МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет**
**(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

# Разработка бэкэнда информационного сервиса о криптовалютах

КУРСОВАЯ РАБОТА
по дисциплине «Программная инженерия»
ЮУрГУ – 02.03.02.2023.308-287.КР

Нормоконтролер,
к.ф.-м.н., доцент каф. СП.
_____Т.Ю. Маковецкая
“___” _____ 2023 г.

Научный руководитель:
к.ф.-м.н., доцент каф. СП.
_____ Т.Ю. Маковецкая

Автор работы:
студент группы КЭ-302
_____ Е.С. Серяков

Работа защищена
с оценкой: _____
“___” _____ 2023 г

Челябинск, 2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет**
**(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский

«___». _____.2023

**ЗАДАНИЕ**
**на выполнение курсовой работы**
по дисциплине «Программная инженерия»
студенту группы КЭ-302 Серякову Егору Сергеевичу,
обучающемуся по направлению 02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы**

Разработка бэкэнда информационного сервиса о криптовалютах

2. **Срок сдачи студентом законченной работы:** 29.05.2023 г.

3. **Исходные данные к работе**

3.1. Adam Freeman. «Pro ASP.NET Core MVC 2» (7-издание, 2017 год);

3.2. Andrew Lock. «ASP.NET Core In Action» (3-е издание, 2022 год);

3.3. Metanit. Руководство по ASP.NET Core 6 [Электронный ресурс] URL:

https://metanit.com/sharp/aspnet6/ (дата обращения 24.02.2023 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ предметной области;

4.2. Изучить методы создания веб-приложения на платформе ASP.NET;

4.3. Спроектировать и реализовать необходимые модули;

4.4. Протестировать возможности разработанной системы.

5. **Дата выдачи задания:** 6 февраля 2023 г.

Научный руководитель Т.Ю. Маковецкая

к.ф.-м.н., доцент каф. СП.

Задание принял к исполнению Е.С. Серяков

**ОГЛАВЛЕНИЕ**

## ГЛОССАРИЙ

1. Бэкэнд (back-end) – часть системы, которая отвечает за обработку данных, бизнес-логику и взаимодействие с базой данных, серверами и другими внешними системами. Бэкэнд обычно состоит из серверного программного обеспечения, которое работает на удаленном сервере и обслуживает запросы от клиентской стороны.

2. Фронтэнд (front-end) – часть системы, которая отвечает за пользовательский интерфейс и взаимодействие с пользователем. Фронтэнд обычно включает в себя компоненты, которые видны и доступны непосредственно пользователю, такие как веб-страницы, элементы управления, графические элементы и другие пользовательские интерфейсы.

3. Web API или API (Application Programming Interface) – это набор определенных методов и точек доступа, предоставляемых веб-сервером или веб-приложением для взаимодействия с другими программами или компонентами через сеть. Он определяет, какие запросы могут быть сделаны к серверу и какие данные можно получить в ответ.

4. Криптовалюта – это форма цифровой валюты, которая использует криптографию для обеспечения безопасности и анонимности транзакций.

5. Криптобиржа (или криптовалютная биржа) – это онлайн-платформа, которая позволяет пользователям покупать, продавать и торговать различными криптовалютами.

6. Тикер (от англ. «Ticker» – метка) – это обычно короткий символьный идентификатор, который представляет конкретную криптовалюту. Они также могут использоваться для отслеживания ценовых данных и котировок криптовалюты на различных биржах и торговых платформах. Тикеры предоставляют компактную и удобную форму для идентификации конкретных активов и их ценовой активности на рынке.

7. Свечи (или японские свечи) – вид графического представления ценовой активности на определенном временном интервале. Они широко используются в анализе технического рынка для прогнозирования ценовых движений и принятия решений о торговле.

8. Стакан ордеров (от англ. «order book» – книга заказов) – это инструмент, используемый на криптовалютных биржах и в финансовых рынках для отображения текущих ордеров на покупку и продажу активов.

9. Фреймворк – это набор программных инструментов, библиотек, стандартов и правил.

10. Эндпоинт (англ. «endpoint» – конечная точка) – конечная точка коммуникации или интерфейс взаимодействия между двумя программными системами. Это конкретный URL (Uniform Resource Locator) или URI (Uniform Resource Identifier), который обозначает определенный ресурс или сервис на сервере.

## ВВЕДЕНИЕ

### Актуальность

На сегодняшний день существует множество способов совершать платежи: от примитивных бумажных банкнот до самых технологичных и сложных с технической точки зрения способов. Как раз таковым является способ оплаты, используя криптовалюту. Сейчас почти невозможно найти того молодого человека, который не слышал бы о данном явлении. Поэтому актуальность криптовалюты, однажды резко поднявшись в 2017 году, не снижается и по сей день.

В силу перечисленных мною факторов, создаётся спрос на различные сервисы, связанные с данной тематикой. Это могут быть как полноценные инструменты, в которых нуждаются профессионалы данной предметной области, так и относительно небольшие информационные сервисы, позволяющие новичкам узнать основную информацию, азы, получить первый толчок в исследовании нового и интересного. В следствии чего, мною было принято решение разработать бэкэнд для подобного сервиса.

### Постановка задачи

Целью данной работы является разработка бэкэнда информационного сервиса по криптовалютам, который будет являться первым шагом для создания полноценного веб-приложения. Для реализации поставленной цели необходимо выполнить следующие задачи:

1) проанализировать предметную область;
2) спроектировать архитектуру системы;
3) реализовать систему;
4) протестировать систему.

### Структура и содержание работы

Работа состоит из глоссария, введения, пяти глав, заключения, списка литературы и приложения. Объем работы составляет 59 страниц, объем

списка литературы – 10 источников.

В первой главе «Анализ предметной области» была описана предметная область проекта и проведен анализ существующих аналогов приложения.

Вторая глава «Требования к системе» посвящена проведению анализа требований к разрабатываемой системе, что включает в себя выявление функциональных и нефункциональных требований к разрабатываемой системе, также была построена диаграмма вариантов использования и раскрыты возможные варианты.

В третьей главе «Проектирование системы» разобрана архитектура системы, перечислены и описаны компоненты системы, отдельно представлены диаграммы компонентов, смоделирована база данных.

В четвёртой главе «Реализация» описаны особенности реализации системы, какие технологии были применены. Детально рассмотрены два варианта использования, представлены диаграммы последовательности для одного сценария использования приложения, для другого же реализована диаграмма деятельности.

В пятой главе «Тестирование» проведено функциональное тестирование реализованной системы с целью убедиться в её работоспособности и удовлетворению поставленных требований.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Описание предметной области

В первой главе рассмотрим предметную область проекта. Предметной областью проекта являются сервисы, предоставляющие информацию о криптовалютах и криптобиржах. Так же сюда можно отнести не только подобные сервисы, но и сами криптобиржи Это очень перспективная область как с технической точки зрения, так и с точки зрения извлечения прибыли. Именно возможности криптовалюты привлекают всё больше и больше людей. Наиболее известными криптовалютами являются Bitcoin, Ethereum, Ripple и BnB [1].

Криптовалюты – это очень обширная сфера, в которой новичку бывает трудно разобраться. Все популярные сервисы для мониторинга цен валют предоставляют информацию для продвинутых пользователей. Как раз эту проблему призван решить наш сайт. Он может помочь нашим читателям понять, что такое криптовалюты, как они работают и каким образом они могут повлиять на их жизнь и финансовое благополучие. Мы также стремимся, чтобы у пользователей сервиса была возможность закрепить свои знания, поэтому были добавлены тесты после каждого из урока.

## 1.2. Анализ аналогичных проектов и существующих решений для реализации системы

Существует целый ряд различных агрегаторов для мониторинга цены, капитализации, объёма торгов, циркулирующего предложения криптовалют. Такими являются, например: CoinMarketCap и CoinGecko. Но они не предоставляют какой-либо информации помимо параметров валют, как, например, делают такие новостные порталы как: Cointelegraph, CoinDesk и CryptoSlate. На них же пользователь может наблюдать развёрнутые статьи, касающиеся самых разных аспектов исследуемой области, начиная от

простейших обзорных новостей, заканчивая подробными исследованиями возможных вариантов развития той или иной валюты.

Рассмотрим подробнее несколько упомянутых выше сервисов. Самым популярным среди них является CoinMarketCap (https://coinmarketcap.com/) [2]. На рисунке 1 мы можем увидеть его стартовую страницу

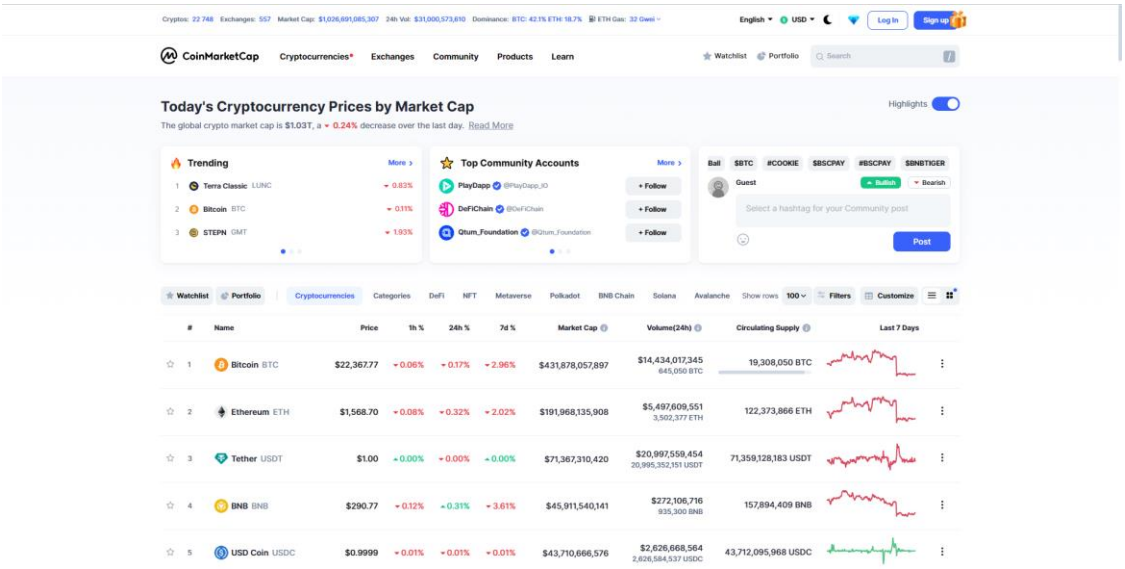


Рисунок 1 – Стартовая страница CoinMarketCap

CoinMarketCap (CMC) – это веб-сайт, предоставляющий информацию о криптовалютах и криптобиржах. Он был основан в 2013 году и является одним из самых популярных источников информации в криптовалютной индустрии. В среднем за месяц его посещает 187 миллионов человек [2].

Основной функцией этого сервиса является предоставление ценовых данных о криптовалютах и токенах, а также статистики о рыночной капитализации и объеме торгов. CMC также предоставляет пользовательский интерфейс для поиска криптовалют и токенов, с возможностью сортировки и фильтрации по различным параметрам. Кроме того, CMC также предоставляет информацию о криптобиржах, включая списки бирж, торгующих тем или иным токеном, а также оценки безопасности и объемы торгов.

