

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Специальность 1-40 04 01 «Информатика и технологии программирования»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
к курсовому проекту  
по дисциплине «Избранные главы информатики»

на тему: **«Web-приложение для автоматизации службы доставки пиццы»**

Исполнитель: студент гр. ИП-32  
Э. С. Суховенко

Руководитель: преподаватель  
М. А. Процкая

Дата проверки: \_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_

Дата защиты: \_\_\_\_\_

Оценка проекта: \_\_\_\_\_

Подписи членов комиссии

по защите курсового проекта: \_\_\_\_\_

Гомель 2022

## СОДЕРЖАНИЕ

Введение.....	3
1 Аналитический обзор.....	4
1.1 Анализ предметной области.....	4
1.2 Обзор существующих аналогов.....	4
1.3 Обзор технологий.....	5
1.4 Постановка задачи .....	7
2 Проектирование программного обеспечения.....	8
2.1 Функциональное проектирование .....	8
2.2 Информационное обеспечение .....	12
2.3 Архитектура приложения.....	18
3 Реализация приложения .....	19
3.1 Описание интерфейса пользователя .....	19
3.2 Описание классов.....	28
3.3 Тестирование и верификация .....	32
Заключение .....	36
Список использованных источников .....	37
Приложение А Листинг программного кода.....	38

## **ВВЕДЕНИЕ**

С ростом численности ресторанов стремительно усиливается и конкуренция, что неизбежно приводит к необходимости эффективно и рационально использовать имеющиеся ресурсы. В этих условиях для успешного ведения своего дела и поддержания требуемого уровня конкурентоспособности необходимо производить инвестиции в инструменты поддержания и развития бизнеса. Один из основных инструментов развития ресторанного бизнеса - это современная система автоматизации ресторанов.

Данный проект направлен на то, чтобы изучить бизнес-процессы, происходящие внутри пиццерии, выявить, так называемые, «узкие» места в структуре построения, функционирования предприятия и указать на них. Предметом детального анализа была выбрана деятельность по обслуживанию клиентов, так как работа этого подразделения является основой деятельности ресторана.

Целью данного курсового проекта является проектировка автоматизированной системы службы доставки пиццы, которая позволит пользователям пройти полный путь от создания заказа, до отзыва после его выполнения, сотрудникам легко обрабатывать заказы от пользователей и позволит организовать упорядочивание списков в базах данных, для составления отчетности.

# 1 АНАЛИТИЧЕСКИЙ ОБЗОР

## 1.1 Анализ предметной области

В настоящее время внедрение технологий автоматизации является повсеместным и уже закрепилось как важная часть жизни людей. Обычный человек не должен думать о том, как он покупает билеты на поезд или же, как он делает заказ доставки еды, ему главное, чтобы все эти процессы были интуитивно понятны и просты.

Из-за растущего населения и возрастающих нужд планеты рестораны еще не скоро забудут, необходимость что-либо кушать появляется всегда, и чтобы избежать временные затраты и некачественные продукты, этим должны заниматься профессионалы. Так и с пищей, такой ответственной и сложной работой должны заниматься профессионалы. Соответственно, при повышении нужд растет и количество предоставляемых услуг. Так мы уже столкнулись с двумя проблемами, первая – проблема доступа к профессионалу и вторая – сложность в хранении всей информации о услугах, заказах и прочее. Данные проблемы может решить служба автоматизации службы доставки пиццы.

Система должна вести учет видов работ, учет прејскуранта, уметь автоматически определять свободного курьера на выбранную дату. Клиенту необходима возможность оставить заказ, в котором он будет заполнять всю необходимую информацию и, в случае успешного завершения оставлять отзыв. Так же немаловажным будет ведение статистики для администратора приложения.

Итогом проектирования станет веб-приложение с подключенной локальной базой данных, в которой будет храниться информация о пользователях, их ролях, курьерах, заказах, услугах и отчетах на основе всей информации из базы данных.

## 1.2 Обзор существующих аналогов

Существует несколько программ, которые позволяют управлять заказами, но они либо являются дорогостоящими, либо узкоспециализированными на определённый вид услуг, таких как доставка пиццы.

Примером такой системы является программа «*FastOperator*». «*FastOperator*» - эффективное решение для автоматизации полного цикла работы службы доставки. Программа оптимально подходит предприятиям общественного питания, осуществляющим прием заказов и доставку готовой продукции на дом или в офис. Главным минусом данной программы является её цена, которая составляет 250 у.е.

Так же есть информационная система «Диспетчер Доставки» предназначенный для автоматизации работы служб доставки: пиццерий, суши-баров, кафе, ресторанов, фаст-фуд сервиса.

Следующий ресурс, располагающий подобным функционалом – *kabanchik.by*. Кабанчик – это сервис поиска частных специалистов для решения бытовых, а также решения бизнес-задач. Данный сервис предоставляет возможность клиенту сделать заказ на услуги вида: домашний мастер, отделочные работы, клининговые услуги, курьерские услуги, строительные работы, ремонт техники, логистические и складские услуги, бытовые услуги, мебельные работы и прочее. Потенциальный клиент оставляет заявку на нужный ему вид услуг и ждет, пока сервис подберет ему команду или специалиста для ее выполнения. В сервисе существует отчетность для выбора лучшего варианта. Обобщая все выше написанное, можно заявить, что данный сервис располагает полным спектром возможностей необходимых для комфортной работы как клиентам, так и компаниям, предоставляющим услуги.

Последний ресурс, строительная биржа *Remline.by* – это уникальная биржа в сфере строительства, которая позволяет создать тендер частному лицу, что позволяет сильно сэкономить денежные средства. *Remline* сервис обладающий всем необходимым функционалом для решения задач курсового проекта, в нем можно создать заказ, сервис его обработает и подберет свободную команду для его выполнения, для услуг существует прейскурант, так же существуют отзывы клиентов и примеры работ различных команд. Сервис проводит автоматизацию работ служб по приготовлению пищи, что позволяет им выполнять свою работу с наибольшей эффективностью.

