 – Диаграмма Ганта

По диаграмме Ганта можно определить даты начала и окончания проекта, часовую переработку работников и их занятость над каждой задачей.

На рисунках 33, 34 представлены обзор проекта и обзор ресурсов.

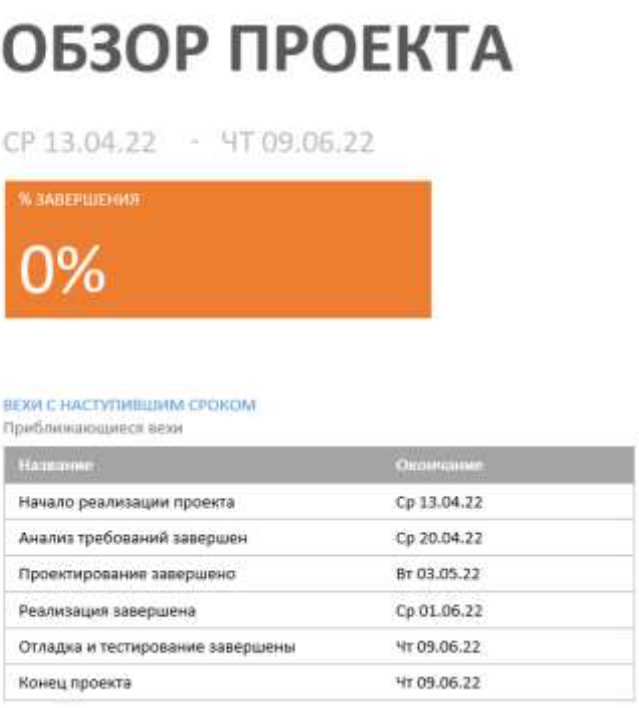

Рисунок 33 – Обзорный отчет проекта

Обзорный отчет проекта позволяет определить процент завершения проекта, если он начат и длительность всего проекта, составляющая 57 дней.

Рисунок 34 – Обзор затрат

Из рисунка видно, что предполагаемые затраты на проект составляют 226 240 руб.

Заключение

В ходе выполнения выпускной квалификационной работы разработано приложение по работе с криптовалютой на демо-счете.

Проведен сравнительный анализ приложений по работе с криптовалютой, выделены основные преимущества и недостатки каждого из них. На данной основе выделены функциональные и нефункциональные требования к системе.

Выбрана архитектура системы и основной шаблон проектирования. Спроектирована логическая и физическая схема БД, отображающая основные сущности и связи в выбранной предметной области. Для основных модулей построены диаграммы классов.

Разработка программных средств проводилась в среде разработки Intellij IDEA на языках программирования Java и JavaScript. Разработанная программа функционирует на любом современном web-браузере.

Разработанное программное обеспечение протестировано. Выполнено функциональное и модульное тестирование. Все тесты пройдены успешно.

Также проведено планирование и оценка бюджета проекта при помощи программы для управления проектами Microsoft Project 2013. Затраты на проект составили 226 240 рублей. Время разработки 41 рабочих дня.

Таким образом, в результате выполнения выпускной квалификационной работы разработано приложение по работе с криптовалютой на демо-счете, отвечающее всем требованиям технического задания.

Список использованных источников

1. Виды криптовалютных кошельков [Электронный ресурс]. – URL: https://stoprazvod.ru/baza-znaniy/vidy-kriptovalyutnykh-koshelkov/ (дата обращения: 09.05.2022).

2. Демо-счет у брокера – гайд и анализ [Электронный ресурс]. – URL: https://invest-mate.com/trading/uroven-nachinaychiy/demo-schet-guide#:~:text=Демонстрационный%20счет%20(ДС)%20–%20это,индикатор%2C%20новый%20подход%20в%20торговле/ (дата обращения: 10.05.2022).

3. Приложения биржи Binance [Электронный ресурс]. URL: https://profinvestment.com/binance-exchange-apps/ (дата обращения: 12.05.2022).

4. Currency – криптовалютная биржа [Электронный ресурс]. URL: https://stoprazvod.ru/currency-com-kriptovalyutnaya-birzha-belarusi/ (дата обращения: 12.05.2022).

5. Биржа ByBit для торговли криптовалютными деривативами [Электронный ресурс]. URL: https://investormaster.com/kriptovalyuta/birzha-bybit-dlya-torgovli-kriptovalyutnymi-derivativami.html/ (дата обращения: 22.05.2022).

6. UML-диаграммы классов [Электронный ресурс]. URL: https://prog-cpp.ru/uml-classes/ (дата обращения: 24.05.2022).

7. Основы Hibernate [Электронный ресурс]. URL: https://habr.com/ru/post/29694/ (дата обращения: 24.05.2022).

8. Учебное пособие по диаграммам последовательностей [Электронный ресурс]. URL: https://creately.com/blog/ru/диаграмма/учебное-пособие-по-последовательной/ (дата обращения: 24.05.2022).

9. Диаграмма развертывания [Электронный ресурс]. URL: https://intuit.ru/studies/courses/32/32/lecture/1024/ (дата обращения: 24.05.2022).

10. Классы эквивалентности и анализ граничных значений [Электронный ресурс]. URL: https://asjudc.gitbooks.io/almeln/content/2016-09-

04-equivalence-classes-and-boundary-value-analysis.html/ (дата обращения: 27.05.2022).

11. SWEN-261 Code Metrics [Электронный ресурс]. URL: https://www.se.rit.edu/~swen-261/activities/Code%20Metrics/index.html (дата обращения: 01.06.2022).

Приложение А

ГЛОССАРИЙ

(справочное)

База данных (БД) – это информативная модель, позволяющая организованно поддерживать данные о группе объектов, имеющих один и тот же набор параметров.

Unified Modeling Language (UML) – графический унифицированный язык моделирования, использующий объектно-ориентированный подход для моделирования в сфере разработки программного обеспечения.

Stop-loss – биржевая заявка, выставленная в торговом терминале трейдером или инвестором с целью ограничить свои убытки при достижении ценой заранее определенного уровня.

ORM – технология, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

CASE-средства – набор инструментов и методов программной инженерии для проектирования программного обеспечения, который помогает обеспечить высокое качество программ, отсутствие ошибок и простоту в обслуживании программных продуктов.

Take-profit – ограничивает величину прибыли. Инвестор определяет цену, при достижении которой брокер автоматически продаст акции после их роста.