CoinMarketCap имеет большую аудиторию и широко используется как инструмент для анализа криптовалютных рынков и инвестирования.

Если же мы хотим познакомиться с сервисами, направленными в новостное русло, то первым делом стоит обратить внимание на CoinDesk (https://www.coindesk.com/). На рисунке 2 можно увидеть его стартовую страницу.

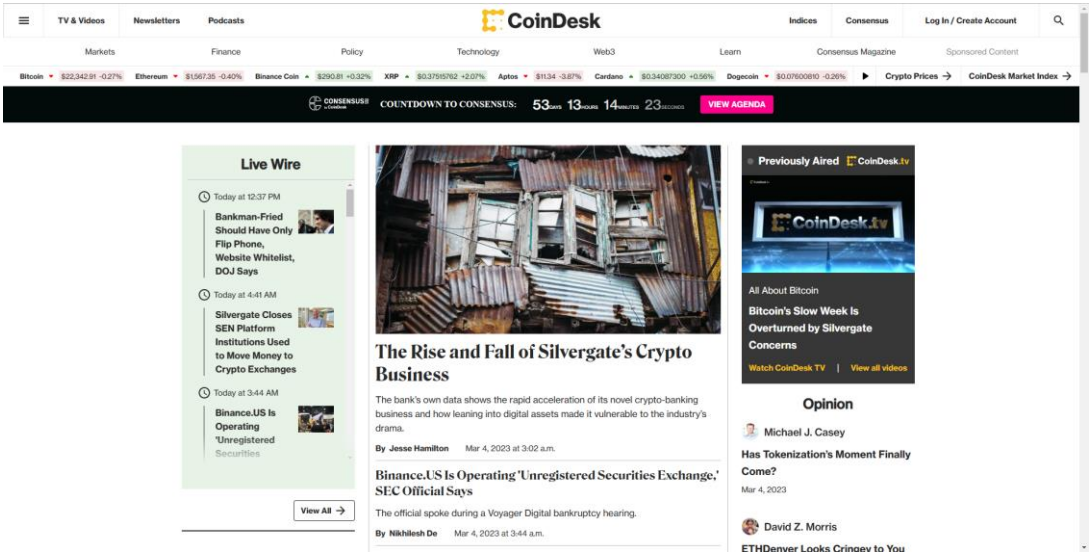


Рисунок 2 – Стартовая страница CoinDesk

CoinDesk – это один из самых популярных криптовалютных новостных порталов в мире. Он основан в 2013 году и предоставляет широкий спектр новостей, аналитики и мнений о криптовалютах и блокчейне. В 2020 году сайт вместе с компанией был выкуплен Digital Currency Group за 600 тысяч долларов. В 2020 году функционал площадки серьезно расширился, так как владельцы запустили свою исследовательскую платформу [3].

Этот сервис предоставляет своим пользователям последние новости о различных аспектах криптовалютной индустрии, включая новости о технологических разработках, правовых изменениях и финансовых рынках. Они также предлагают уникальные материалы, включая статьи с экспертными

мнениями, интервью с ключевыми фигурами криптовалютного сообщества и обзоры технологий и продуктов, связанных с криптовалютами.

Но у него есть аналоги, например, ProfInvestment (https://profinvestment.com/). По своей сути, это два одинаковых сервиса, отличающиеся набором статей, широтой охвата предметных областей. Так же ProfInvestment – русскоязычный новостной портал, а CoinDesk является англоязычным. На рисунке 3 можно увидеть стартовую страницу указанного русскоязычного портала

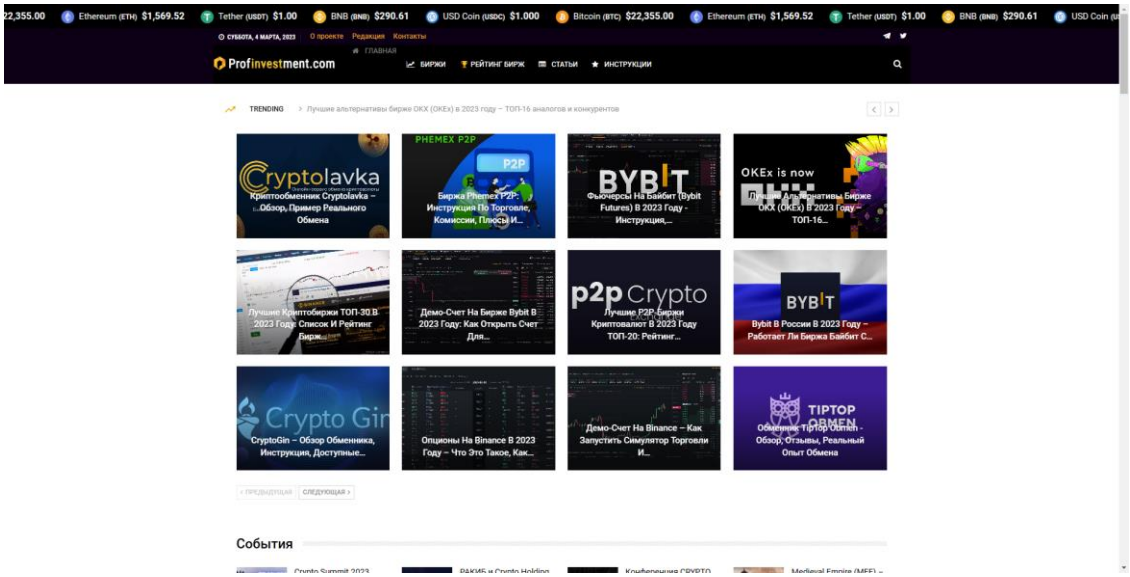


Рисунок 3 – Стартовая страница ProfInvestment

Данный продукт является онлайн-ресурсом, посвященным инвестированию в криптовалюты. Согласно информации на сайте, он предлагает своим пользователям широкий спектр материалов о том, как заработать деньги в Интернете, в том числе статьи о торговле на бирже, инвестировании в различные активы. Сайт предоставляет своим пользователям возможность подписки на рассылку новостей, обзоров и аналитики, а также имеет небольшой раздел с обучающими материалами, кратко рассказывающими про основные моменты, связанные с криптоактивами [4].

По окончанию изучения конкурентов, был определён некоторый набор возможностей, в таблице 1 представлена информация о наличии той или иной возможности в вышеуказанных продуктах.

Таблица 1 – Обзор аналогов

| **Возможность** | **CoinMarketCap** | **CoinDesk** | **ProfInvestment** |
|---|---|---|---|
| Предоставление актуальных данных о той или иной паре валют | + | + | + |
| Предоставление графиков свечей о той или иной криптовалюте | + | − | − |
| Поддержка пользователей | + | + | − |
| Наличие новостных статей | + | + | + |
| Наличие обучающих статей | − | − | − |
| Наличие тестов для закрепления полученных знаний из обучающих статей | − | − | − |

На сегодняшний день нет ни одного решения для реализации конкретно такого проекта, как наш. Все представленные выше решения являются либо очень сложными, либо заключены в рамках чисто новостного ресурса, который может быть интересен уже продвинутым пользователям, но никак не тем, кто только знакомится с криптовалютой. Подобное положение дел оставляет новичков наедине со сложными на первый взгляд терминами, аббревиатурами, цифрами и формулами, подобная ситуация легко может отпугнуть новичка. Как раз создание нашего сервиса позволит им избежать сложностей в начале его пути криптоинвестора.

Проект будет представлять из себя сервис с отдельной частью бэкэнда и фронтэнда, такая структура позволит масштабировать его по мере разрастания. Конкретно передо мной лежит задача разработки бэкэнда данного сервиса.

### 1.3. Вывод к первой главе

В результате сравнения аналогов был выявлен необходимый для реализации функционал, учтены преимущества и недостатки конкурентов. Итоговая система должна удовлетворять критериям, перечисленным в таблице 1. Фактически, техническим названием данного приложения будет определение «Web API». Названием же самого сервиса мною было выбрано «CryptoBook».

# 2. ТРЕБОВАНИЯ К СИСТЕМЕ

## 2.1. Анализ требований к программной системе

На основе информации, полученной в результате изучения предметной области и обзора приложений-аналогов, были сформированы следующие функциональные и нефункциональные требования к Web API «CryptoBook». В качестве пользователя моей системы будет выступать фронтэнд.

**Функциональные требования**

Ниже представлены функциональные требования к системе.

1. Пользователь должен иметь возможность получить основные данные о всех криптовалютах из установленного списка;
2. Пользователь должен иметь возможность получить основные данные о конкретной криптовалюте из установленного списка;
3. Пользователь должен иметь возможность получить данные о свечах конкретной пары за установленное время;
4. Пользователь должен иметь возможность воспользоваться системой регистрации и аутентификации;
5. Пользователь должен иметь возможность производить действия с учётной записью;
6. Пользователь должен иметь возможность получить теоретические и учебные материалы;
7. Пользователь должен иметь возможность воспользоваться системой тестов по теоретическим и учебным материалам.

**Нефункциональные требования**

Ниже представленные нефункциональные требования к системе.

1. Приложение должно быть стабильным и надёжным, не должно выходить из строя или зависать во время использования.

2. Приложение должно быть быстрым и отзывчивым, не должно занимать много системных ресурсов.

3. Приложение должно быть защищено от вирусов и других вредоносных программ, не должно быть уязвимостей.

4. Приложение должно быть легким в использовании, должно иметь понятный интерфейс.

## 2.2. Варианты использования

Для моделирования ранее выделенных функциональных требований к приложению с помощью языка объектного моделирования UML была создана диаграмма вариантов использования, которая призвана отобразить отношения между актерами и прецедентами. В моделируемой системе присутствует только один актер – система фронтэнда. Построенная диаграмма вариантов использования представлена на рисунке 4.