Проанализировав приведенные ранее сервисы, можно сделать вывод о том, какой основной функционал должен быть у приложения, а именно: оформление, подтверждение и выполнение заказов, хранения базы данных о клиентах, их заказах, различного рода отчеты для администраторов приложений.

### 1.3 Обзор технологий

Для реализации данного приложения необходимо использовать набор технологий, который позволит нам реализовать весь необходимый функционал. В качестве языка написания был выбран язык *Typescript*. В качестве хранилища данных была выбрана *SQLite*. Согласно [1], *SQLiteStudio* – это реляционная система управления базой данных (СУБД). В реляционных базах данных данные хранятся в таблицах.

Для доступа к данным будет использована технология *Sequelize*. Согласно [2], *Sequelize* представляет *ORM*-технологию (*object-relational mapping* – отображения данных на реальные объекты) для доступа к данным. *Sequelize Models* позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными как с объектами класса независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает *Sequelize*, мы уже работаем с объектами.

Платформой для создания *Web*-приложения была выбрана *Node.js*. *Node.js* это кроссплатформенная внутренняя среда выполнения *JavaScript* с открытым исходным кодом, которая работает на движке *V8* и выполняет код *JavaScript* вне веб-браузера. *Node.js* позволяет разработчикам использовать *JavaScript* для написания инструментов командной строки и для создания сценариев на стороне сервера – выполнение сценариев на стороне сервера для создания динамического содержимого веб – страницы перед отправкой страницы в веб-браузер пользователя. Следовательно, *Node.js* представляет собой парадигму «*JavaScript* везде», объединяющую разработку веб-приложений вокруг единого языка программирования, а не разных языков для серверных и клиентских скриптов. *Node.js* имеет управляемую событиями архитектуру, способную выполнять асинхронный ввод-вывод. Эти варианты дизайна направлены на оптимизацию пропускной способности и масштабируемости в веб-приложениях с большим количеством операций ввода/вывода, а также для веб-приложений реального времени (например, коммуникационных программ в реальном времени и браузерных игр).

Пользовательский интерфейс был реализован с использованием библиотеки *React* также на платформе *Node.js*. *React* (также известный как *React.js* или *ReactJS* - это бесплатная интерфейсная библиотека *JavaScript* с открытым исходным кодом для создания пользовательских интерфейсов на основе компонентов пользовательского интерфейса. Он поддерживается *Meta* (ранее *Facebook*) и сообществом отдельных разработчиков и компаний. *React* можно использовать в качестве основы при разработке одностраничных, мобильных или серверных приложений с такими фреймворками, как *Next.js*. Однако *React* занимается только управлением состоянием и отображением этого состояния в DOM, поэтому для создания приложений *React* обычно требуется использование дополнительных библиотек для маршрутизации, а также определенных функций на стороне клиента.

В качестве архитектурного паттерна был выбран *MVC*. Согласно [4], концепция паттерна *MVC* предполагает разделение приложения на три компонента:

- модель: описывает используемые в приложении данные, а также логику, которая связана непосредственно с данными, например, логику валидации данных. Как правило, объекты моделей хранятся в базе данных. В *MVC* модели представлены двумя основными типами: модели представлений, которые используются представлениями для отображения и передачи данных, и модели домена, которые описывают логику управления данными. Модель может содержать данные, хранить логику управления этими данными. В то же время модель не должна содержать логику взаимодействия с пользователем и не должна определять механизм обработки запроса. Кроме того, модель не должна содержать логику отображения данных в представлении;

- представление: отвечают за визуальную часть или пользовательский интерфейс, нередко *html*-страница, через который пользователь взаимодей-

ствуется с приложением. Также представление может содержать логику, связанную с отображением данных. В то же время представление не должно содержать логику обработки запроса пользователя или управления данными;

– контроллер: представляет центральный компонент *MVC*, который обеспечивает связь между пользователем и приложением, представлением и хранилищем данных. Он содержит логику обработки запроса пользователя. Контроллер получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления, наполненного данными моделей;

Схема паттерна *MVC* представлена на рисунке 1.1.