Приложение Б

ЛИСТИНГ ПРИЛОЖЕНИЯ

(рекомендуемое)

Листинг Б.1 – Файл SpringBootSecurityDemoApplication.java

```
package ru.kata.spring.boot_security.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import ru.kata.spring.boot_security.demo.parser.ThreadData;

@SpringBootApplication
public class SpringBootSecurityDemoApplication {

   public static void main(String[] args) {
      SpringApplication.run(SpringBootSecurityDemoApplication.class, args);
      ThreadData threadData = new ThreadData();
      threadData.run();
   }

}
```

Листинг Б.2 – Файл UserService.java

```
package ru.kata.spring.boot_security.demo.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ru.kata.spring.boot_security.demo.model.Role;
import ru.kata.spring.boot_security.demo.model.User;
import ru.kata.spring.boot_security.demo.repository.UserRepository;

import java.util.ArrayList;
import java.util.List;

@Service
public class UserService {


    private final UserRepository userRepository;

    @Autowired
    public UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    public List<User> index() {
        return userRepository.findAll();
    }

    public User getUserById(long id) {
        return userRepository.findById(id).get();
    }

    public void addUser(User user) {
        List<Role> roles = (List<Role>) user.getRoles();
        if(roles.size()==2){
            roles = roleByName(new String[]{"USER","ADMIN"});
            user.setRoles(roles);
        }
        else if (roles.get(0).getId()==2){
            roles = roleByName(new String[]{"ADMIN"});
            user.setRoles(roles);
        }
        else if (roles.get(0).getId()==1){
            roles = roleByName(new String[]{"USER"});
            user.setRoles(roles);
        }
        userRepository.save(user);
    }
```

```
    public List<Role> roleByName(String[] roles){
        List<Role> l = new ArrayList<>();
        for (int i = 0; i < roles.length; i++){
            if (roles[i].equals("USER")){
                l.add(new Role(2L,"ROLE_USER"));
            }
            if (roles[i].equals("ADMIN")){
                l.add(new Role(1L,"ROLE_ADMIN"));
            }
        }
        return l;
    }

    public void removeUser(long id) {
        userRepository.deleteById(id);
    }

    public void updateUser(User user) {
        List<Role> roles = (List<Role>) user.getRoles();
        if(roles.size()==2){
            roles = roleByName(new String[]{"USER","ADMIN"});
            user.setRoles(roles);
        }
        else if (roles.get(0).getId()==1){
            roles = roleByName(new String[]{"ADMIN"});
            user.setRoles(roles);
        }
        else if (roles.get(0).getId()==2){
            roles = roleByName(new String[]{"USER"});
            user.setRoles(roles);
        }
        userRepository.save(user);
    }

    public User getUserByUsername(String username) {
        return userRepository.getUserByUsername(username);
    }
}
```

Листинг Б.3 – Файл RoleService.java

```
package ru.kata.spring.boot_security.demo.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import ru.kata.spring.boot_security.demo.model.Role;
import ru.kata.spring.boot_security.demo.repository.RoleRepository;
import java.util.List;

@Service
public class RoleService {
    private final RoleRepository roleRepository;

    @Autowired
    public RoleService(RoleRepository roleRepository) {
        this.roleRepository = roleRepository;
    }

    public List<Role> findAllRoles() {
        return roleRepository.findAll();
    }

    public Role getRoleByName(String name) {
        return roleRepository.getRoleByName(name);
    }
}
```

Листинг Б.4 – Файл UserDetailService.java

```
package ru.kata.spring.boot_security.demo.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import ru.kata.spring.boot_security.demo.model.User;

@Service
public class UserDetailServiceImpl implements UserDetailsService {

    private final UserService userService;

    @Autowired
    public UserDetailServiceImpl(UserService userService) {
        this.userService = userService;
    }

    @Transactional
    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userService.getUserByUsername(username);
        if (user == null) {
            throw new UsernameNotFoundException(String.format("User '%s' not
found!", username));
        }
        return new org.springframework.security.core.userdetails.User(
                user.getUsername(), user.getPassword(),
user.getAuthorities());
    }
}
```

Листинг Б.5 – Файл ThreadData.java

```
package ru.kata.spring.boot_security.demo.parser;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;



public class ThreadData{
    ArrayList<Crypto> cryptoData = new ArrayList<>();
    DataBaseHandler dataBaseHandler = new DataBaseHandler();
    public String[] arrayPolozenye = new String[]{"Биткоин", "Эфириум",
"Solana", "Binance", "Tether", "Кардано", "Ripple", "Polkadot", "USD",
"Dogecoin"};

    public void run() {
        while(true){
            try {
                Thread.sleep(1000*60);
                cryptoData.clear();
                initData();
                if(cryptoData.size()>1){
                    System.out.println(cryptoData.get(1).name);
                    System.out.println(cryptoData.get(1).sale);
                }
                System.out.println("done");
```

```
            } catch (SQLException | ClassNotFoundException |
InterruptedException throwables) {
                throwables.printStackTrace();
            }
        }
    }

    public void initData() throws SQLException, ClassNotFoundException {
        ParserSite parserSite = new ParserSite();
        ArrayList<Crypto> arrayList = parserSite.Go();
        arrayList = swapPlaces(arrayList);

        for (int i = 0; i < 10; i++) {
            if (arrayList.get(0) != null) {
                cryptoData.add(arrayList.get(i));
            }
            else {
                ResultSet rs = dataBaseHandler.getOfflineData(i);
                rs.next();
                cryptoData.add(new Crypto(arrayPolozenye[i],rs.getFloat(3)));
            }
            ResultSet resultSet = dataBaseHandler.getCountToNowDate(i);

dataBaseHandler.saveInDatabaseCryptoCurrency(i,cryptoData.get(i).getSale());
            ArrayList<Integer> minutka = new ArrayList<>();
            while (resultSet.next()){
                minutka.add(resultSet.getInt(1));
            }

            if(minutka.size()==24){

dataBaseHandler.updateHistory(minutka.get(23),cryptoData.get(i).getSale());
            }
            else{


dataBaseHandler.saveInDatabaseHistoryCryptoCurrency(i,cryptoData.get(i).getSa
le());
            }
        }
    }
    private ArrayList<Crypto> swapPlaces(ArrayList<Crypto> resultArr){
        ArrayList<Crypto> arr = new ArrayList<>();
        ArrayList<String> arrexample = new ArrayList<>();
        for (int i = 0; i <10 ; i++) {
            arr.add(null);
        }
        for (int i = 0; i < resultArr.size(); i++) {
            String [] a = resultArr.get(i).getName().split(" ");
            arrexample.add(a[0]);

        }
        for (int i = 0; i < resultArr.size(); i++) {
            for (int j = 0; j < arrayPolozenye.length; j++) {
                if(arrexample.get(i).equals(arrayPolozenye[j]) && i==j){
                    arr.set(i,resultArr.get(i));
                }
                else if(arrexample.get(i).equals(arrayPolozenye[j])){
                    arr.set(j,resultArr.get(i));
                }
            }

        }
        return arr;
    }
}
```