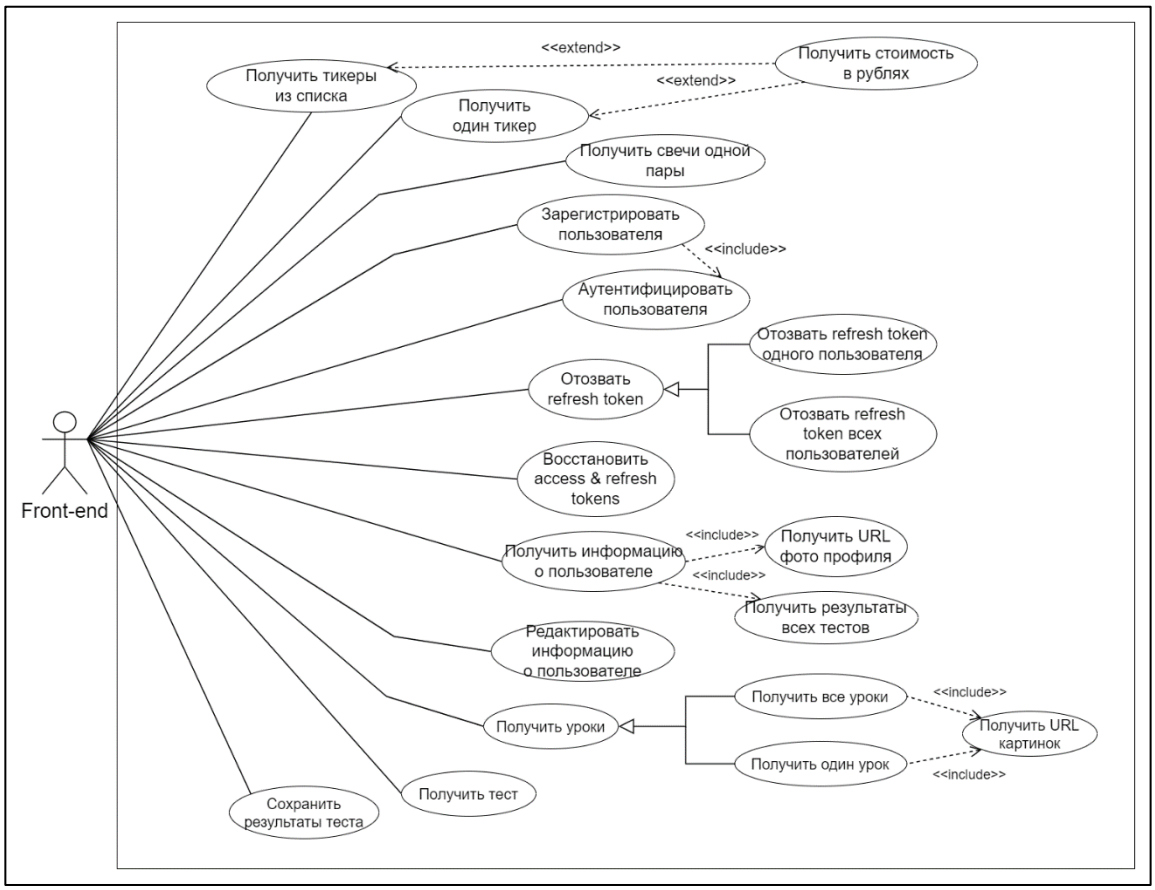


Рисунок 4 – Диаграмма вариантов использования

Рассмотрим действия, которые может совершать актер.

1. Получение тикеров из списка – получение основной информации о криптовалютных парах;
2. Получение одного тикера – получение основной информации о криптовалютной паре;
3. Получение свечей одной пары – получение данных о японских свечах за определённый промежуток времени о криптовалютной паре;
4. Зарегистрировать пользователя – создание учётной записи нового пользователя;
5. Аутентифицировать пользователя – вход в систему под учётной записью пользователя;
6. Восстановить access & refresh tokens – отзыв access & refresh tokens с восстановлением доступа к сессии пользователя;
7. Отозвать refresh token – отзыв refresh token без восстановления доступа к сессии(-ям) пользователя(-ей);
8. Получить информацию о пользователе – получение всей информации о пользователе;
9. Редактировать информацию о пользователе – запись новой информации в учетную запись пользователя;
10. Получить уроки – получение учебно-методических материалов согласно тематике сервиса;
11. Получить тест – получение данных о тесте по конкретному уроку;
12. Сохранить результаты теста – сохранение результатов пройденного теста.

Так же мною была составлена спецификация вариантов использования (См. приложение №1).

## 2.3. Вывод ко второй главе

В разделе были выведены функциональные и нефункциональные требования к системе, на основе которых была составлена диаграмма вариантов использования. В окончании этого был выделен главный и единственный актёр.

# 3. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## 3.1. Архитектура приложения

Самым распространённым архитектурным подходом для проектирования приложений типа Web API, пожалуй, является тот, что известен нам как MVC (Model-View-Controller) [9], в случае с API его так же могут называть API MVC. Именно этот подход был выбран мной. Схему MVC можно увидеть на рисунке 5.

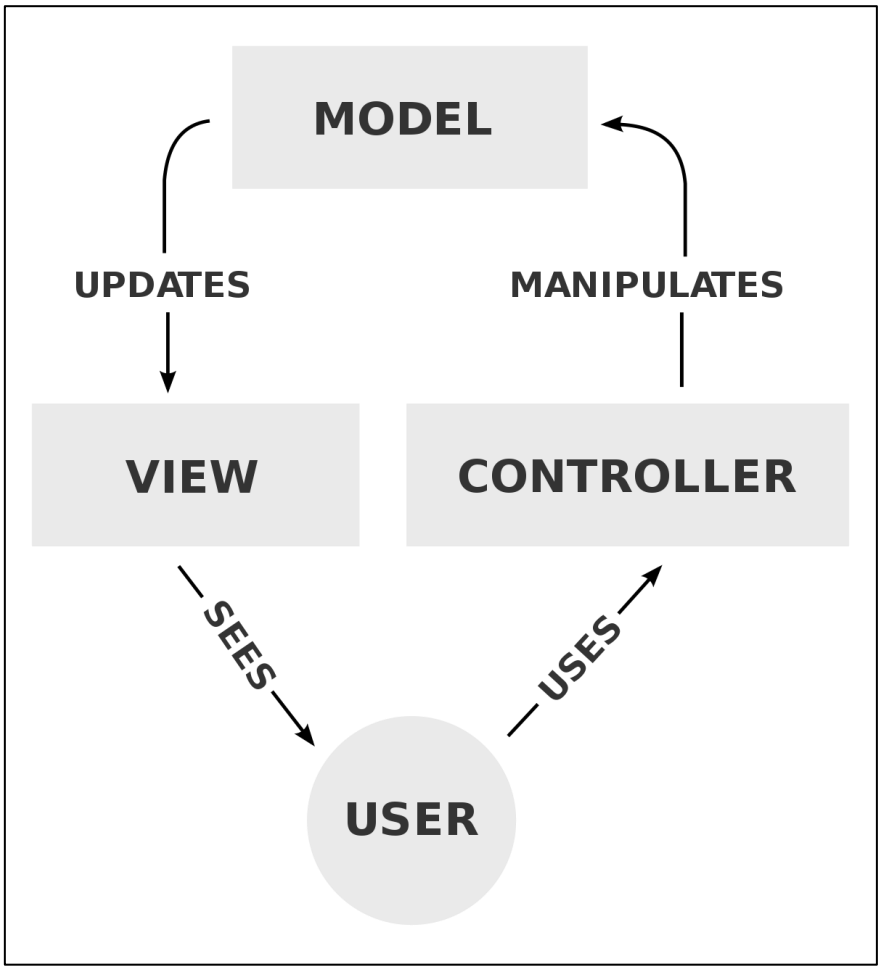


Рисунок 5 – Схема паттерна проектирования MVC

Этот подход разделяет приложение на три основных компонента:

1. Модель (Model): Модель представляет данные и бизнес-логику приложения. Она может включать классы, объекты или структуры данных,

а также методы для их манипуляции и взаимодействия с базой данных или внешними сервисами;

2. Представление (View): В случае с Web API, представление обычно отсутствует, поскольку API возвращает данные в формате JSON или XML. Однако, в некоторых случаях, представление может использоваться для форматирования данных перед отправкой в ответе;

3. Контроллер (Controller): Контроллер обрабатывает входящие запросы от клиента и координирует работу модели и представления. Он содержит методы действий (action methods), которые принимают запросы и возвращают соответствующие ответы. В контексте Web API, контроллеры отвечают за маршрутизацию запросов и обработку бизнес-логики для получения и обработки данных.

Кроме основных компонентов MVC, ASP.NET Core Web API также может использовать другие паттерны и архитектурные подходы в зависимости от требований проекта. Например, инверсия управления (Inversion of Control – IoC) и внедрение зависимостей (Dependency Injection – DI) [10]. Схему примера использования DI в разрабатываемом приложении можно увидеть на рисунке 6.

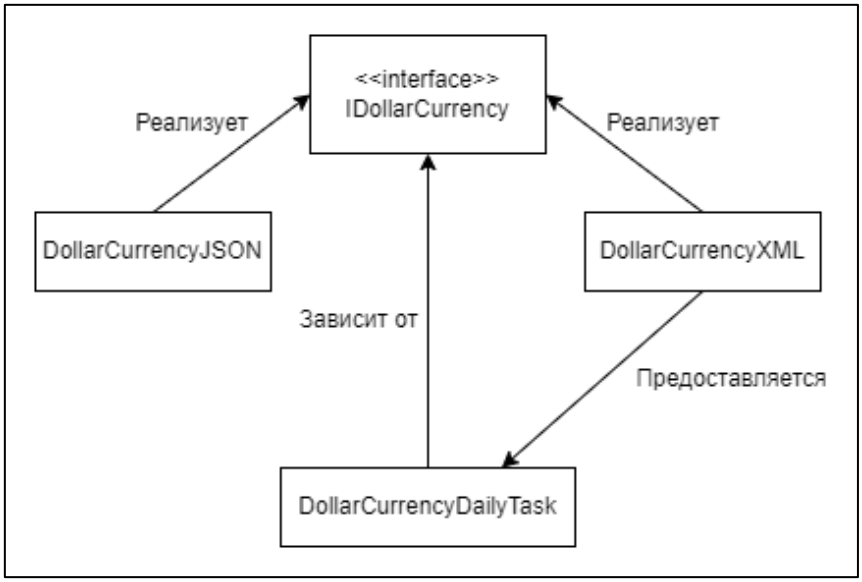


Рисунок 6 – Схема использования паттерна DI

Принцип проектирования IoC является родительским по отношению к DI. IoC преследует цель создания слабосвязанных классов, что позволяет сделать их более тестируемыми, поддерживаемыми и расширяемыми. Основная идея тут в перенаправлении управления внешнему обработчику/контроллеру. Паттерн внедрения зависимостей является более специфичной версией IoC. ASP.NET Core предоставляет встроенный контейнер DI, который можно использовать для регистрации зависимостей и их разрешения в различных компонентах приложения. С использованием данных паттернов появляется возможность замены реализации без изменения кода в контроллере, классе или другом компоненте.

### 3.2. Компоненты системы

Общая диаграмма компонентов разрабатываемой системы, спроектированной с помощью паттерна MVC, представлена на рисунке 7.