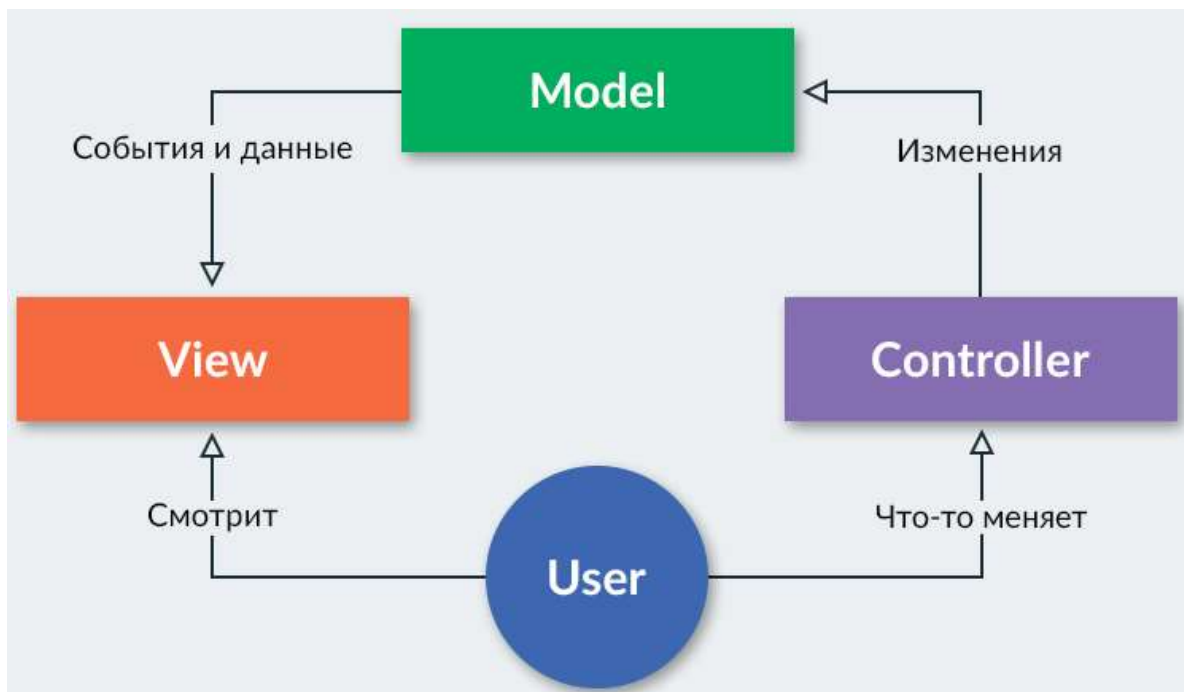


Рисунок 1.1 – Схема паттерна *MVC*

#### 1.4 Постановка задачи

Как упоминалось ранее, проектируемая система из нескольких частей: система составление заказов и система составление отчетности. Пусть, при заполнении заказа, данные для заполнения выбираются напрямую из базы данных и после заполнения данные попадают в базу данных. Данные для составления статистики выбираются из базы данных используя необходимый фильтр.

Для данной курсовой работы можно выделить несколько задач:

- проектировка и разработка реляционной базы данных;
- проектировка и разработка пользовательских функций;
- проектировка и разработка пользовательского интерфейса;
- разработка *Unit*-тестов.

## 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### 2.1 Функциональное проектирование

Предметная область должна описывать все объекты, которые являются частью предметной области, а также любые взаимоотношения между ними. Следовательно, необходимо описать все зависимости между сущностями. Важно знать какие именно объекты попадают в предметную область и какие связи между ними существуют и необходимо помнить цель проектирования данного приложения.

Для формирования представления о предметной области используют *UML* диаграммы, которые представлены на рисунках 2.1 – 2.4.

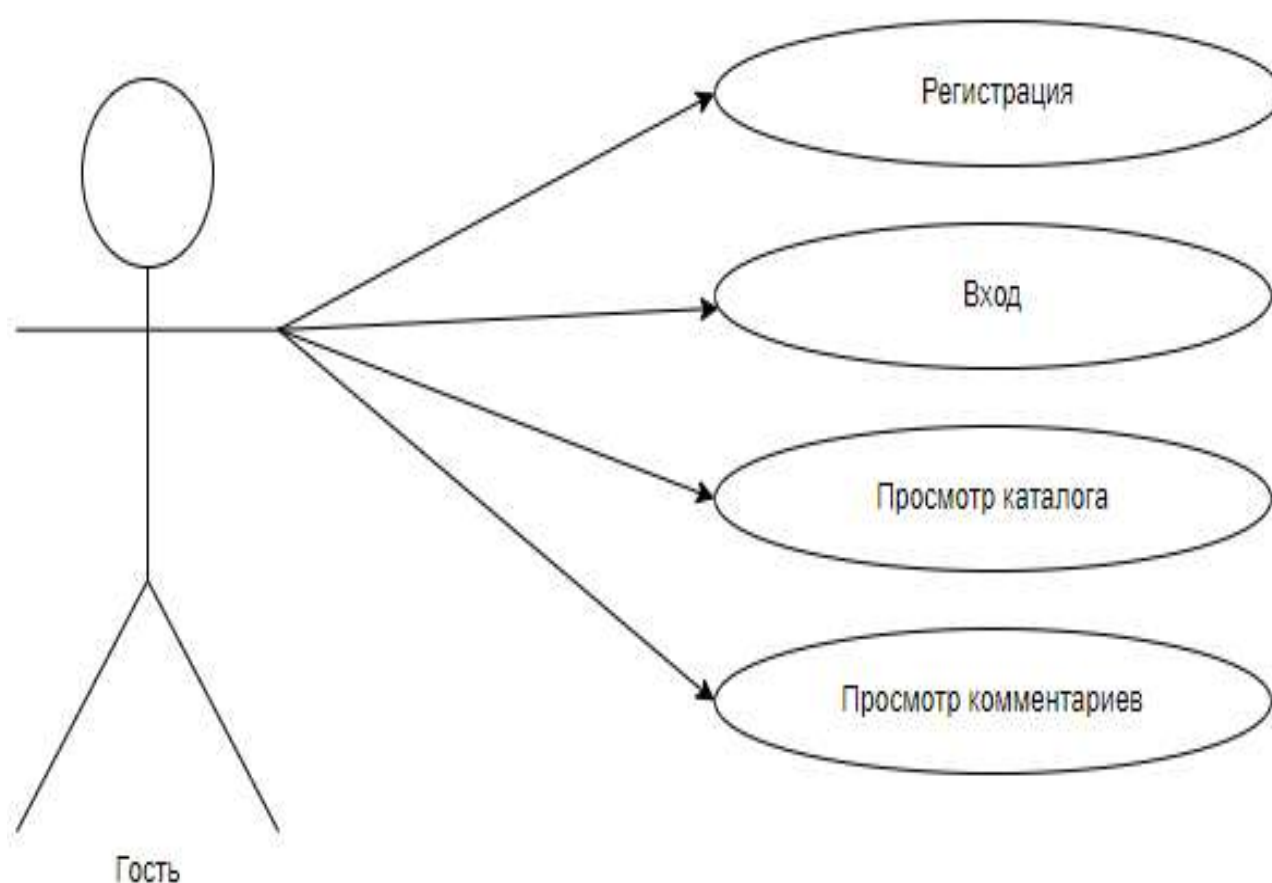


Рисунок 2.1 – Прецеденты актера «Гость»