Листинг Б.6 – Файл ParserSite.java

```
package ru.kata.spring.boot_security.demo.parser;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import javax.net.ssl.HttpsURLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.Arrays;


public class ParserSite {
    public ArrayList<Crypto> cryptos = new ArrayList<>();

    public ArrayList<Crypto> Go() {
        String query = "https://www.coingecko.com/ru";
        HttpsURLConnection connection = null;
        try {
            connection = (HttpsURLConnection) new
URL(query).openConnection();
            connection.setRequestMethod("GET");
            connection.setUseCaches(false);
            connection.setConnectTimeout(250);
            connection.setReadTimeout(250);
            connection.connect();
            StringBuilder sb = new StringBuilder();
            if (HttpURLConnection.HTTP_OK == connection.getResponseCode()) {
                BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                String line;
                while ((line = in.readLine()) != null) {
                    sb.append(line);
                    sb.append('\n');
                }
                String html = String.valueOf(sb);
                Document document = Jsoup.parse(html);
                parseDocument(document);
                connection.disconnect();
                return cryptos;

            } else {
                System.out.println(connection.getResponseCode());
                connection.disconnect();
                return null;
            }
        } catch (Throwable t) {
            System.out.println(t);
            return cryptos;

        } finally {
            if (connection != null) {
                connection.disconnect();
            }
        }
    }

    private void parseDocument(Document document) {
```

```
        int i = 0;
        ProcessingArrayForParserSite p = new ProcessingArrayForParserSite();
        Elements tables = document.select("table");
        for (Element table : tables) {
            Elements rows = table.select("tr");
            for (Element row : rows) {
                Elements cols = row.select("td");
                String[] arr = new String[8];
                int index = 0;
                for (Element col : cols) {
                    String s = col.text();
                    if (!col.text().equals("")) {
                        arr[index] = col.text();
                        index = index + 1;
                    }
                }
                boolean notMoreTen = true;
                if (arr[0] != null) {
                    if (p.countNotNull(arr) <= 2) {
                        notMoreTen = false;
                    }
                    if (i < 10) {

                        if (notMoreTen && !p.containsMy(p.a, arr[1].split("
")[0]).equals("")) {
                            String name = p.containsMy(p.a, arr[1].split("
")[0]);
                            String sales = "";
                            sales = arr[2].substring(1);
                            String[] begin = sales.split(",")[0].split(" ");
                            String resBegin = "";
                            for (String s : begin) {
                                resBegin = resBegin + s;
                            }
                            sales = resBegin + "." + sales.split(",")[1];
                            Crypto crypto = new Crypto(name,
Float.parseFloat(sales));
                            cryptos.add(crypto);
                            i = i + 1;
                        } else if (!notMoreTen && !p.containsMy(p.a,
arr[0].split(" ")[0]).equals("")) {

                            String sales = "";
                            sales = arr[1].substring(1);
                            String begin = sales.substring(0, sales.length()
- 3);
                            begin = begin.replaceAll("\\pP", "");
                            String end = sales.substring(sales.length() - 3,
sales.length());
                            String name = p.containsMy(p.a, arr[0].split("
")[0]);

                            sales = begin + end;
                            Crypto crypto = new Crypto(name,
Float.parseFloat(sales));
                            cryptos.add(crypto);
                            i = i + 1;
                        }
                    }
                }
            }
        }
    }
}
```

Листинг Б.6 – Файл DataBaseHandler.java

```
package ru.kata.spring.boot_security.demo.parser;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
public class DataBaseHandler extends Configs {
    Connection dbConnection;

    public Connection getDbConnection() throws ClassNotFoundException,
SQLException {
        String connectionString = "jdbc:mysql://" + dbHost + ":" + dbPort +
"/" + dbName;
        Class.forName("com.mysql.cj.jdbc.Driver");
        dbConnection = DriverManager.getConnection(connectionString, dbUser,
dbPass);

        return dbConnection;

    }

    public ResultSet getDate(String d1,String d2, String g) throws
SQLException, ClassNotFoundException {
        ResultSet resultSet = null;
        String select = new StringBuilder().append("SELECT currency,date FROM
").append(Const.CRYPTO_HISTORY).append(" WHERE idcrypto = ").append("'")
                .append(g).append("'").append(" and date BETWEEN
").append("'")
                .append(d1).append("'").append(" AND
").append("'").append(d2).append("'").toString();
        PreparedStatement preparedStatement =
getDbConnection().prepareStatement(select);
        resultSet = preparedStatement.executeQuery();
        return resultSet;
    }

    public void saveInDatabaseCryptoCurrency(int id,float currency) throws
SQLException, ClassNotFoundException {
        String update = new StringBuilder().append("UPDATE
").append(Const.CRYPTO_TABLE).append(" SET ").append
                ("currency = ").append("'").append(currency).append("' , ")
                .append("date = NOW()")
                .append
                        (" WHERE idcrypto =
").append("'").append(id+1).append("'").toString();
        PreparedStatement preparedStatement =
getDbConnection().prepareStatement(update);
        preparedStatement.executeUpdate();

    }

    public void saveInDatabaseHistoryCryptoCurrency(int i, float sale) throws
SQLException, ClassNotFoundException {
        String insert = "INSERT INTO " + Const.CRYPTO_HISTORY + "("+"
idcrypto, date, currency "+ ")" +
                "VALUES(?,NOW(),?)";
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, String.valueOf(i+1));
        prSt.setString(2, String.valueOf(sale));
```

```
        prSt.executeUpdate();
    }

    public void saveInDatabaseHistoryCotirovkaCryptoCurrency(int i,String
date, float sale) throws SQLException, ClassNotFoundException {
        String insert = "INSERT INTO " + Const.CRYPTO_HISTORY + "("+"
idcrypto, date, currency "+ ")" +
                "VALUES(?,?,?)";
        PreparedStatement prSt = getDbConnection().prepareStatement(insert);
        prSt.setString(1, String.valueOf(i+1));
        prSt.setString(2, date);
        prSt.setString(3, String.valueOf(sale));
        prSt.executeUpdate();
    }


    public ResultSet getCountToNowDate(int i) throws SQLException,
ClassNotFoundException {
        ResultSet resultSet = null;
        String select = "Select idhistory,date from " + Const.CRYPTO_HISTORY
+ " where idcrypto = " + (i+1) +" and date >= curdate()";
        PreparedStatement preparedStatement =
getDbConnection().prepareStatement(select);
        resultSet = preparedStatement.executeQuery();
        return resultSet;
    }

    public void updateHistory(int resultSetInt, float anInt) throws
SQLException, ClassNotFoundException {
        String update = "UPDATE " + Const.CRYPTO_HISTORY + " set currency = "
+ (anInt) +
                " where idhistory = " + resultSetInt;
        PreparedStatement preparedStatement =
getDbConnection().prepareStatement(update);
        preparedStatement.executeUpdate();
    }

    public ResultSet getOfflineData(int i) throws SQLException,
ClassNotFoundException {
        ResultSet resultSet = null;
        String SELECT = "SELECT * FROM serenity.crypto_history where idcrypto
= " + (i+1) + " ORDER BY idhistory DESC LIMIT 1";
        PreparedStatement preparedStatement =
getDbConnection().prepareStatement(SELECT);
        resultSet = preparedStatement.executeQuery();
        return resultSet;
    }

    public ResultSet getLastPrice(String s) throws SQLException,
ClassNotFoundException {
        ResultSet resultSet = null;
        String SELECT = "SELECT * FROM serenity.crypto_history where idcrypto
= " + s + " ORDER BY idhistory DESC LIMIT 1";
        PreparedStatement preparedStatement =
getDbConnection().prepareStatement(SELECT);
        resultSet = preparedStatement.executeQuery();
        return resultSet;
    }
}
```

Листинг Б.7 – Файл Crypto.java

```
package ru.kata.spring.boot_security.demo.parser;

public class Crypto {
```

```
    public String name;
    public float sale;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public float getSale() {
        return sale;
    }

    public void setSale(int sale) {
        this.sale = sale;
    }



    public  Crypto(String name, float v){
        this.name = name;
        this.sale = v;
    }

}
```