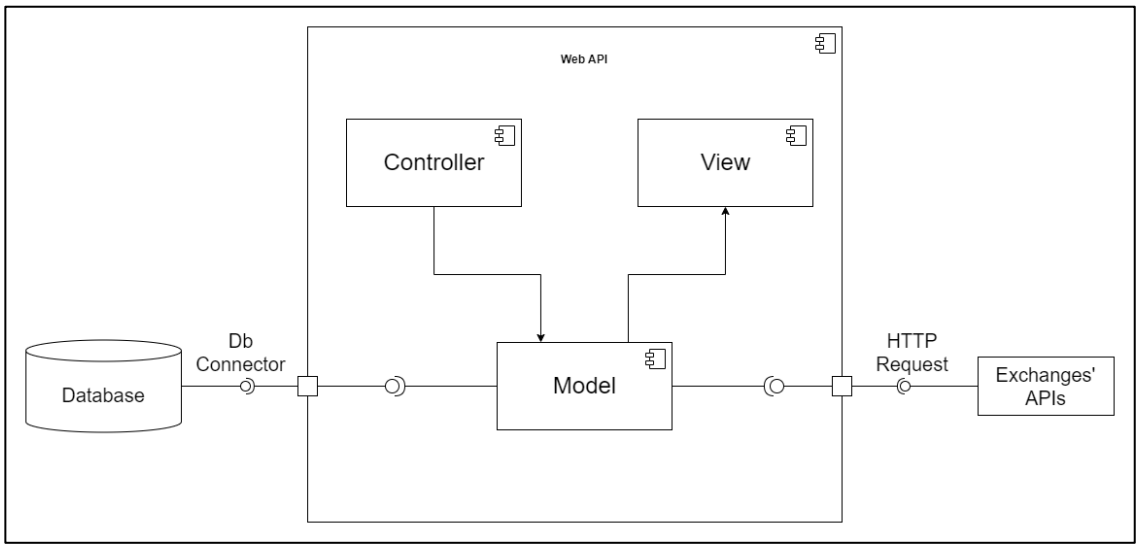


Рисунок 7 – Общая диаграмма компонентов Web API

Отдельно хотелось бы рассмотреть компонент Модели (Model) и компонент Контроллера (Controller), что можно увидеть на рисунке 8

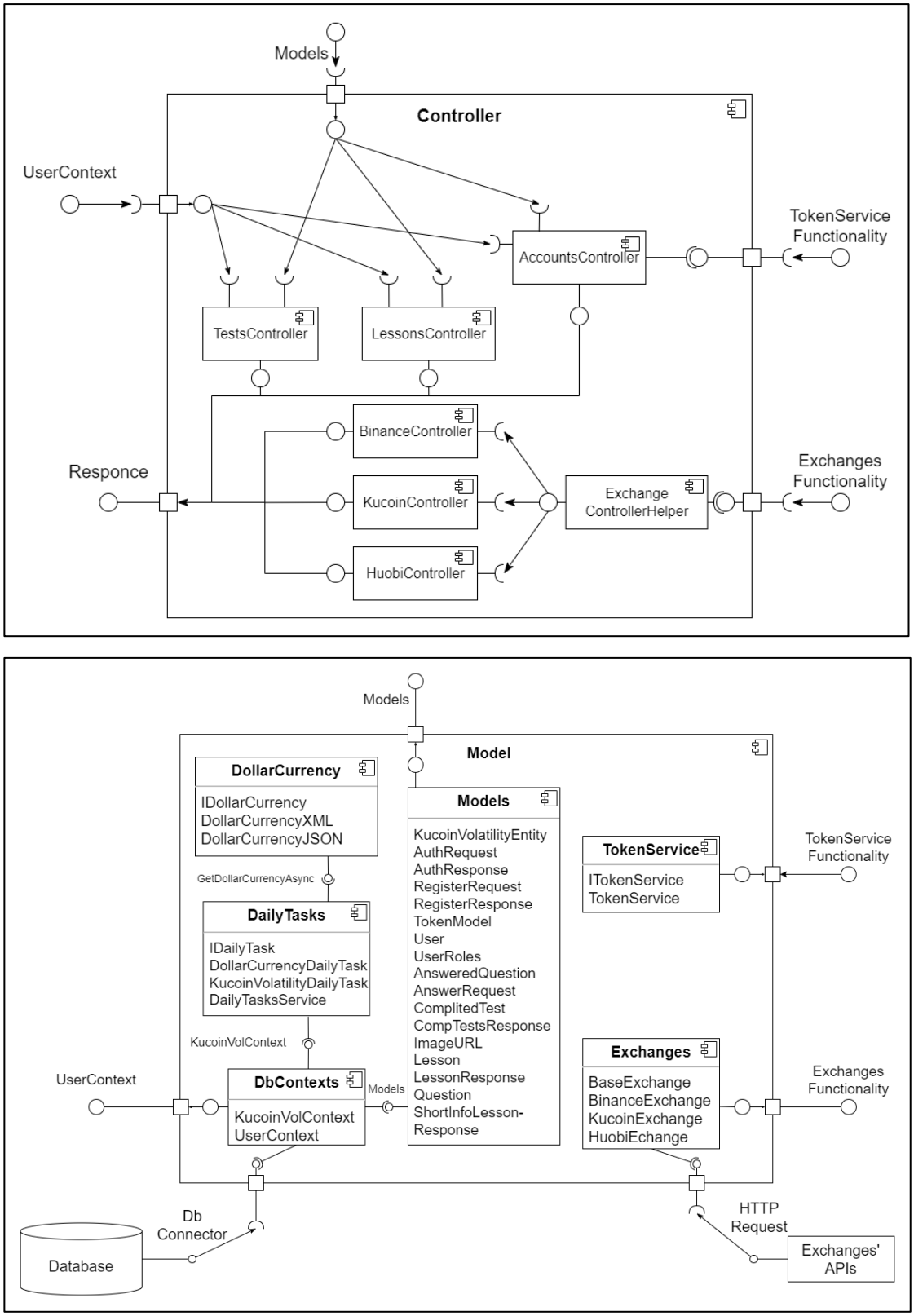


Рисунок 8 – Диаграммы компонентов Model и Controller

Компонент представления (View) весь инкапсулирован в логику работы платформы ASP.NET Core, в следствие чего мы не может узнать тонкости реализации данного компонента. Фактически, в рамках Web API главной внутренней частью компонента View является сериализатор NewtonsoftJson, который можно подключить отдельной одноимённой библиотекой. Общую диаграмму компонентов View можно увидеть на рисунке 9.

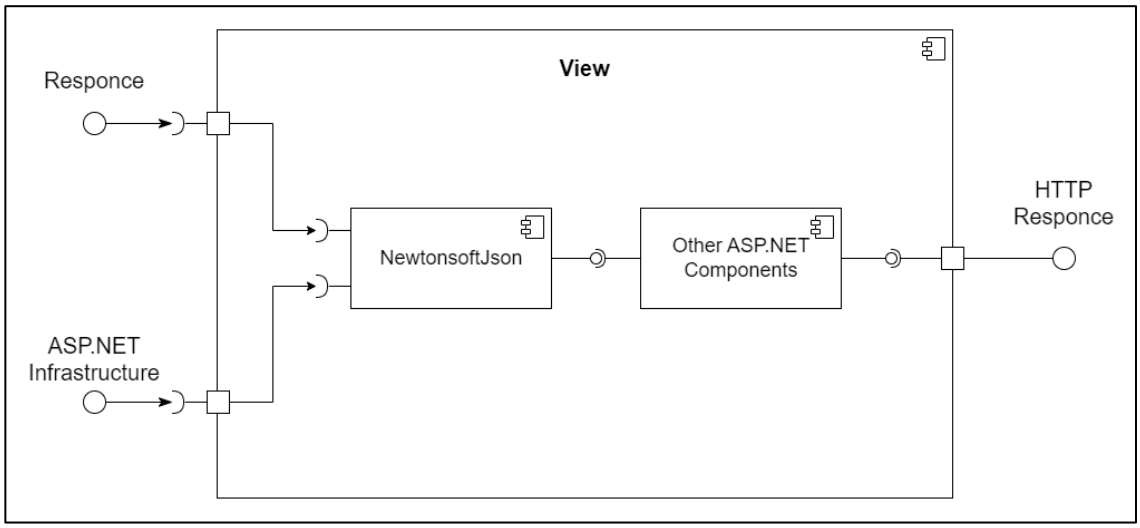


Рисунок 9 – Диаграмма компонентов View

Ниже представлено короткое описание компонентов Model:

1. DollarCurrency – пространство имён, в котором определенны следующие элементы:

1.1. IDollarCurrency – интерфейс для курса доллара;

1.2. DollarCurrencyXML – реализация интерфейса IDollarCurrency;

1.3. DollarCurrencyJSON – реализация интерфейса IDollarCurrency;

2. TokenService – пространство имён, в котором определенны следующие элементы:

2.1. ITokenSevice – интерфейс для работы с токенами;

2.2. TokenService – реализация интерфейса ITokenService.

3. DailyTasks – пространство имён, в котором определенны следующие элементы:

3.1. IDailyTask – интерфейс ежедневной задачи;

3.2. DollarCurrencyDailyTask – реализация интерфейса IDailyTask, нужная для получения актуального курса доллара;

3.3. KucoinVolatilityDailyTask – реализация интерфейса IDailyTask, получающая данные для подсчёта взвешенной волатильности и записывающая их в базу данных;

4. DbContexts – пространство имён, в котором определенны следующие элементы:

4.1. KucoinVolContext – контекст таблиц базы данных, связанных с подсчётом волатильности пар с криптобиржи Kucoin;

4.2. UserContext – контекст таблиц базы данных, связанный с работой с пользователями, уроками и тестами.

5. Exchanges – пространство имён, в котором определенны следующие элементы:

5.1. BaseExchange – класс-родитель для классов трёх криптобирж;

5.2. BinanceExchage – класс криптобиржи Binance;

5.3. KucoinExchange – класс криптобиржи Kucoin;

5.4. HuobiExchange – класс криптобиржи Huobi.

6. Models – пространство имён, в котором определенны следующие элементы:

6.1. KucoinVolatilityEntity – модель для сохранения данных для подсчёта взвешенной волатильности;

6.2. AuthRequest – модель запроса на аутентификацию;

6.3. AuthResponse – модель ответа на запрос на аутентификацию;

6.4. RegisterRequest – модель запроса на регистрацию;

6.5. RegisterResponse – модель ответа на запрос на регистрацию;

6.6. TokenModel – модель для хранения токена аутентификации;

6.7. User – модель пользователя;

6.8. UserRoles – модель для хранения ролей пользователя;

6.9. AnsweredQuestion – модель для хранения отвеченных вопросов;

6.10. AnswerRequest – модель запроса ответа на вопрос;

6.11. ComplitedTest – модель для хранения завершенных тестов;

6.12. CompTestsResponse – модель ответа с завершенными тестами;

6.13. ImageURL – модель для хранения URL изображения;

6.14. Lesson – модель урока;

6.15. LessonResponse – модель ответа с информацией о уроке;

6.16. Question – модель вопроса;

6.17. ShortInfoLessonResponse – модель ответа с краткой информацией об уроке.

Ниже представлено короткое описание компонентов Controller:

1. ExchangeControllerHelper – контроллер, объединяющий в себе логику делегирования задачей классам криптобирж, где содержится бизнес-логика;
2. BinanceController – контроллер для криптобиржи Binance;
3. KucoinController – контроллер для криптобиржи Kucoin;
4. HuobiController – контроллер для криптобиржи Huobi;
5. AccountsController – контроллер для работы с пользователями;
6. LessonsController – контроллер для работы с уроками;
7. TestsController – контроллер для работы с тестами.

Ниже представлено короткое описание компонентов View:

1. NewtonsoftJson – система сериализации данных в формат JSON;
2. Other ASP.NET Components – другие компоненты платформы ASP.NET, которые производят обработку ответа.

### 3.3. Модель базы данных

База данных является неотъемлемой частью веб-приложений, так как она позволяет хранить и обрабатывать данные в одном централизованном месте. Благодаря базе данных мы можем сохранять и получать информацию для последующего использования. Как мною было упомянуто ранее, использоваться будет в связке с данной системой СУБД PostgreSQL. На рисунке 10 представлена первая часть схемы базы данных разрабатываемого ресурса. Именно здесь можно увидеть модели таблиц, хранящие данные о пользователях, уроках, тестах и связи между ними.