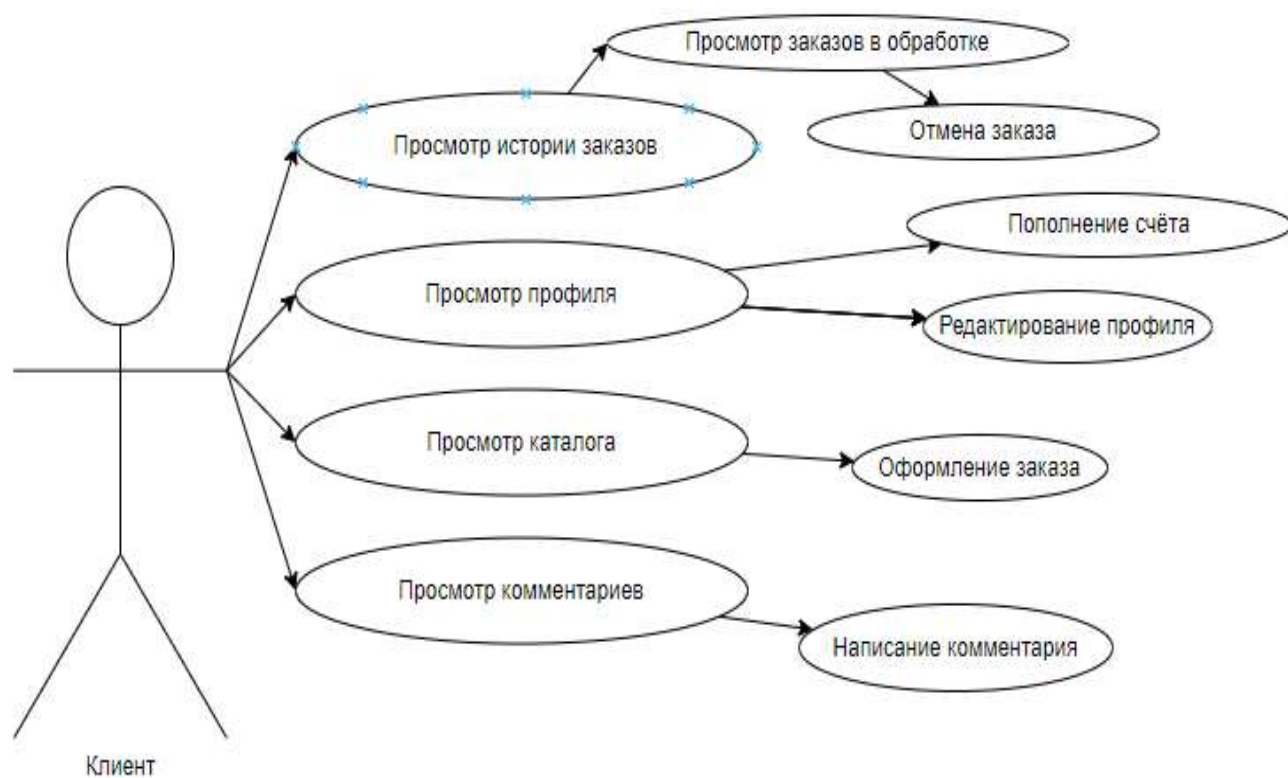


Рисунок 2.2 – Прецеденты актера «Клиент»

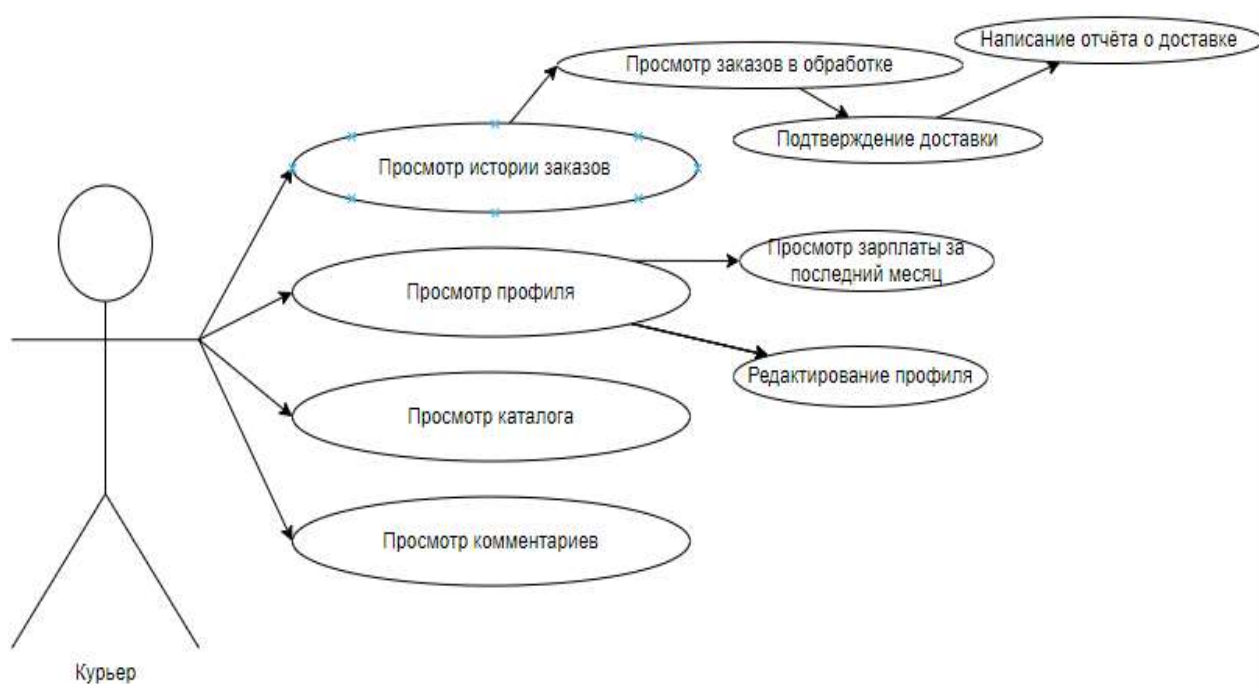


Рисунок 2.3 – Прецеденты актера «Курьер»



Рисунок 2.4 – Прецеденты актера «Менеджер»

В соответствии с рисунками 2.1 – 2.4 можно сделать анализ задачи. Необходимо реализовать 4 роли, менеджер, курьер и клиент будут связаны между собой через заказ. Роль гостя обладает самыми скудными возможностями, а именно просмотром отзывов и прейскуранта.

В соответствии с анализом задачи, можно сделать описание прецедентов и актеров.