Листинг Б.8 – Файл User.java

```
package ru.kata.spring.boot_security.demo.model;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import java.util.Collection;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;
import java.util.stream.Collectors;

@Entity
@Table(name = "users")
public class User implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long idusers;

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
    @Column(name = "gender")
    private  String gender;

    public String getGender() {
        return gender;
    }
```

```
    public void setGender(String gender) {
        this.gender = gender;
    }

    @Column(name = "location")
    private  String location;
    @Column(name = "firstname")
    private  String firstname;
    @Column(name = "surname")
    private String lastname;
    @Column(name = "email")
    private String email;
    @Column(name = "username")
    private String username;
    @Column(name = "password")
    private String password;


    @ManyToMany
    @JoinTable(name = "users_roles",
            joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id"))
    private Collection<Role> roles  = new HashSet<>();

    public User() {
    }

    public String getLocation() {
        return location;
    }

    public void setLocation(String location) {
        this.location = location;
    }

    public User(String firstname, String lastname, String gender, String
email, String username, String location, String password, Set<Role> roles) {
        this.firstname = firstname;
        this.lastname = lastname;
        this.gender = gender;
        this.email = email;
        this.username = username;
        this.location = location;
        this.password = password;
        this.roles = roles;
    }

    public Long getId() {
        return idusers;
    }

    public void setId(Long id) {
        this.idusers = id;
    }



    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getEmail() {
```

```
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Collection<Role> getRoles() {
        return roles;
    }

    public void setRoles(Collection<Role> roles) {
        this.roles = roles;
    }

    public void addRole(Role role) { roles.add(role); }

    public boolean containsRoleName(String roleName) {
        return
roles.stream().map(Role::getName).collect(Collectors.toList()).contains(roleN
ame);
    }

    public Set<String> getNamesOfRoles() {
        return getRoles().stream().map(Role::getName)
                .map(name -> name.startsWith("ROLE_") ? name.substring(5) :
name).collect(Collectors.toSet());
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return getRoles().stream().map(r -> new
SimpleGrantedAuthority(r.getName())).collect(Collectors.toList());
    }

    @Override
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
```

```
    }

    @Override
    public boolean isEnabled() {
        return true;
    }



    @Override
    public int hashCode() {
        return Objects.hash(idusers, lastname, email, username, password,
roles);
    }

    @Override
    public String toString() {
        return "User{" +
                "id=" + idusers +
                '\'' +
                ", lastname='" + lastname + '\'' +
                ", email='" + email + '\'' +
                ", username='" + username + '\'' +
                '}';
    }
}
```

Листинг Б.9 – Файл RestUserController.java

```
package ru.kata.spring.boot_security.demo.controller;

import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.web.bind.annotation.*;
import ru.kata.spring.boot_security.demo.model.User;
import ru.kata.spring.boot_security.demo.service.UserService;

import java.util.List;

@RestController
@RequestMapping("/rest/api")
public class RestUsersController {

    private final UserService userService;
    private final PasswordEncoder passwordEncoder;

    public RestUsersController(UserService userService, PasswordEncoder
passwordEncoder) {
        this.userService = userService;
        this.passwordEncoder = passwordEncoder;
    }

    @GetMapping("/users")
    public List<User> getAllUsers() {
        return userService.index();
    }

    @GetMapping("/users/{id}")
    public User getUserById(@PathVariable long id) {
        return userService.getUserById(id);
    }

    @PostMapping("/users")
    public void addNewUser(@RequestBody User user) {
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userService.addUser(user);
    }
```

```
    @PutMapping("/users")
    public void updateUser(@RequestBody User user) {
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userService.updateUser(user);
    }

    @DeleteMapping("/users/{id}")
    public void deleteUser(@PathVariable long id) {
        userService.removeUser(id);
    }
}
```

Листинг Б.10 – Файл AdminController.java

```
package ru.kata.spring.boot_security.demo.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import ru.kata.spring.boot_security.demo.model.User;
import ru.kata.spring.boot_security.demo.service.RoleService;
import ru.kata.spring.boot_security.demo.service.UserService;

import java.io.IOException;
import java.security.Principal;

@Controller
@RequestMapping("/admin")
public class AdminController {


    private final UserService userService;
    private final RoleService roleService;
    private final PasswordEncoder passwordEncoder;

    @Autowired
    public AdminController(UserService userService, RoleService roleService,
PasswordEncoder passwordEncoder) {
        this.userService = userService;
        this.roleService = roleService;
        this.passwordEncoder = passwordEncoder;
    }

    @GetMapping
    public String allUsers(Model model, Principal principal) {
        model.addAttribute("users", userService.index());
        model.addAttribute("userLogin",
userService.getUserByUsername(principal.getName()));
        model.addAttribute("allRoles", roleService.findAllRoles());
        model.addAttribute("newUser", new User());
        return "admin";
    }

    @GetMapping("/{id}/delete")
    public String delete(@PathVariable("id") long id) {
        userService.removeUser(id);
        return "redirect:/admin";
    }
    @GetMapping("/prilozAdmin")
    public String getApp(Principal principal,Model model) throws IOException
{
        User u = userService.getUserByUsername(principal.getName());
        String id = String.valueOf(u.getId());
```

```
        String username = String.valueOf(u.getUsername());
        ProcessBuilder pb = new ProcessBuilder("java", "-jar",
"E:/diplom/Serenity/out/artifacts/SerenityCrypto_jar/SerenityCrypto.jar",
id,username);
        Process p = pb.start();
        return "redirect:/admin";

    }

    @PostMapping("/{id}")
    public String update(@ModelAttribute("editUser") User user,
                         @PathVariable("id") long id) {
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userService.updateUser(user);
        return "redirect:/admin";
    }

    @PostMapping("/new")
    public String createUser(@ModelAttribute("newUser") User user) {
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        userService.addUser(user);
        return "redirect:/admin";
    }
}
```