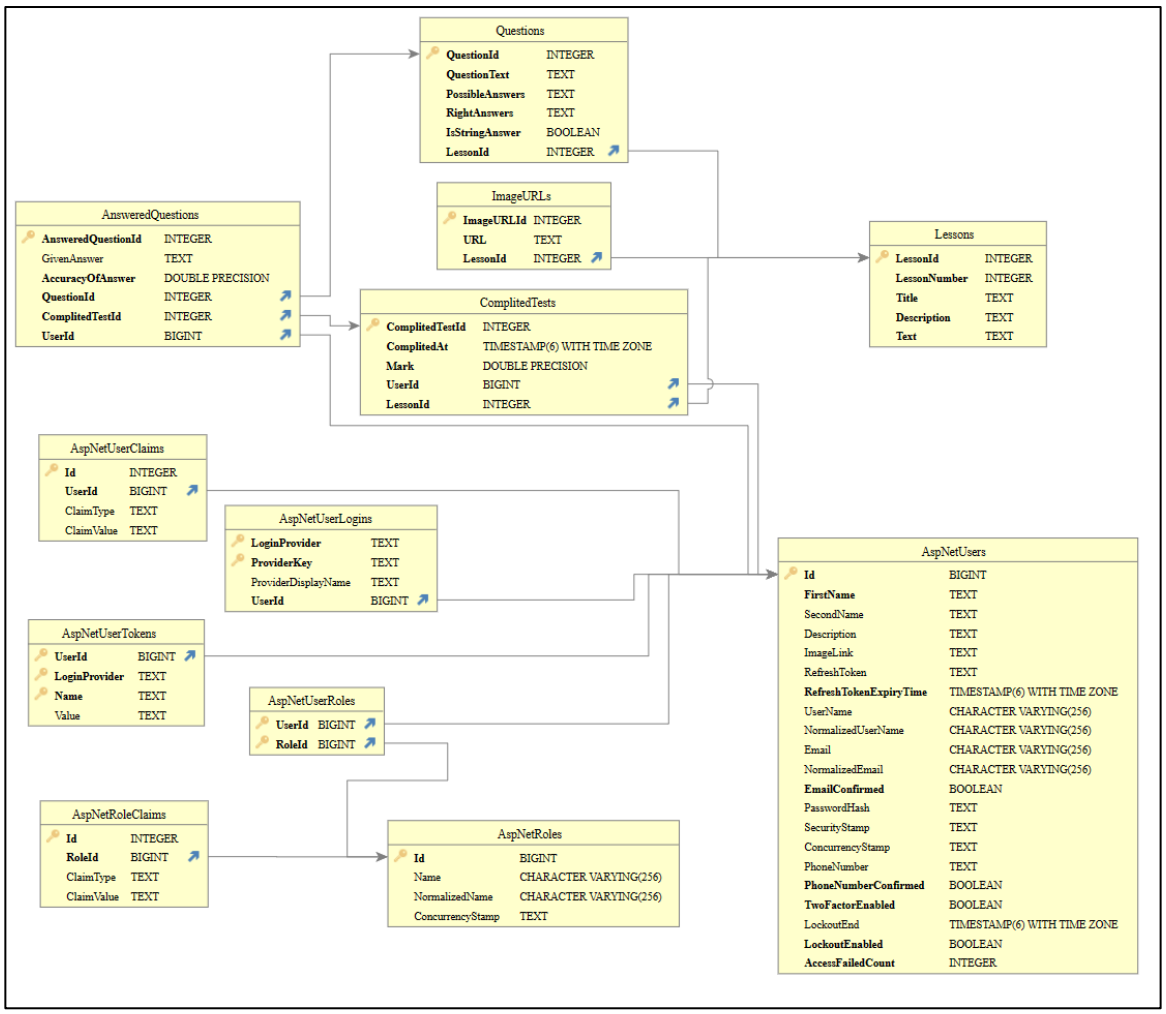


Рисунок 10 – Диаграмма части базы данных, связанной с пользователями, уроками и тестами

Далее, в таблице 2, я расписал за что отвечает каждая из таблиц в БД, представленных в первой части диаграммы.

Таблица 2 – описание таблиц БД

| **Название таблицы** | **Описание** |
|---|---|
| AspNetUsers | Содержит основную информацию о пользователях |
| AspNetRoles | Содержит доступные роли, которые могут быть присвоены пользователям |
| AspNetUserRoles | Устанавливает связь между пользователями и их ролями, содержит идентификаторы ролей и пользователей |
| AspNetUserClaims | Содержит пользовательские утверждения (claims) |
| AspNetUserLogins | Содержит информацию о внешних учетных записях, связанных с учетной записью пользователя, например, Google, Facebook |
| AspNetUserTokens | Содержит токены аутентификации пользователей |
| Lessons | Содержит информацию об уроках |
| ImageURLs | Содержит URL картинок |
| Questions | Содержит все вопросы |
| ComplitedTests | Устанавливает связь между пользователем и ответами на вопросы, содержит информацию о пройденном тесте |
| AnsweredQuestions | Содержит ответы на вопросы каждого пользователя |

После этого стоит уделить внимание обособленной части базы данных с теми таблицами, которые нужны для хранения служебных данных о парах валют, торгуемых на криптобирже Kucoin. Эти таблицы имеют идентичное содержание и назначение, лишь только каждая из них хранит данные о

какой-либо паре валют, что указано в названиях таблиц. На рисунке 11 можно увидеть диаграмму, содержащую в себе модели этих таблиц.

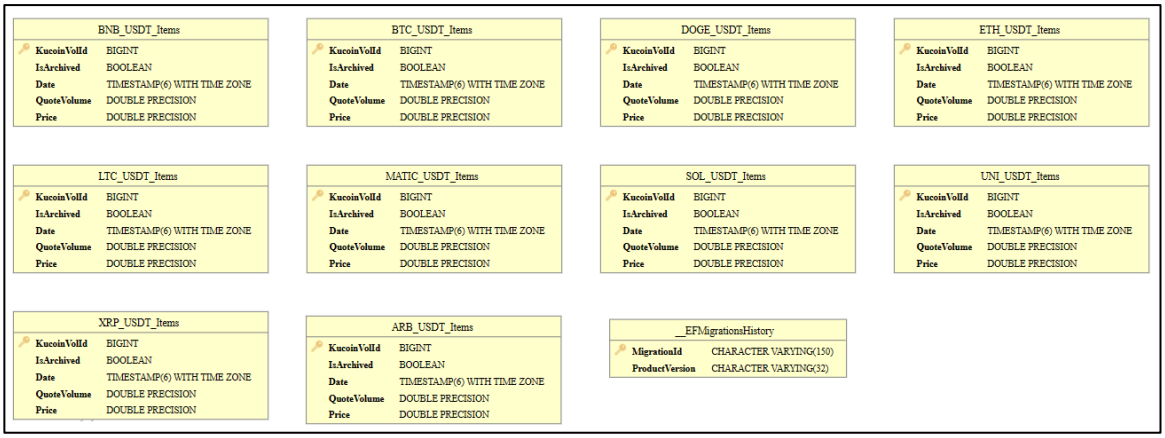


Рисунок 11 – Диаграмма части базы данных, отвечающей за хранение служебных данных о парах криптовалют

Их использование обусловлено тем, что криптобиржа Kucoin не предоставляет данные о количестве продаж за рассматриваемый промежуток времени, по умолчанию это 24 часа. Именно эта переменная нужная для расчёта волатильности с помощью метода среднеквадратичного отклонения, с помощью этого метода считается волатильность для пар валют, полученных с двух других криптобирж: Binance и Huobi. Рассчитывается он по формуле (1):

$$\sigma = \sqrt{\frac{(P(t) - P_{aver})^2}{n}}, \qquad (1)$$

где $\sigma$ – волатильность;

$P(t)$ - цена криптовалюты в конкретный момент времени;

$P_{aver}$ - средняя цена криптовалюты за определенный период;

$n$ – количество цен в выборке.

Для подсчёта волатильности пар, полученных с криптобиржи Kucoin используется метод взвешенной волатильности. Он отличается большей

точность в силу того, что тут происходит выборка по нескольким периодам, именно для хранения данных об этих периодах нужны таблицы БД, о которых было сказано выше. Первым нужно посчитать логарифмическую доходность для каждого периода времени по формуле (2):

$$LogReturn_i = \ln \frac{P(t_0)}{P(t_i)}, \qquad (2)$$

где $LogReturn_i$ – логарифмическая доходность;

$P(t_0)$ – цена в конкретный момент времени;

$P(t_i)$ – предыдущая цена.

Следующим действием будет подсчёт взвешенного среднего логарифмических доходностей, для этого нам пригодится формула (3):

$$WAR = \frac{LogReturn_0 + LogReturn_1 + \ldots + LogReturn_n}{Volume_0 + Volume_1 + \ldots + Volume_n}, \qquad (3)$$

где $WAR$ – взвешенное среднее лог. доходностей;

$Volume_i$ – объём торгов;

$LogReturn_i$ – логарифмическая доходность;

$n$ – количество периодов в выборке.

Предпоследним шагом будет подсчёт отклонения от взвешенного среднего логарифмических доходностей для каждого из периода, для этого стоит воспользоваться формулой (4):

$$Dev_i = (LogReturn_i - WAR)^2, \qquad (4)$$

где $Dev_i$– отклонение от WAR;

$WAR$ – взвешенное среднее лог. доходностей.

Для подсчёта волатильности описанным методом требует применить последнюю формулу (5):

$$\sigma = \sqrt{\frac{Dev_0 + Dev_1 + \ldots + Dev_n}{n}}, \qquad (5)$$

где $\sigma$ – волатильность;

$Dev_i$ – отклонение от WAR;

$n$ – количество периодов в выборке.

### 3.4. Вывод к третьей главе

В этой главе были представлена архитектура системы, описаны паттерны проектирования, что является важным шагом к созданию программы, которую можно удобно масштабировать и поддерживать. Так же были представлены и описаны диаграммы компонентов программы. После чего разобрана модель базы данных с подробным её описанием.

# 4. РЕАЛИЗАЦИЯ

## 4.1. Особенности реализации системы

Рассмотрев множество языков и фреймворков, мною была выбрана высокопроизводительная платформа разработки веб-приложений ASP.NET Core. Она предоставляет мощный инструментарий для создания веб-приложений, включая поддержку маршрутизации, контроллеров, моделей представления и многое другое.[6]. ASP.NET Core также предоставляет различные возможности для создания безопасных веб-приложений, например, инструменты для защиты от атак вроде SQL-инъекций [7].

В качестве СУБД мною было выбрано одно из самых популярных – PostgreSQL [8]. Работа с ним удобна, его производительность является достойной для масштабных проектов, которые являются намного более нагруженными системами, чем та, что разработана мной.

В качестве интегрированной среды разработки была выбрана IDE Visual Studio 2022, которая является одной из самых популярных для разработки приложений на платформе .NET в силу наличия широкого набора инструментов и возможностей, которые значительно упрощают и ускоряют процесс разработки.

В процессе разработки были использованы следующие программные продукты и библиотеки:

**Binance.NET (8.6.2)** [11]

Библиотека для взаимодействия с API Binance.

**Kucoin.NET (4.3.3)** [12]

Библиотека для взаимодействия с API Kucoin.

**Huobi.NET (4.2.4)** [13]

Библиотека для взаимодействия с API Huobi.

**Microsoft.AspNetCore.Authentication.JwtBearer (6.0.16)** [14]

Эта библиотека предоставляет поддержку аутентификации JSON Web Token (JWT) в ASP.NET Core приложениях, позволяя проверять и аутентифицировать пользователей на основе JWT токенов.