Описание актёров:

- актёр «Гость» – актер с минимальными возможностями, может просматривать прейскурант и отзывы;

- актёр «Клиент» – актер с возможностью создавать заказ и с возможностью его оплаты, так же может оставить отзыв на уже выполненный заказ. Как и «гость» имеет возможность просматривать прейскурант;

- актёр «Курьер» – актер, основная функция которого – подтвердить или отклонять поступивший от пользователя заказ, а также писать отчёты по выполненным заказам;

- актер «Менеджер» – актер, контролирующий работу магазина, в его обязанности входит: просмотр и оценка отчётов по заказам с каждого курьера, а также вынесение решений о премировании сотрудников, в результате их хорошей продуктивной работы либо депремировании или вообще увольнении курьеров, в случае нарушения трудового договора.

Описание прецедентов:

- прецедент «регистрация» – регистрация нового клиента, отображается на всех страницах, если пользователь не авторизован, доступен только гостю;

- прецедент «вход» – авторизация пользователя, отображается на всех страницах, если пользователь не авторизован, доступен только гостю;
- прецедент «просмотр каталога» – просмотр всех доступных для заказа пицц, отображается на странице «*Catalog*», доступен всем актерам;
- прецедент «оформление заказа» – оформление заказа на выбранную пиццу с выбором способа оплаты и доставки, отображается на странице «*Catalog*», доступен только клиенту;
- прецедент «просмотр комментариев» – прецедент для просмотра всех комментариев, оставленных пользователями, отображается по умолчанию на главной странице;
- прецедент «написание комментария» – написание комментария после успешного оформления и подтверждения заказа, отображается на главной странице, доступен только клиенту;
- прецедент «просмотр истории заказов» – просмотр истории заказов, не находящихся в обработке, то есть выполненных или отменённых, отображается на странице «*Menu*», доступен клиенту и курьеру;
- прецедент «просмотр заказов в обработке» – просмотр заказов, находящихся в обработке со статусом «*Pending*», отображается на странице «*Menu*», доступен клиенту и курьеру;
- прецедент «отмена заказа» – отмена заказа в случае неверного выбора, ошибки при доставке, либо невозможности принять заказ в назначенное время, находящегося в обработке, отображается на странице «*Menu*», доступен только клиенту;
- прецедент «подтверждение доставки» – подтверждение доставки по заказу, находящемуся в обработке, отображается на странице «*Menu*», доступен только курьеру;
- прецедент «написание отчёта по доставке» – написание отчёта по доставке пиццы, то есть выполнению заказа, находящегося в обработке, отображается на странице «*Menu*», доступен только курьеру;
- прецедент «просмотр профиля» – просмотр профиля, отображается на странице «*Profile*», доступен клиенту, курьеру и менеджеру;
- прецедент «просмотр зарплаты за последний месяц» – просмотр заработной платы работника, отображается на странице «*Profile*», доступен курьеру и менеджеру;
- прецедент «пополнение счёта» – пополнение счёта клиента, для оплаты последующих заказов, отображается на странице «*Profile*», доступен только клиенту;
- прецедент «редактирование профиля» – редактирование данных профиля, таких как полное имя, электронная почта, номер телефона. Отображается на странице «*Profile*», доступен клиенту, курьеру и менеджеру;

– прецедент «просмотр списка всех курьеров» – просмотр списка всех курьеров, а также краткой информации о каждом из них, в том числе заработная плата и данные профиля. Отображается на странице «*Мени*», доступен только менеджеру;

– прецедент «просмотр отчётов» – просмотр отчётов, оформленных каждым курьером, сюда входит просмотр краткой информации о количестве отчётов и суммарной выручке за последний месяц. Отображается на странице «*Мени*», доступен только менеджеру;

– прецедент «просмотр деталей отчёта» – просмотр деталей отчёта, таких как дата написания и описание. Отображается на странице «*Мени*», доступен только менеджеру;

– прецедент «просмотр деталей заказа» – просмотр деталей заказа, а именно информация о заказанной пицце, информация о клиенте, информация о курьере, информация о статусе, нужна в случае отказа, дата оформления заказа и дата его рассмотрения, производится по каждому отчёту каждого курьера. Отображается на странице «*Мени*», доступен только менеджеру;

– прецедент «редактирование зарплаты курьера» – редактирование зарплаты курьера, сюда входит оформление внеочередных премий за переработки или вынесение выговоров в случае прогулов. Отображается на странице «*Мени*», доступен только менеджеру.

## 2.2 Информационное обеспечение

В результате проектирования необходимо получить базу данных, которая сможет обеспечить хранение всех необходимых данных для корректной работы приложения.

Описание таблиц базы данных представлено в таблице 2.1.

Таблица 2.1 – Описание таблиц базы данных

Название таблицы	Назначение
1	2
<i>Accounts</i>	Хранение информации о аккаунтах
<i>Clients</i>	Хранение информации о клиентах
<i>Comments</i>	Хранение информации о комментариях
<i>Couriers</i>	Хранение информации о курьерах
<i>Managers</i>	Хранение информации о менеджерах
<i>Orders</i>	Хранение информации о заказах
<i>Pizzas</i>	Хранение информации о пиццах

Продолжение таблицы 2.1

1	2
<i>Pizzas</i>	Хранение информации о пиццах
<i>Reports</i>	Хранение информации о отчетах по доставке
<i>Statuses</i>	Хранение информации о возможных состояниях заказа

Описание базы данных:

Описание атрибутов таблицы *Accounts* представлено в таблице 2.2.

Таблица 2.2 – Описание атрибутов таблицы «*Accounts*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
<i>id</i>	<i>int</i>	Хранение номера записи в таблице	Да	Да
<i>email</i>	<i>text</i>	Хранение электронной почты	Нет	Нет
<i>passwordHash</i>	<i>text</i>	Хранение зашированного пароля	Нет	Нет

Описание атрибутов таблицы *Clients* представлено в таблице 2.3.