Листинг Б.11 – Файл users.js

```
const allUsersTableBody = document.getElementById('all-users-table-body');
const URLUsersApi = 'http://localhost:8080/rest/api/users';
let users;




async function loadIntoTablesBody(url, tableBody) {
    const response = await fetch(url);
    users = await response.json();

    tableBody.innerHTML = ''

    for (const user of users) {
        const bodyElement = document.createElement("tr")
        bodyElement.id = user.id
        bodyElement.appendChild(createTd(user.id));
        bodyElement.appendChild(createTd(user.firstname));
        bodyElement.appendChild(createTd(user.lastname));
        bodyElement.appendChild(createTd(user.email));
        bodyElement.appendChild(createTd(user.location));
        bodyElement.appendChild(createTd(user.gender));
        bodyElement.appendChild(createTd(user.username));
        bodyElement.appendChild(createTd(`${user.namesOfRoles.join(', ')}`));
        const editCell = document.createElement("td");
        editCell.innerHTML = `
            <button type="button" onClick="editModalOpen(event, ${user.id})"
class="btn btn-primary" data-bs-toggle="modal" data-bs-
target="#editModal">Edit</button>
        `;
        bodyElement.appendChild(editCell);

        const deleteCell = document.createElement("td");
        deleteCell.innerHTML = `
            <button type="button" onClick="deleteModalOpen(event,
${user.id})" class="btn btn-danger" data-bs-toggle="modal" data-bs-
target="#deleteModal">Delete</button>
        `;
        bodyElement.appendChild(deleteCell);
```

```
        tableBody.appendChild(bodyElement)
    }
}

function createTd(user) {
    const newElement = document.createElement("td");
    newElement.textContent = user;
    return newElement;
}

loadIntoTablesBody(URLUsersApi, allUsersTableBody);

function editModalOpen(e, id) {
    e.preventDefault()
    fetch('http://localhost:8080/rest/api/users/' + id, {
        method: 'GET'
    })
        .then(res => res.json())
        .then((data) => {
            let controlId = document.getElementById('idEdit')
            let controlFirstname = document.getElementById('firstnameEdit')
            let controlLocation = document.getElementById('locationEdit')
            let controlGender = document.getElementById('genderEdit')
            let controllastname = document.getElementById('lastnameEdit')
            let controlEmail = document.getElementById('emailEdit')
            let controlUsername = document.getElementById('usernameEdit')
            let controlPassword = document.getElementById('passwordEdit')
            let controlRoles = document.getElementById('rolesEdit')
            let optionSelect = controlRoles.getElementsByTagName('option')

            reSelectRolesOption(optionSelect, data);
            fillValuesFromData(controlId, controllastname, controlEmail,
controlUsername,controlFirstname,controlLocation, controlGender, data)
            controlPassword.value = data.password
        })
}

function fillValuesFromData(controlId, controlName, controlEmail,
controlUsername,controlFirstname,controlLocation, controlGender, data) {
    controlId.value = data.id
    controlName.value = data.lastname
    controlEmail.value = data.email
    controlUsername.value = data.username
    controlFirstname.value = data.firstname
    controlLocation.value = data.location
    controlGender.value = data.gender
}

function deleteModalOpen(e, id) {
    e.preventDefault()
    fetch('http://localhost:8080/rest/api/users/' + id, {
        method: 'GET'
    })
        .then(res => res.json())
        .then((data) => {
            let controlDelId = document.getElementById('idDelete')
            let controlDelFir = document.getElementById('firstnameDelete')
            let controlDelLoc = document.getElementById('locationDelete')
            let controlDelGen = document.getElementById('genderDelete')
            let controlDellastname =
document.getElementById('lastnameDelete')
            let controlDelEmail = document.getElementById('emailDelete')
            let controlDelUsername =
document.getElementById('usernameDelete')
            let controlRoles = document.getElementById('rolesDelete')
            let optionSelect = controlRoles.getElementsByTagName('option')
```

```
            reSelectRolesOption(optionSelect, data)
            fillValuesFromData(controlDelId,  controlDellastname,
controlDelEmail, controlDelUsername, controlDelFir,
controlDelLoc,controlDelGen, data)
        })
}

function sendDeleteForm(e) {
    e.preventDefault()
    let form = document.getElementById('deleteForm')
    let id = form.querySelector('#idDelete').value
    fetch(`http://localhost:8080/rest/api/users/`+id, {
        method: 'DELETE'
    })
    let row = document.getElementById(`${id}`)
    row.remove()
    document.getElementById('deleteFormClose').click()
}

function reSelectRolesOption(optionSelect, data) {
    for (let i = 0; i < optionSelect.length; i++) {
        optionSelect[i].selected = false
    }
    for (let i = 0; i < optionSelect.length; i++) {
        data.roles.forEach(role => {
            if (role.name.includes(optionSelect[i].id)) {
                optionSelect[i].selected = true
            }
        })
    }
}

async function sendSaveForm(e) {
    e.preventDefault()

    let form = document.getElementById('saveForm')
    let formData = new FormData(form)
    let values = Object.fromEntries(formData.entries())

    let option = form.querySelectorAll('option')
    let roleArr = []

    for (let i = 0; i < option.length; i++) {
        if (option[i].selected === true) {
            let eachRole = {}
            eachRole.id = option[i].value
            roleArr.push(eachRole)
        }
    }
    values.roles = roleArr
    console.log(values.roles)

    await fetch(URLUsersApi, {
        method: 'POST',
        headers: {
            // 'Accept': 'application/json',
            'Content-Type': 'application/json',
            // 'Referer': null
        },
        body: JSON.stringify(values)
    }).then(form.reset())
    loadIntoTablesBody(URLUsersApi, allUsersTableBody);
    document.getElementById('nav-users-table-tab').click()
}
```

```
async function sendEditForm(e) {
    e.preventDefault()
    let form = document.getElementById('editForm')
    let formData = new FormData(form)
    let values = Object.fromEntries(formData.entries())
    let option = form.querySelectorAll('option')
    let roleArr = []

    for (let i = 0; i < option.length; i++) {
        if (option[i].selected === true) {
            let eachRole = {}
            eachRole.id = option[i].value
            roleArr.push(eachRole)
        }
    }

    values.roles = roleArr
    await fetch(URLUsersApi, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify(values)
    }).then(form.reset())
    loadIntoTablesBody(URLUsersApi, allUsersTableBody);
    document.getElementById('editClose').click()
}
```

Приложение В

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

(рекомендуемое)