**Microsoft.AspNetCore.Identity.EntityFrameworkCore (6.0.16)** [15]

Эта библиотека расширяет ASP.NET Core Identity для работы с базой данных, основанной на Entity Framework Core. Она обеспечивает функциональность управления пользователями, ролями и аутентификацией.

**Microsoft.AspNetCore.Mvc.NewtonsoftJson (6.0.16)** [16]

Эта библиотека обеспечивает интеграцию Newtonsoft.Json с ASP.NET Core MVC, позволяя сериализовать и десериализовать JSON данные в модели MVC контроллеров.

**Microsoft.EntityFrameworkCore (7.0.5)** [17]

Эта библиотека предоставляет функциональность Entity Framework Core для работы с базами данных. Она упрощает доступ к данным и выполнение операций CRUD (создание, чтение, обновление, удаление) с помощью ORM (Object-Relational Mapping).

**Microsoft.EntityFrameworkCore.Design (7.0.5)** [18]

Эта библиотека предоставляет инструменты дизайна для Entity Framework Core, включая возможность создания и миграции базы данных.

**Microsoft.EntityFrameworkCore.Proxies (7.0.5)** [19]

Эта библиотека позволяет использовать прокси-классы для отложенной загрузки связанных данных в Entity Framework Core.

**Microsoft.EntityFrameworkCore.Tools (7.0.5)** [20]

Эта библиотека содержит инструменты командной строки для работы с Entity Framework Core, включая создание и применение миграций базы данных.

**Npgsql.EntityFrameworkCore.PostgreSQL (7.0.4)** [21]

Эта библиотека предоставляет поддержку PostgreSQL базы данных в Entity Framework Core.

**Swashbuckle.AspNetCore (6.5.0)** [22]

Эта библиотека предоставляет инструменты для создания документации и интерактивного интерфейса для ASP.NET Core Web API на основе спецификации OpenAPI (ранее известной как Swagger).

**Newtonsoft.Json (13.0.3)** [23]

Это популярная библиотека для работы с JSON в .NET. Она предоставляет функциональность сериализации и десериализации объектов в формат JSON.

Мною был реализованный весь требуемый функционал в полном объёме, но особенно хотелось бы обратить внимание на два варианта использования: получение тикеров из списка и аутентификация пользователей.

### 4.2. Реализация получения данных о тикерах с криптобиржи

Одной из особенностей разрабатываемого Web API является то, что фронтэнд может получить информацию о криптовалютах с разных криптобирж. Но логика работы их унифицирована, следовательно, описав реализацию лишь одной, можно понять, как работают остальные.

Диаграмма последовательности является наглядным представлением некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента. С помощью этой диаграммы я хочу продемонстрировать процесс получения всех тикеров с криптобиржи Binance, что показано на рисунке 12.

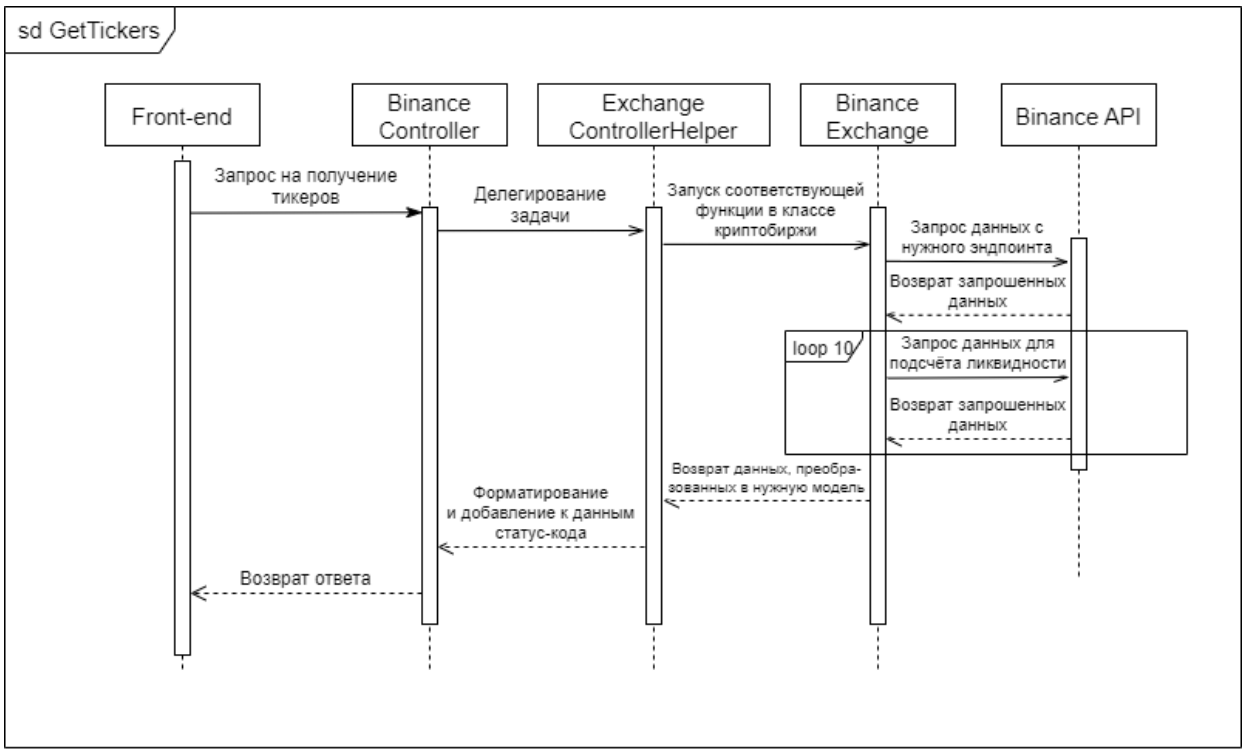


Рисунок 12 – Диаграмма последовательности, описывающая процесс получения тикеров с криптобиржи

Рассмотрим диаграмму последовательности подробнее.

Данные, полученные с Binance API, преобразуются к модели Product. В листинге 1 представлена эта модель, к которой в конце концов происходит преобразование.

Листинг 1 – Модель Product

```
public class Product
{
        public int Id { get; set; }
        public int Exchange { get; set; }
        public string Symbol { get; set; } = null!;
        public decimal? LastPrice { get; set; }
        public decimal? BaseVolume { get; set; }
        public decimal? QuoteVolume { get; set; }
        public double? Volatility { get; set; }
        public decimal? Liquidity { get; set; }
        public decimal? PriceChange { get; set; }
        public decimal? PriceChangePercent { get; set; }
}
```

Начинается обработка запроса с метода GetTickers контроллера BinanceController, его можно увидеть в листинге 2.

Листинг 2 – Метод контроллера BinanceController

```
[HttpGet]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status200OK)]
public async Task<ActionResult<IEnumerable<Product>>> GetTickers()
{
        return await _helper.GetTickersAsync(_exchange.GetTickersAsync);
}
```

Далее происходит делегирование задачи методу GetTickersAsync в классе ExchangeControllerHelper (листинг 3).

Листинг 3 – Метод контроллера ExchangeControllerHelper

```
public async Task<ActionResult<IEnumerable<Product>>> GetTick-
ersAsync(Func<Task<IEnumerable<Product>>> getTickersAsync)
{
        Task<IEnumerable<Product>> task = getTickersAsync();
        var result = await task;
        if (task.IsCompletedSuccessfully)
                return Ok(result);
        else
                return BadRequest();
}
```

Основная логика обработки запроса находится в классе BinanceExchange, где запускается соответствующий метод (листинг 4).

Приведенный код представляет собой часть реализации метода GetTickersAsync, который выполняет запрос данных о тикерах для выбранных торговых пар. В данном случае, запрос осуществляется для торговых пар, связанных с долларом.

Полученные данные обрабатываются внутри метода. Создается пустой список products, в который добавляются объекты типа Product. Для каждого элемента результата запроса, вызывается метод ToProduct, который преобразует данные из формата Binance API в объект Product. Затем, данные объекта Product заполняются необходимой информацией, такой как идентификатор, символ, цена, объемы, волатильность и ликвидность.

Метод ToProduct также используется для расчета волатильности и ликвидности. В методе GetVolatility производится расчет волатильности на основе данных о последней цене, средневзвешенной цене и общем количестве сделок. Метод GetLiquidity выполняет запрос данных о стакане ордеров для указанного символа и вычисляет ликвидность. Именно здесь и произойдёт цикл, изображённый на диаграмме, в котором будет произведено 10 запросов на Binance API.

Листинг 4 – Бизнес-логика обработки запроса

```
public override async Task<IEnumerable<Product>> GetTickersAsync()
{
        var result = await client.SpotApi.ExchangeData.GetTick-
ersAsync(pairsUSDT);
        List<Product> products = new List<Product>();

        if (result.Success)
                result.Data.ToList().ForEach(p => products.Add(ToProd-
uct(p)));
        idProduct = 0;
        return products;
}

protected override Product ToProduct(object product, decimal usdCurrency =
1)
{
```

```
        Binance24HPrice binanceProduct = (Binance24HPrice)product;
        Product p = new Product();
        p.Id = ++idProduct;
        p.Symbol = binanceProduct.Symbol;
        p.Exchange = id;
        p.LastPrice = binanceProduct.LastPrice * usdCurrency;
        p.BaseVolume = binanceProduct.Volume;
        p.QuoteVolume = binanceProduct.QuoteVolume;
        p.Volatility = GetVolatility(binanceProduct);
        p.Liquidity = GetLiquidity(binanceProduct.Symbol).Result;
        p.PriceChange = binanceProduct.PriceChange;
        p.PriceChangePercent = binanceProduct.PriceChangePercent;
        return p;
}

private double GetVolatility(Binance24HPrice binanceProduct)
{
        double result = Math.Sqrt(
                (double)(binanceProduct.LastPrice - binance-
Product.WeightedAveragePrice) *
                (double)(binanceProduct.LastPrice - binance-
Product.WeightedAveragePrice) /
                binanceProduct.TotalTrades);
        return Double.IsNormal(result) ? Math.Round(result, 5) : 0;
}
private async Task<decimal> GetLiquidity(string symbol)
{
        var result = await client.SpotApi.ExchangeData.GetOrder-
BookAsync(symbol, 10);
        decimal asks = 0;
        decimal bids = 0;

        if (result.Success)
        {
                if (result.Data.Asks.Count() != 0 && re-
sult.Data.Bids.Count() != 0)
                {
                        bids = result.Data.Bids.Sum(b => b.Price) / re-
sult.Data.Bids.Count() * result.Data.Bids.Sum(b => b.Quantity);
                        asks = result.Data.Asks.Sum(a => a.Price) / re-
sult.Data.Asks.Count() * result.Data.Asks.Sum(a => a.Quantity);
                }
        }

        return asks > 0 ? Math.Round(bids / asks, 5) : 0;
}
```

При полностью успешной работе мы имеем следующий результат, представленный на рисунке 13. Для демонстрации ответа, я использую Swagger. Swagger – это инструмент для создания, документирования и тестирования API.


Рисунок 13 – Ответ на запрос GetTickers

### 4.3. Реализация процесса аутентификации

В силу того, что в данном сервисе реализована система пользователей, то интересно было бы взглянуть как она работает хотя бы на диаграмме. Для такого случая я выбрал диаграмму деятельности, которую можно увидеть на рисунке 14.