Таблица 2.3 – Описание атрибутов таблицы «*Clients*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
1	2	3	4	5
<i>id</i>	<i>int</i>	Хранение номера записи клиента в таблице « <i>Clients</i> »	Да	Да
<i>accountId</i>	<i>int</i>	Хранение номера аккаунта из таблицы « <i>Accounts</i> »	Нет	Да

Продолжение таблицы 2.3

1	2	3	4	5
<i>name</i>	<i>text</i>	Хранение имени клиента	Нет	Нет
<i>phoneNumber</i>	<i>text</i>	Хранение телефонного номера	Нет	Нет
<i>description</i>	<i>text</i>	Хранение описания	Нет	Нет

Описание атрибутов таблицы *Comments* представлено в таблице 2.4.

Таблица 2.4 – Описание атрибутов таблицы «*Comments*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
<i>id</i>	<i>int</i>	Хранение номера записи в таблице	Да	Да
<i>clientId</i>	<i>int</i>	Хранение номера клиента	Нет	Да
<i>content</i>	<i>text</i>	Хранение текста комментария	Нет	Нет
<i>date</i>	<i>text</i>	Хранение даты	Нет	Нет

Описание атрибутов таблицы *Couriers* представлено в таблице 2.5.

Таблица 2.5 – Описание атрибутов таблицы «*Couriers*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
<i>id</i>	<i>int</i>	Хранение номера записи курьера в таблице	Да	Да
<i>accountId</i>	<i>int</i>	Хранение номера аккаунта	Нет	Да
<i>name</i>	<i>text</i>	Хранение полного имени курьера	Нет	Нет
<i>salary</i>	<i>int</i>	Хранение заработной платы курьера	Нет	Нет
<i>description</i>	<i>text</i>	Хранение описания	Нет	Нет

Описание атрибутов таблицы *Managers* представлено в таблице 2.6.

Таблица 2.6 – Описание атрибутов таблицы «*Managers*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
<i>id</i>	<i>int</i>	Хранение номера записи менеджера в таблице	Да	Да
<i>accountId</i>	<i>int</i>	Хранение номера аккаунта	Нет	Да
<i>name</i>	<i>text</i>	Хранение полного имени менеджера	Нет	Нет
<i>salary</i>	<i>int</i>	Хранение заработной платы менеджера	Нет	Нет
<i>description</i>	<i>text</i>	Хранение описания	Нет	Нет

Описание атрибутов таблицы *Orders* представлено в таблице 2.7

Таблица 2.7 – Описание атрибутов таблицы «*Orders*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
<i>id</i>	<i>int</i>	Хранение номера записи в таблице	Да	Да
<i>pizzaId</i>	<i>int</i>	Хранение номера пиццы	Нет	Да
<i>clientId</i>	<i>int</i>	Хранение номера клиента	Нет	Да
<i>courierId</i>	<i>int</i>	Хранение номера курьера	Нет	Да
<i>statusId</i>	<i>int</i>	Хранение номера статуса заказа	Нет	Да
<i>address</i>	<i>text</i>	Хранение адреса доставки	Нет	Нет
<i>startDate</i>	<i>text</i>	Хранение даты оформления заказа доставки	Нет	Нет
<i>endDate</i>	<i>text</i>	Хранение предполагаемой даты доставки	Нет	Нет

Описание атрибутов таблицы *Pizzas* представлено в таблице 2.8.

Таблица 2.8 – Описание атрибутов таблицы «*Pizzas*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
<i>id</i>	<i>int</i>	Хранение номера записи в таблице	Да	Да
<i>name</i>	<i>text</i>	Хранение названия пиццы	Нет	Нет
<i>description</i>	<i>text</i>	Хранение описания пиццы	Нет	Нет
<i>price</i>	<i>int</i>	Хранение цены пиццы	Нет	Нет
<i>imageUrl</i>	<i>text</i>	Хранение ссылки к фотографии пиццы	Нет	Нет

Описание атрибутов таблицы *Reports* представлено в таблице 2.9.

Таблица 2.9 – Описание атрибутов таблицы «*Reports*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
<i>id</i>	<i>int</i>	Хранение номера записи в таблице	Да	Да
<i>orderId</i>	<i>int</i>	Хранение номера заказа	Нет	Да
<i>date</i>	<i>text</i>	Хранение даты составления отчёта	Нет	Нет
<i>description</i>	<i>text</i>	Хранение описания отчёта	Нет	Нет

Описание атрибутов таблицы *Statuses* представлено в таблице 2.10.

Таблица 2.10 – Описание атрибутов таблицы «*Statuses*»

Имя атрибута	Тип атрибута	Назначение	Первичный ключ	Внешний ключ
<i>id</i>	<i>int</i>	Хранение номера записи	Да	Да
<i>type</i>	<i>text</i>	Хранение типа статуса	Нет	Нет



Схема приведенных выше таблиц представлена на рисунке 2.5.