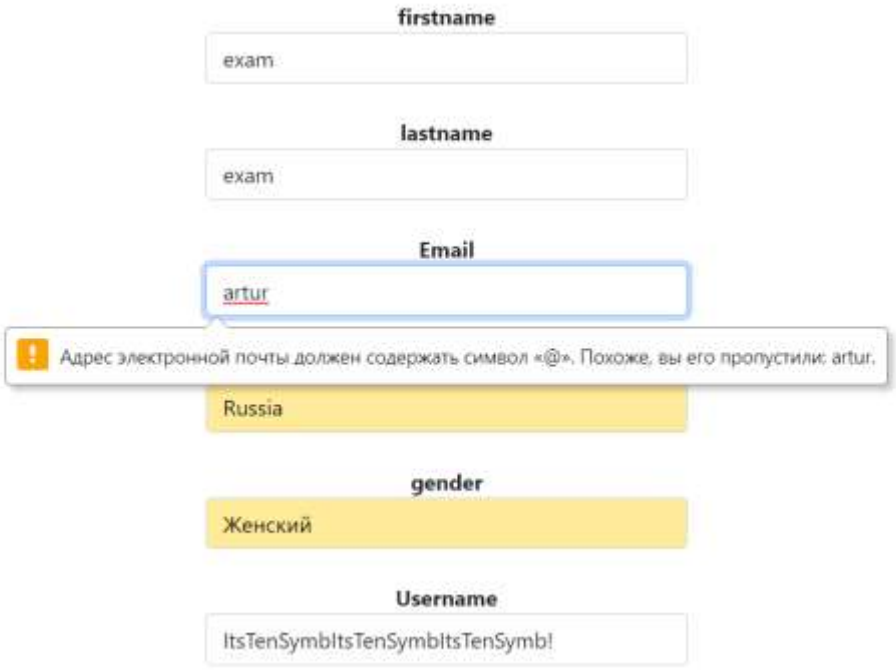

Рисунок В.1 – Тест 1, покрывающий таблицу решений

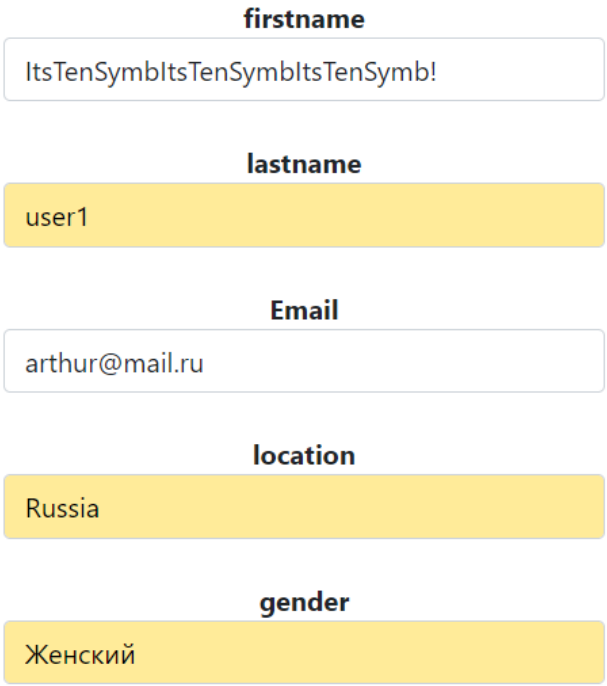

Рисунок В.2 – Тест 4, покрывающий таблицу решений

Рисунок В.3 – Тест 4, покрывающий таблицу решений

Рисунок В.4 – Тест 7, покрывающий таблицу решений

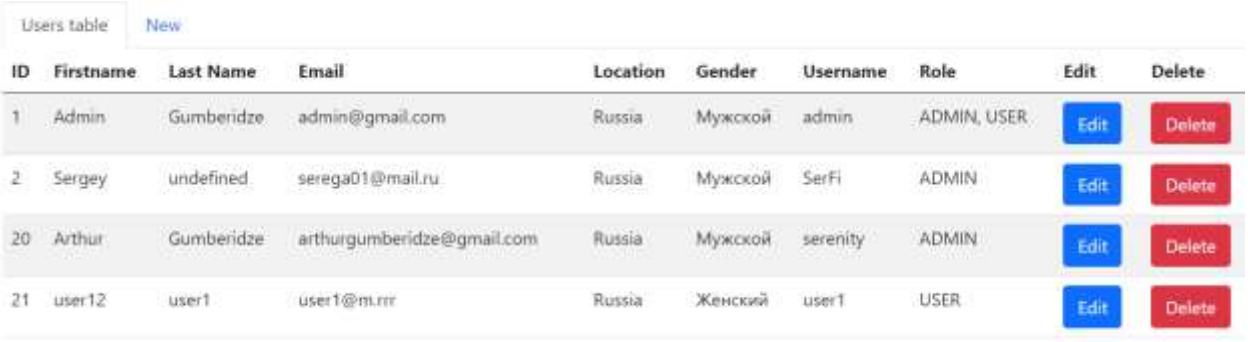

Рисунок В.5 – Тест 7, покрывающий таблицу решений

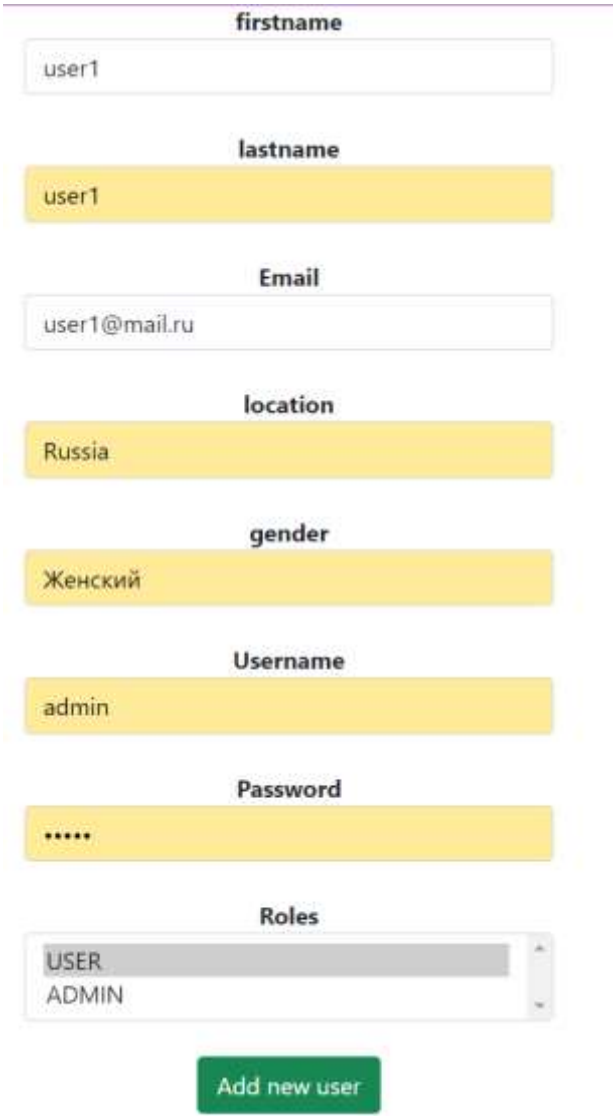

Рисунок В.6 – Тест 12, покрывающий таблицу решений

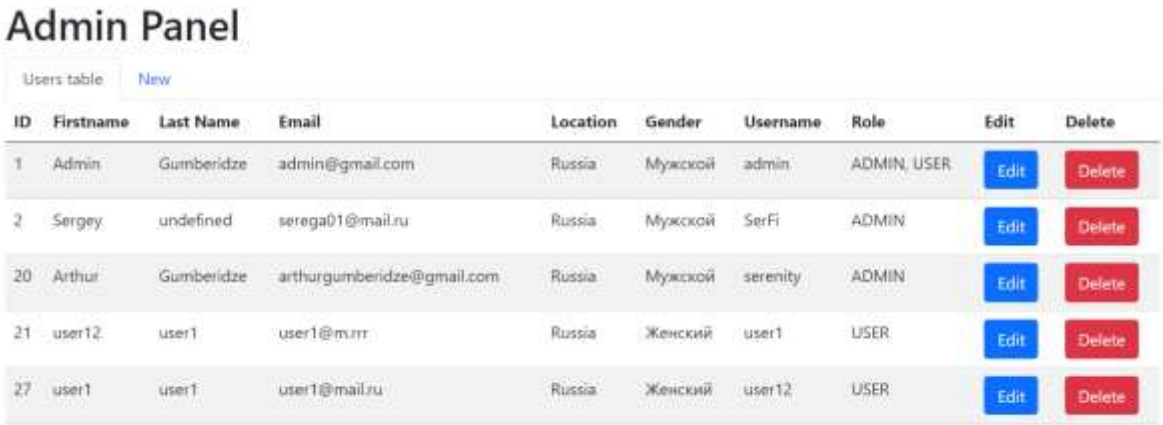

Рисунок В.7 – Тест 12, покрывающий таблицу решений

Приложение Г

ДИАГРАММЫ ПРОГРАММНЫХ КЛАССОВ

(рекомендуемое)