Мною реализована аутентификация пользователей с использованием JWT Bearer-аутентификации. Для обеспечения безопасности и контроля доступа к различным ресурсам приложения, используется механизм аутентификации на основе токенов JWT (JSON Web Tokens).

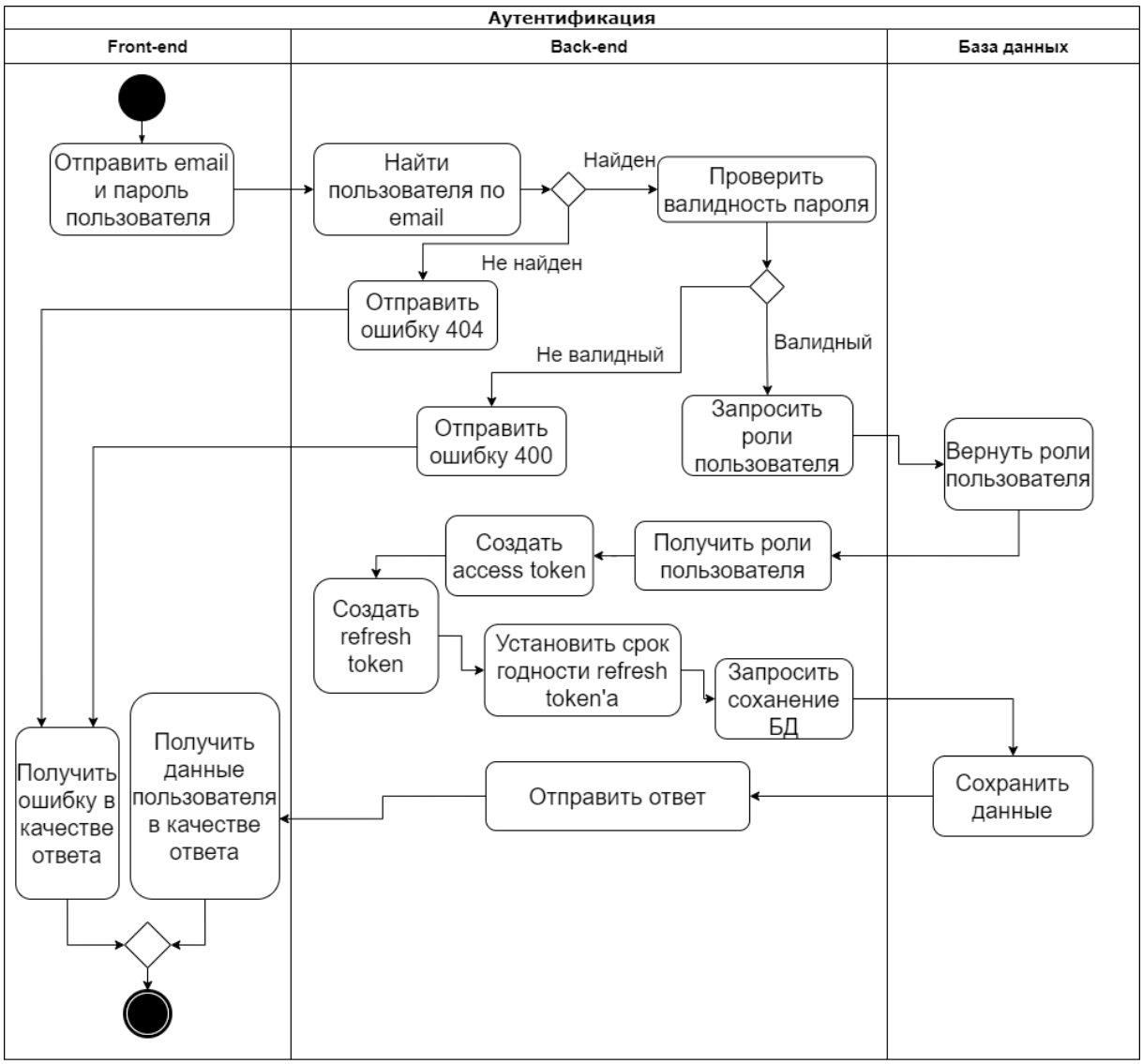


Рисунок 14 – Диаграмма деятельности, описывающая процесс аутентификации

Рассмотрим диаграмму деятельности подробнее.

Первоначально, фротэнд должен отослать запрос, в теле которого содержится следующая модель (листинг 5).

Листинг 5 – Модель запроса на аутентификацию

```
public class AuthRequest
{
        public string Email { get; set; } = null!;
        public string Password { get; set; } = null!;
}
```

Далее начинает свою работу метод Authenticate контроллера AccountsController, который отвечает за проверку учетных данных пользователя и выдачу JWT-токена для авторизации. При получении запроса с данными аутентификации, метод ищет пользователя по указанному e-mail. Если пользователь найден, происходит проверка пароля, иначе отсылается сообщение об отсутствии пользователя с указанным e-mail со статус-кодом 404. В случае несовпадения пароля, отсылается сообщение об этом со статус-кодом 400. После получения какой-либо из ошибок прецедент завершается. В ином случае бэкэнд запрашивает роли пользователя, они нужны для генерации access токена. База данных отсылает ответ с нужными ролями, а бэкэнд их получает. Далее создаётся access токен и refresh токен, устанавливается срок годности refresh токена и отправляется запрос на сохранение этих данных в БД. В результате этой работы бэкэнд отсылает возвращает объект AuthResponse, содержащий информацию о пользователе, выданный access токен и refresh токен. Фронтэнд получает эти данные.

Для создания JWT-токена и обработки его параметров используется TokenService, который выполняет необходимые операции кодирования, декодирования и подписи токена.

Эта реализация аутентификации на основе JWT Bearer обеспечивает безопасность и контроль доступа к ресурсам приложения, а также предоставляет возможность автоматического продления срока действия токена с помощью refresh токена.

Код описанного метода представлен в листинге 6.

Листинг 6 – Метод Authenticate

```
[HttpPost("login")]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
public async Task<ActionResult<AuthResponse>> Authenticate([FromBody]
AuthRequest authRequest)
{
        User? managedUser = await _userManager.FindByEmailAsync(authRe-
quest.Email);
        if (managedUser == null) return NotFound("Invalid e-mail");

        if (!await _userManager.CheckPasswordAsync(managedUser, authRe-
quest.Password)) return BadRequest("Invalid password");

        User? user = _userContext.Users.FirstOrDefault(user => user.Email
== authRequest.Email);
        if (user == null) return Unauthorized();

        List<long> roleIds = await _userContext.UserRoles.Where(role =>
role.UserId == user.Id).Select(role => role.RoleId).ToListAsync();
        var roles = await _userContext.Roles.Where(role => roleIds.Con-
tains(role.Id)).ToListAsync();

        string accessToken = _tokenService.CreateAccessToken(user, roles);
        user.RefreshToken = _tokenService.CreateRefreshToken();
        user.RefreshTokenExpiryTime = DateTime.UtcNow.AddDays(_configura-
tion.GetSection("Jwt:RefreshTokenValidityInDays").Get<int>());

        await _userContext.SaveChangesAsync();

        return Ok(new AuthResponse
        {
                Username = user.UserName!,
                Email = user.Email!,
                Token = accessToken,
                RefreshToken = user.RefreshToken
        });
}
```

Как было упомянуто ранее, ответ содержит объект AuthResponse, модель которого можно просмотреть в листинге 7.

Листинг 7 – Модель AuthResponse

```
public class AuthResponse
{
        public string Username { get; set; } = null!;
        public string Email { get; set; } = null!;
        public string Token { get; set; } = null!;
        public string RefreshToken { get; set; } = null!;
}
```

Таким образом могут выглядеть входные данные (рисунок 15).

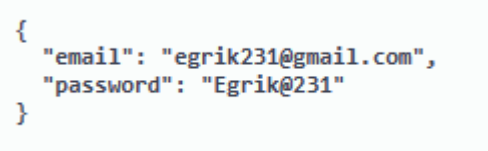


Рисунок 15 – Пример входных данных для аутентификации

Таким образом может выглядеть ответ (рисунок 16).


Рисунок 16 – Ответ на запрос аутентификации

### 4.4. Вывод по четвёртой главе

В данной главе были рассмотрены два аспекта реализации: получение данных с Binance API и аутентификация с использованием JWT Bearer. Оба этих аспекта реализации имеют важное значение для разработки веб-приложений. Получение данных с внешних API, таких как Binance, позволяет использовать актуальную информацию о рыночных тенденциях и активах. Аутентификация с использованием JWT Bearer обеспечивает безопасность и контроль доступа к различным ресурсам приложения.

## 5. ТЕСТИРОВАНИЕ

В силу особенностей системы, а именно отсутствие UI-интерфейса, я не могу провести юзабилити тестирование, поэтому мною было выбрано функциональное тестирование. Оно призвано для того, чтобы проверить корректность работы бизнес-логики приложения, что позволяет проанализировать правильно ли работает приложение, соответствует ли оно заявленным функциональным требованиям.

Результаты этого тестирования можно увидеть в таблице 3.

Таблица 3 – Результаты функционального тестирования

| № | Название теста | Шаги | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 1 | Запуск фоновых сервисов | Запустить приложение | Выведены сообщения о запуске фоновых сервисов в консоли | Да |
| 2 | Регистрация | Отправить объект на основе модели RegisterRequest на нужный эндпоинт | Получен объекта на основе модели AuthResponse | Да |
| 3 | Восстановить access и refresh токены | Отправить объект на основе модели TokenModel на нужный эндпоинт | Получен ответ, содержащий новые access и refresh токены | |
| 4 | Запрос данных о пользователе | Аутентифицироваться, отправить username на нужный эндпоинт | Получен объекта на основе модели User | Да |
| 5 | Отправка фото профиля пользователя | Аутентифицироваться, отправить фото на нужный эндпоинт с указанием username | Получен ответ со статус-кодом 200 | Да |

Окончание таблицы 3

| № | Название теста | Шаги | Ожидаемый результат | Тест пройден? |
|---|---|---|---|---|
| 6 | Запрос данных о тикерах | Запросить данные с нужного эндпоинта | Получен список объектов на основе модели Product | Да |
| 7 | Запрос данных об одном тикере | Запросить данные с нужного эндпоинта | Получен объект на основе модели Product | Да |
| 8 | Получение данных об уроке | Аутентифицироваться, запросить данные с нужного эндпоинта с указанием lessonId | Получен объект на основе модели LessonResponse | Да |
| 9 | Получение теста | Аутентифицироваться, запросить данные с нужного эндпоинта с указанием username и lessonId | Получен список объектов на основе модели Question | Да |
| 10 | Сохранить пройденный тест | Аутентифицироваться, отправить список объектов на основе модели AnswerRequest на нужный эндпоинт с указанием username и lessonId | Получен ответ со статус-кодом 201, complitedTestId и объектом на основе модели ComplitesdTest | Да |

**Вывод к пятой главе**

По результатам проведенного тестирования приложения можно сделать вывод, что все основные функции работают безупречно и не было обнаружено никаких ошибок. Можно с уверенностью сказать, что приложение удовлетворяет всем заявленным ранее требованиям.