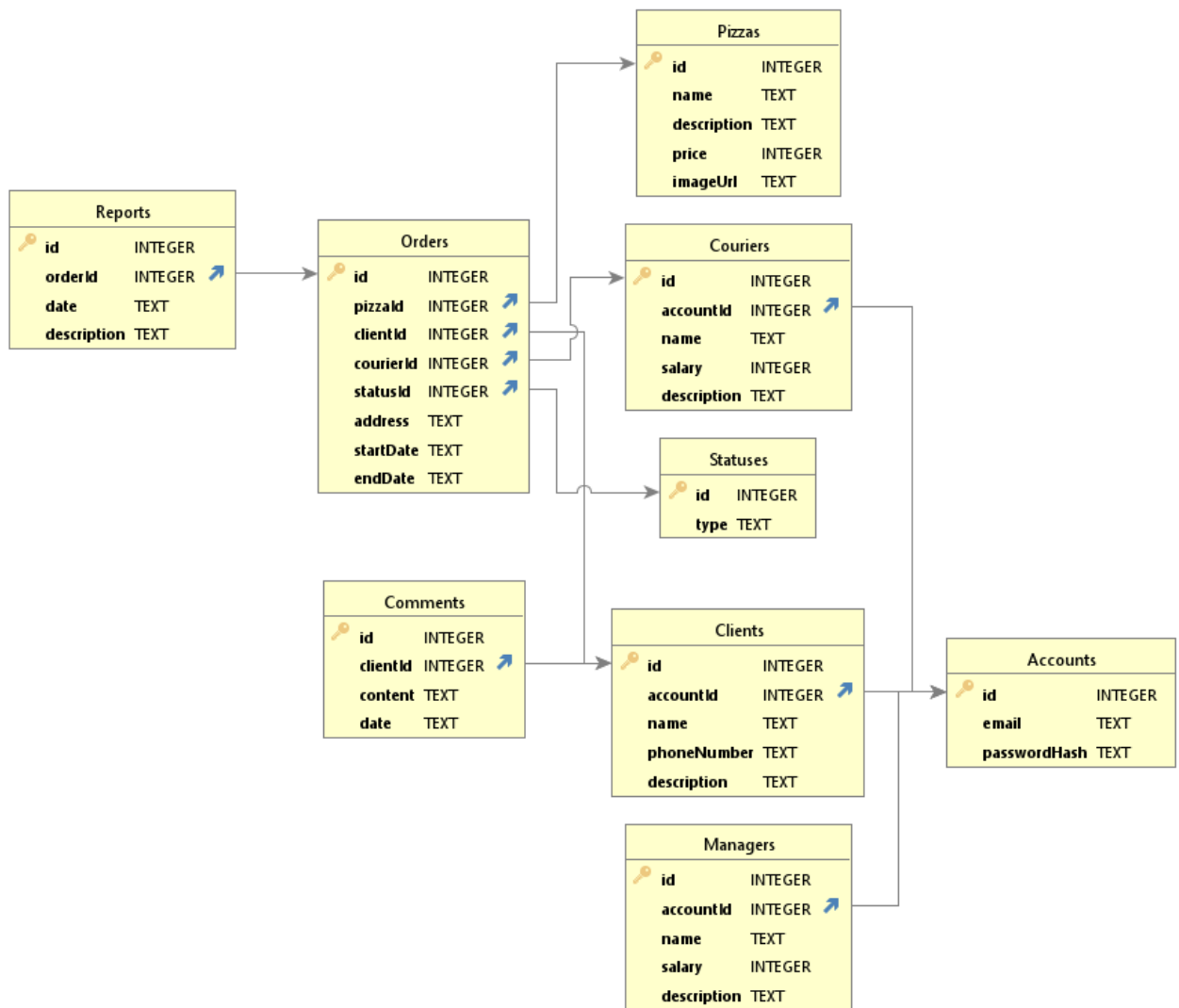


Рисунок 2.5 – Структура базы данных

Описание связей между сущностями:

- связь между таблицами «*Reports*» и «*Orders*» реализуется через внешний ключ «*orderId*». Ограничение целостности – *cascade*;
- связь между таблицами «*Orders*» и «*Pizzas*» реализуется через внешний ключ «*pizzaId*». Ограничение целостности – *cascade*;
- связь между таблицами «*Orders*» и «*Clients*» реализуется через внешний ключ «*clientId*». Ограничение целостности – *cascade*;
- связь между таблицами «*Orders*» и «*Couriers*» реализуется через внешний ключ «*courierId*». Ограничение целостности – *cascade*;
- связь между таблицами «*Orders*» и «*Statuses*» реализуется через внешний ключ «*statusId*». Ограничение целостности – *cascade*;
- связь между таблицами «*Comments*» и «*Clients*» реализуется через внешний ключ «*clientId*». Ограничение целостности – *cascade*;

- связь между таблицами «*Clients*» и «*Accounts*» реализуется через внешний ключ «*accountId*». Ограничение целостности – *cascade*;
- связь между таблицами «*Couriers*» и «*Accounts*» реализуется через внешний ключ «*accountId*». Ограничение целостности – *cascade*;
- связь между таблицами «*Managers*» и «*Accounts*» реализуется через внешний ключ «*accountId*». Ограничение целостности – *cascade*.

## 2.3 Архитектура приложения

В качестве архитектуры было выбрано трехслойное приложение. Которое состоит из трех уровней:

- уровень данных, по сути, является сервером, хранящим все данные приложения. Уровень данных содержит таблицы базы данных, файлы *JSON*, *PNG* и другие средства хранения данных приложения;
- бизнес-уровень работает как мост между уровнем данных и уровнем представления. Все данные проходят через бизнес-уровень перед их передачей уровню представления. Бизнес-уровень – сумма слоя бизнес-логики, слоя доступа к данным, объекта значения и других компонентов, используемых для добавления бизнес-логики;
- уровень представления – уровень, на котором пользователи взаимодействуют с приложением. Уровень представления содержит общий код интерфейса пользователя, отделенный код (код, содержащийся в отдельном файле, что позволяет отделить разметку от поведения, которое реализовано в коде) и конструкторов, используемых для представления информации пользователю.

Этот подход действительно очень важен, когда несколько разработчиков работают над одним и тем же проектом, и некоторые модули нужно повторно использовать в другом проекте. В некотором смысле, можно распределить работу между разработчиками и сопровождать ее в дальнейшем без особых проблем.

Так же, неотъемлемой частью проекта является тестирование. Тестирование – очень важный вопрос для архитектуры, когда рассматривается написание тестовых примеров для проекта. Так как она похожа на модульную архитектуру, очень удобно тестировать каждый модуль и отслеживать ошибки без прохождения через весь код.

## 3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

### 3.1 Описание интерфейса пользователя

Все современные приложения стараются делать интуитивно понятными и простыми в использовании, исходя из этого стоит уделить время для реализации хорошего графического интерфейса.

Так для незарегистрированного пользователя доступна лишь главная страница с регистрацией и авторизацией, страница с каталогом пицц и страница с комментариями других пользователей. Главная страница для незарегистрированного пользователя представлена на рисунке 3.1.