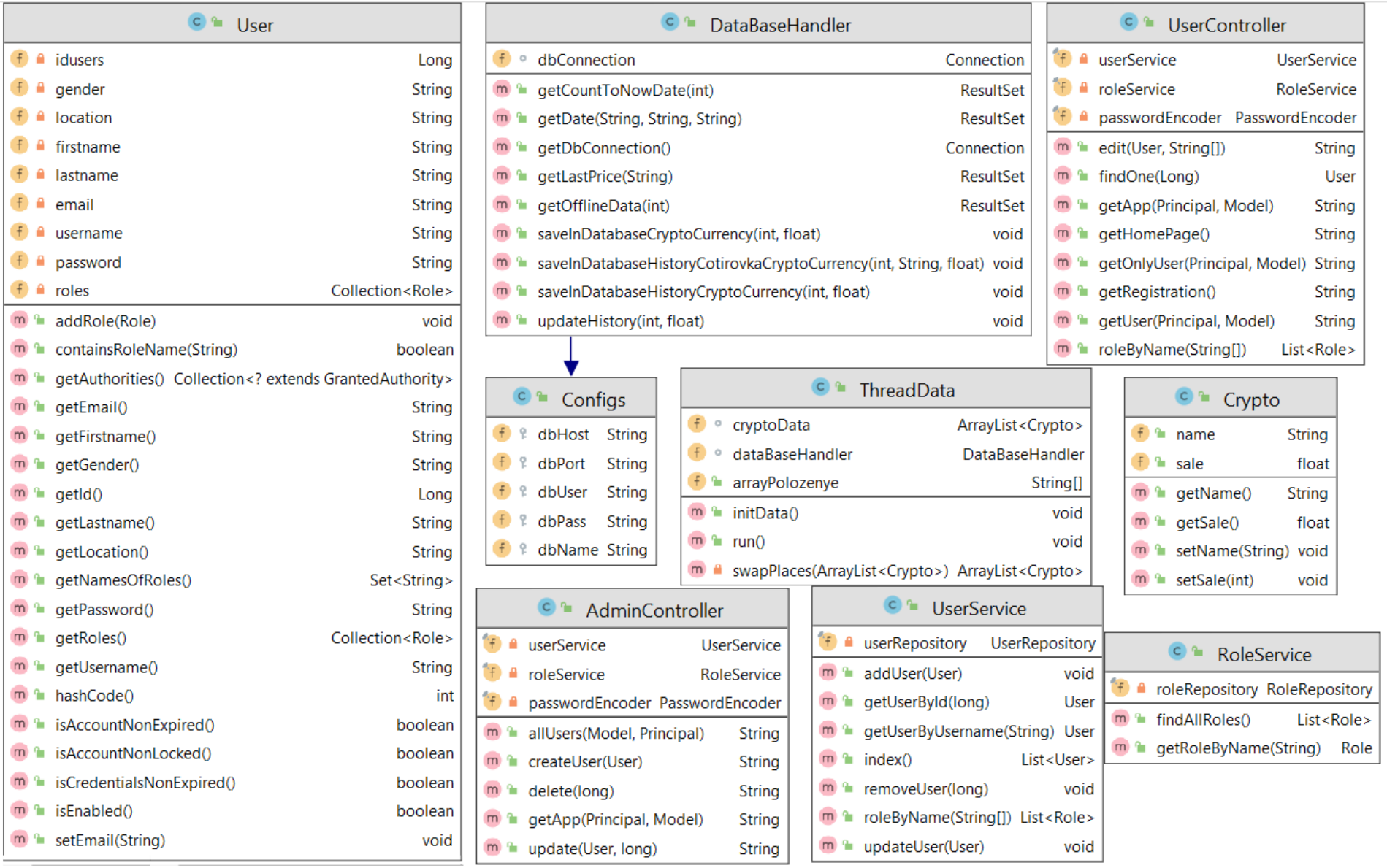

Рисунок Г.1 – Первая часть диаграммы классов первого модуля

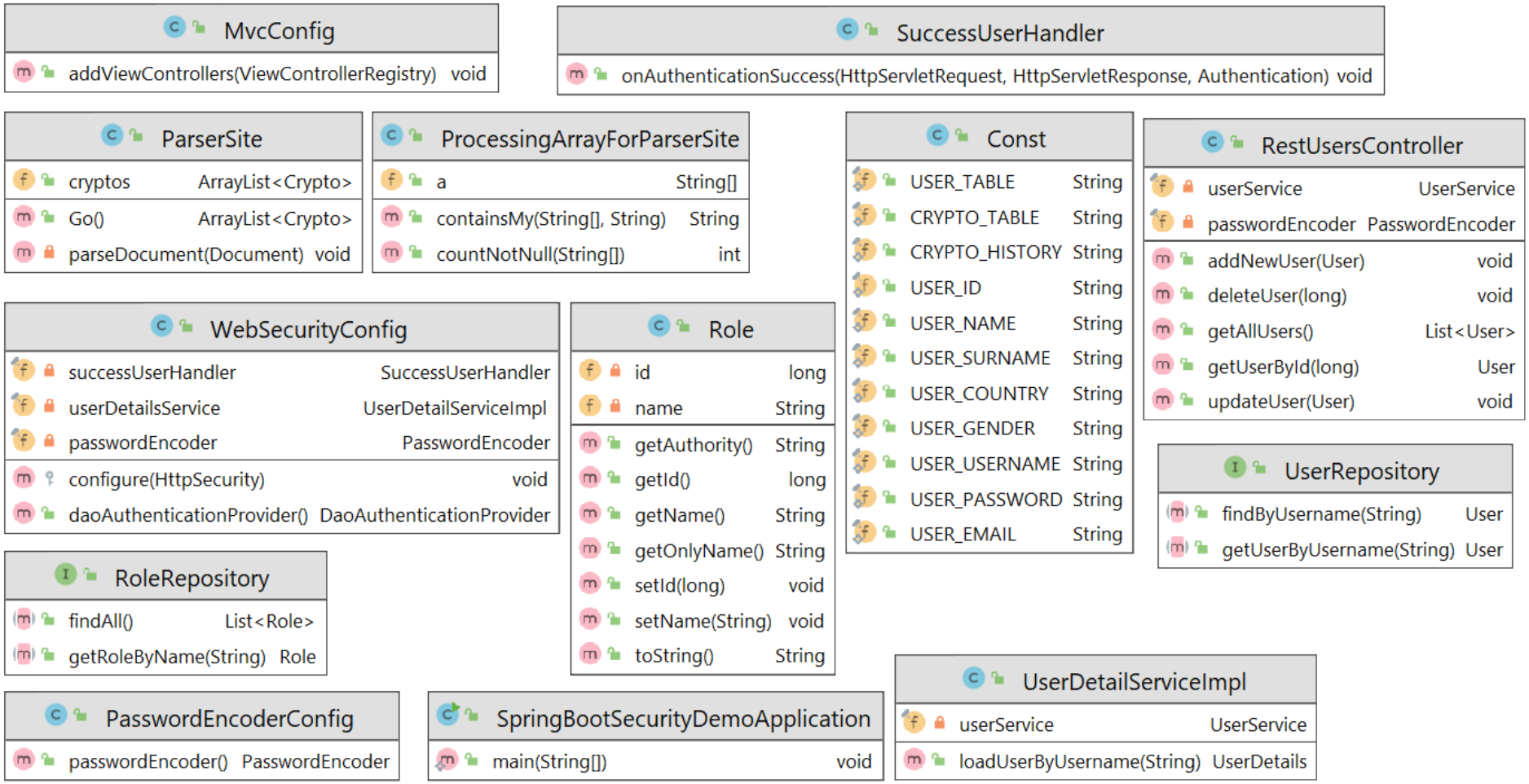

Рисунок Г.2 – Вторая часть диаграммы классов второго модуля