## ЗАКЛЮЧЕНИЕ

В рамках данной работы был разработан бэкэнд информационного сервиса по криптовалютам, позволяющий получать актуальные котировки валют, информацию, помогающую оценить ситуацию на данном рынке, предоставляющий возможность получить учебно-методические материалы, а также тесты для закрепления знаний, полученных в течение их изучения.

В результате работы были достигнуты следующие цели:

1. Изучена предметная область;
2. На основе диаграммы вариантов использования сформулированы требования к приложению;
3. Спроектирована архитектура приложения с помощью диаграмм компонентов, диаграммы последовательности и диаграммы деятельности;
4. Разработано Web API, с использованием платформы ASP.NET Core;
5. Проведено тестирование приложения с целью подтверждения его работоспособности.

В заключении хочется сказать, что в будущей новой версии данного приложения будет добавлено множество новых функций, в первую очередь будет расширен функционал, связанный с личным кабинетом пользователя, а также планируется расширение направленности сервиса.

**ЛИТЕРАТУРА**

1. White Bulls Crypto Club. Лучшие криптовалюты 2021 года [Электронный ресурс] URL: https://wbcc-club.com/top-10-kriptovalyut-2021/ (дата обращения 04.03.2023);

2. Media Sigen.Pro. Гайд по самым полезным сервисам на крипторынке. [Электронный ресурс] URL: https://media.sigen.pro/guides/7021 (дата обращения 04.03.2023);

3. Крипто-курс. CoinDesk: Топ-10 криптотрейдеров и аналитиков прошлого года. [Электронный ресурс] URL: https://endnomer.ru/vidy-kriptovalyuty/new-altcoins.html (дата обращения 04.03.2023);

4. ProfInvestment. Пошаговая инструкция по работе с криптовалютой для новичков, гайд с примерами и пояснениями. [Электронный ресурс] URL: https://profinvestment.com/cryptocurrency-guide/ (дата обращения 04.03.2023);

5. Хабр. Самые популярные языки программирования бэкенда: для чего они подходят лучше всего и какие компании их используют. [Электронный ресурс] URL: https://habr.com/ru/company/skillbox/blog/534684/ (дата обращения 04.03.2023);

6. Microsoft. A framework for building web apps and services with .NET and C#. [Электронный ресурс] URL: https://dotnet.microsoft.com/en-us/apps/aspnet (дата обращения 04.03.2023);

7. Нелюбов Р. П. Безопасность в веб-приложениях ASP.NET Core. E-Scio. 2022. № 5 (68). С. 131-139;

8. Proglib. ТОП-10 систем управления базами данных в 2019 году. [Электронный ресурс] URL: https://proglib.io/p/databases-2019 (дата обращения 04.03.2023);

9. Overview of ASP.NET Core MVC. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-7.0 (дата обращения 04.03.2023);

10. Dependency injection in ASP.NET Core [Электронный ресурс] URL: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-7.0 (дата обращения 04.03.2023);

11. Binance.NET [Электронный ресурс] URL: https://jkorf.github.io/Binance.Net/ (дата обращения 04.03.2023);

12. Kucoin.NET [Электронный ресурс] URL: https://jkorf.github.io/ Kucoin.Net/ (дата обращения 04.03.2023);

13. Huobi.NET [Электронный ресурс] URL: https://jkorf.github.io/ Huobi.Net/ (дата обращения 04.03.2023);

14. Microsoft.AspNetCore.Authentication.JwtBearer Namespace [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.authentication.jwtbearer?view=aspnetcore-7.0 (дата обращения 04.03.2023);

15. Microsoft.AspNetCore.Identity.EntityFrameworkCore Namespace [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.entityframeworkcore?view=aspnetcore-7.0 (дата обращения 04.03.2023);

16. Microsoft.AspNetCore.Mvc.NewtonsoftJson Namespace [Электронный ресурс] URL: https://learn.microsoft.com/de-de/dotnet/api/microsoft.aspnetcore.mvc.newtonsoftjson?view=aspnetcore-7.0 (дата обращения 04.03.2023);

17. Microsoft.EntityFrameworkCore Namespace [Электронный ресурс] URL: https://learn.microsoft.com/en-us/ef/core/what-is-new/ef-core-5.0/whatsnew (дата обращения 04.03.2023);

18. Microsoft.EntityFrameworkCore.Design Namespace [Электронный ресурс] URL: https://learn.microsoft.com/en-us/ef/core/cli/services (дата обращения 04.03.2023);

19. Microsoft.EntityFrameworkCore.Proxies Namespace [Электронный ресурс] URL: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Proxies/3.1.19 (дата обращения 04.03.2023);

20. Microsoft.EntityFrameworkCore.Tools Namespace [Электронный ресурс] URL: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Tools/3.1.20 (дата обращения 04.03.2023);

21. Npgsql Entity Framework Core Provider [Электронный ресурс] URL: https://www.npgsql.org/efcore/ (дата обращения 04.03.2023);

22. Swashbuckle.AspNetCore [Электронный ресурс] URL: https://github.com/domaindrivendev/Swashbuckle.AspNetCore (дата обращения 04.03.2023);

23. Newtonsoft.Json [Электронный ресурс] URL: https://www.newtonsoft.com/json (дата обращения 04.03.2023);

## ПРИЛОЖЕНИЯ

### Приложение 1. Спецификация вариантов использования

Спецификация вариантов использования (ВИ) системы приведена в таблицах 4–17.

Таблица 4 – Спецификация ВИ «Получить тикеры из списка»

| |
|---|
| Прецедент «Получить тикеры из списка» |
| ID: 1 |
| Краткое описание:<br>Получает тикеры из списка |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Система подключена к интернету |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на получение всех тикеров из списка;<br>2. Система отправляет запрос на API криптобиржи;<br>3. Если front-end указывает, что нужно получить стоимость в рублях;<br>3.1. Система преобразует цену в соответствии с актуальным курсом валют;<br>4. Система отправляет ответ с данными. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 5 – Спецификация ВИ «Получить один тикер»

| Прецедент «Получить один тикер» |
| --- |
| ID: 2 |
| Краткое описание:<br>Получает один тикер |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Система подключена к интернету |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на получение одного тикера;<br>2. Система отправляет запрос на API криптобиржи;<br>3. Если front-end указывает, что нужно получить стоимость в рублях;<br>3.1. Система преобразует цену в соответствии с актуальным курсом валют;<br>4. Система отправляет ответ с данными. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 6 – Спецификация ВИ «Получить свечи одной пары»

| |
|---|
| Прецедент «Получить свечи одной пары» |
| ID: 3 |
| Краткое описание:<br>Получение японских свечей по одной паре за указанный промежуток времени |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Система подключена к интернету |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на получение свечей;<br>2. Система отправляет запрос на API криптобиржи;<br>3. Система отправляет ответ с данными. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 7 – Спецификация ВИ «Зарегистрировать пользователя»

| Прецедент «Зарегистрировать пользователя» |
|---|
| ID: 4 |
| Краткое описание:<br>Регистрирует нового пользователя |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Нет |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на регистрацию нового пользователя с указанием данных;<br>2. Система регистрирует нового пользователя;<br>3. Система аутентифицирует пользователя;<br>4. Система отправляет ответ с указанием информации о пользователе. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 8 – Спецификация ВИ «Аутентифицировать пользователя»

| Прецедент «Аутентифицировать пользователя» |
| --- |
| ID: 5 |
| Краткое описание:<br>Аутентифицирует пользователя |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Пользователь зарегистрирован |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на аутентификацию с указанием данных;<br>2. Система аутентифицирует пользователя;<br>3. Система отправляет ответ с указанием информации о пользователе. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 9 – Спецификация ВИ «Отозвать refresh токен одного пользователя»

| |
|---|
| Прецедент «Отозвать refresh токен одного пользователя» |
| ID: 6 |
| Краткое описание:<br>Отзывает refresh токен одного пользователя |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Пользователь аутентифицирован |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на отзыв refresh токена;<br>2. Система отзывает refresh токен;<br>3. Система отправляет ответ об успехе. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 10 – Спецификация ВИ «Отозвать refresh токен всех пользователей»

| |
|---|
| Прецедент «Отозвать refresh токен всех пользователей» |
| ID: 7 |
| Краткое описание:<br>Отзывает refresh токен всех пользователей |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Пользователь, отправляющий запрос, авторизирован как администратор |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на отзыв refresh токена всех пользователей;<br>2. Система отзывает refresh токены;<br>3. Система отправляет ответ об успехе. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 11 – Спецификация ВИ «Восстановить access & refresh токены»

| Прецедент «Восстановить access & refresh токены» |
|---|
| ID: 8 |
| Краткое описание:<br>Восстанавливает access & refresh токены |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Нет |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на восстановление;<br>2. Система восстанавливает токены;<br>3. Система отправляет ответ с указанием новых токенов. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 12 – Спецификация ВИ «Получить информацию о пользователе»

| |
|---|
| Прецедент «Получить информацию о пользователе» |
| ID: 9 |
| Краткое описание:<br>Получение информации о пользователе |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Пользователь аутентифицирован |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на получение информации;<br>2. Система собирает URL фото профиля пользователя;<br>3. Система собирает результаты всех тестов, пройденных пользователем;<br>4. Система отправляет ответ с указанием информации. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 13 – Спецификация ВИ «Редактировать информацию о пользователе»

| |
|---|
| Прецедент «Редактировать информацию о пользователе» |
| ID: 10 |
| Краткое описание:<br>Редактирование информации о пользователе |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Пользователь аутентифицирован |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет новое описание профиля;<br>2. Система сохраняет новое описание пользователя;<br>3. Система отправляет ответ об успехе. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 14 – Спецификация ВИ «Получить все уроки

| |
|---|
| Прецедент «Получить все уроки» |
| ID: 11 |
| Краткое описание:<br>Получение короткой информации о всех уроках |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Нет |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на получение информации;<br>2. Система собирает URL фото, связанных с уроками;<br>3. Система собирает описания уроков;<br>4. Система отправляет ответ с указанием информации. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 15 – Спецификация ВИ «Получить один урок»

| |
|---|
| Прецедент «Получить один урок» |
| ID: 12 |
| Краткое описание:<br>Получение полной информации об уроке |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Пользователь аутентифицирован |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на получение информации;<br>2. Система собирает URL фото, связанных с уроком;<br>3. Система собирает учебно-методические материалы урока;<br>4. Система отправляет ответ с указанием информации. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 16 – Спецификация ВИ «Получить тест»

| |
|---|
| Прецедент «Получить тест» |
| ID: 13 |
| Краткое описание:<br>Получение вопросов по тесту |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Пользователь аутентифицирован |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет запрос на получение информации;<br>2. Система собирает вопросы;<br>3. Система отправляет ответ с указанием информации. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |

Таблица 17 – Спецификация ВИ «Сохранить результаты теста»

| |
|---|
| Прецедент «Сохранить результаты теста» |
| ID: 14 |
| Краткое описание:<br>Сохранение результатов теста |
| Главные актёры:<br>Front-end |
| Второстепенные актёры:<br>Нет |
| Предусловия:<br>Пользователь аутентифицирован |
| Основной поток:<br>1. Вариант использования начинается, когда front-end отправляет результаты теста;<br>2. Система сохраняет результаты;<br>3. Система отправляет ответ со статус-кодом 201, id завершённого теста и объектом этого теста. |
| Постусловия:<br>Нет |
| Альтернативные потоки:<br>Нет |