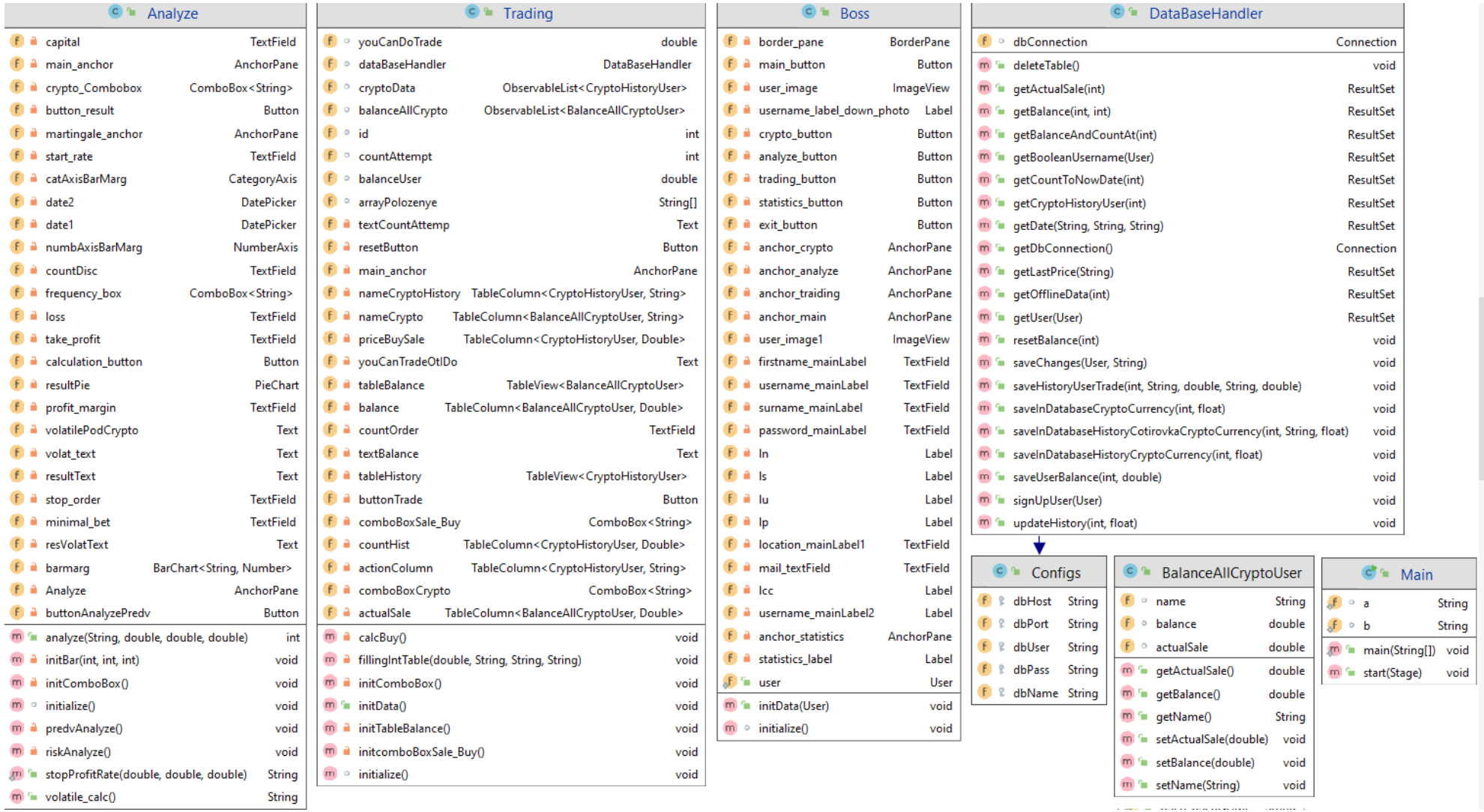

Рисунок Г.3 – Первая часть диаграммы классов второго модуля

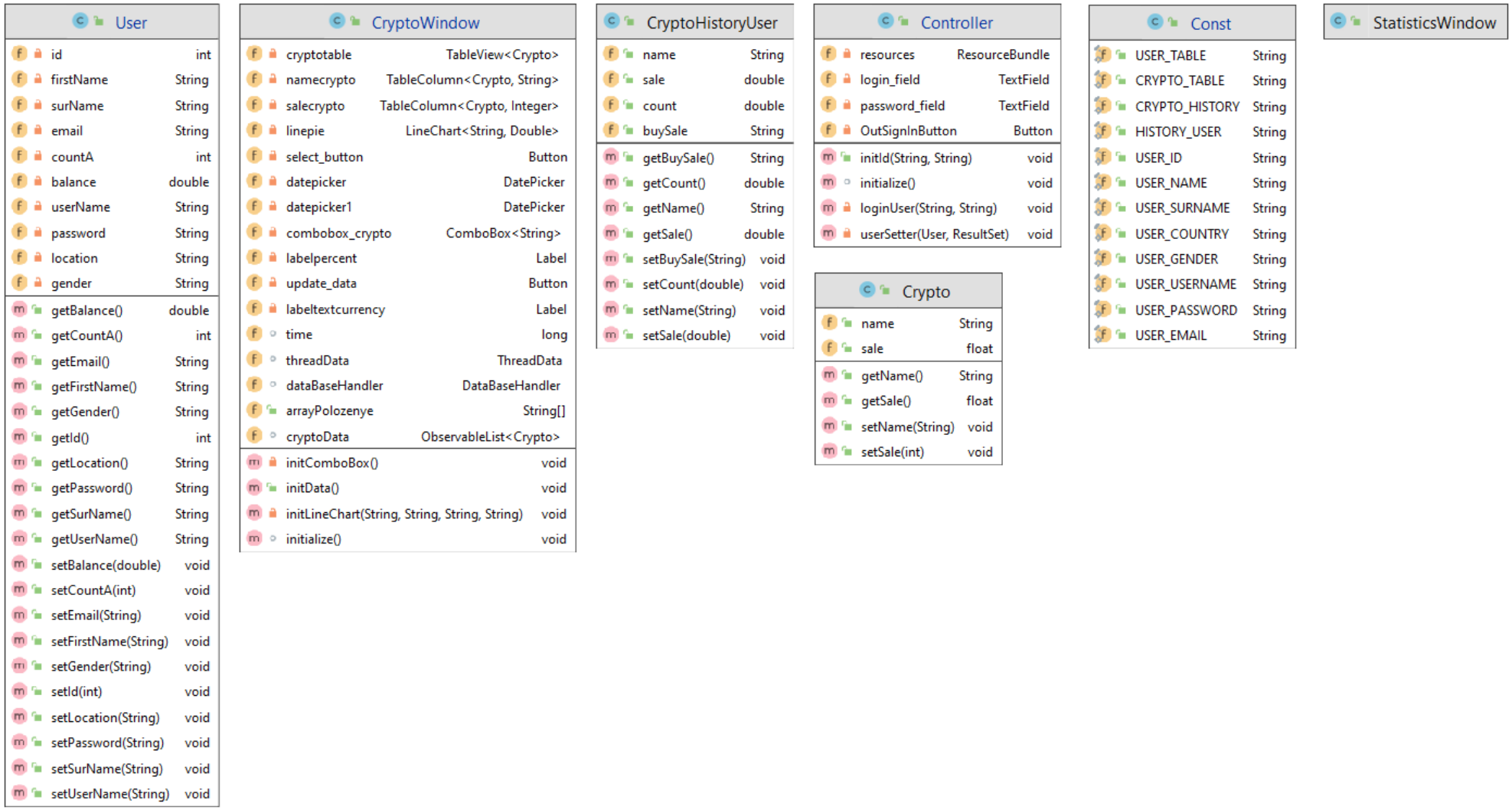

Рисунок Г.4 – Вторая часть диаграммы классов второго модуля