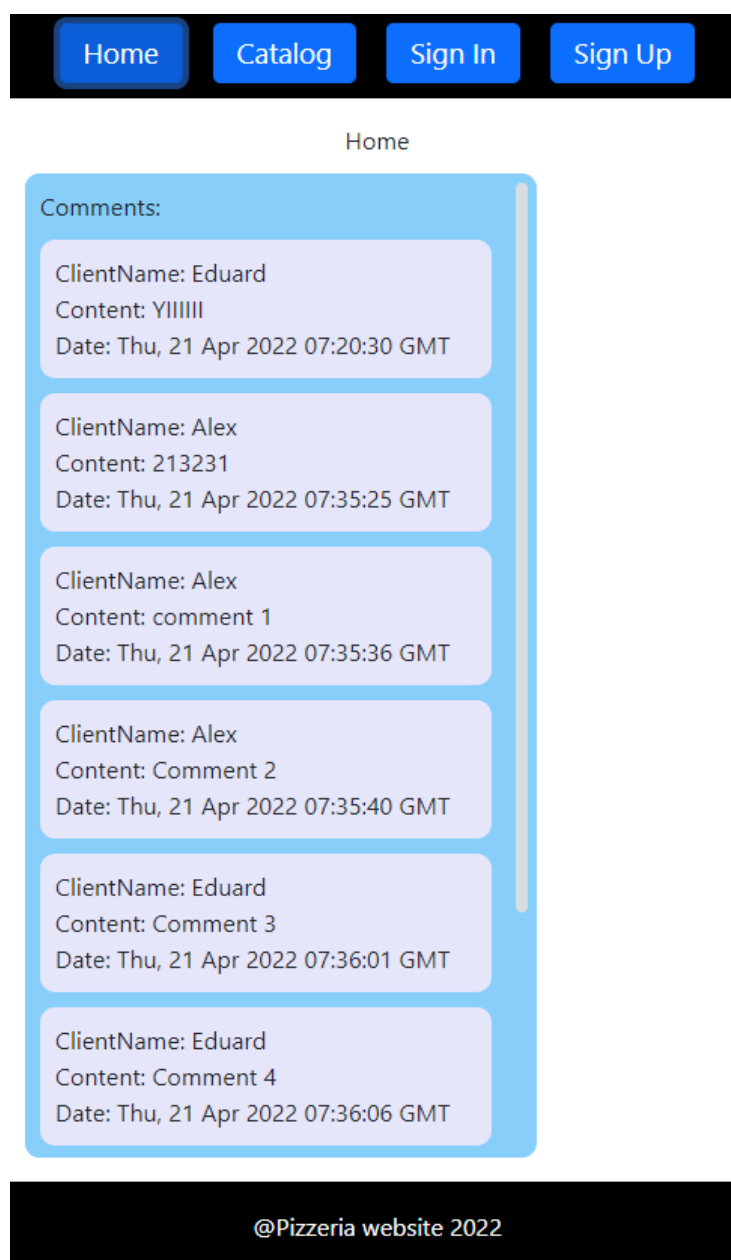


Рисунок 3.1 – Главная страница для незарегистрированного пользователя

Страница с каталогом пицц для гостя представлена на рисунке 3.2.

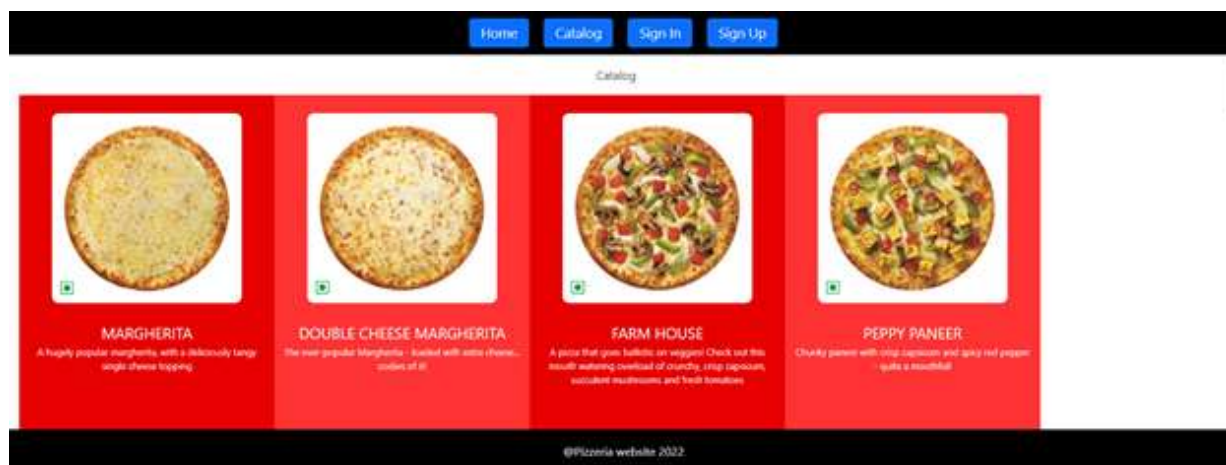


Рисунок 3.2 – Страница с каталогом пицц для гостя

На главной странице у гостя есть возможность зарегистрироваться, если он новый пользователь или же зайти в свой аккаунт, если он регистрировался ранее. Страница регистрации представлена на рисунке 3.3.

The image shows a web browser window displaying the 'Sign Up' page of a pizzeria website. At the top, there is a black navigation bar with four blue buttons labeled 'Home', 'Catalog', 'Sign In', and 'Sign Up'. Below the navigation bar, the page title 'Sign Up' is centered. The main content area is a white box with a blue border containing a registration form. The form has five input fields: 'Name', 'Phone Number', 'Description', 'Email address', and 'Password'. Each field has a placeholder text matching the label. Below the input fields, there are three buttons: 'Sign In' (blue), 'Home' (white), and 'Sign Up' (red). At the bottom of the page, a black footer contains the text '@Pizzeria website 2022'.

Рисунок 3.3 – Страница регистрации

Страница авторизации представлена на рисунке 3.4.

Home Catalog Sign In Sign Up

Sign In

Email address:

Email

Password

Password

Sign Up Home Sign In

@Pizzeria website 2022

Рисунок 3.4 – Страница авторизации

После авторизации пользователю доступны вкладки *Menu*, *Sign Out*, *Home*, *Catalog*, *Profile*. На странице *Catalog* пользователь может создать и составить необходимый заказ. Страница *Catalog* представлена на рисунке 3.5.

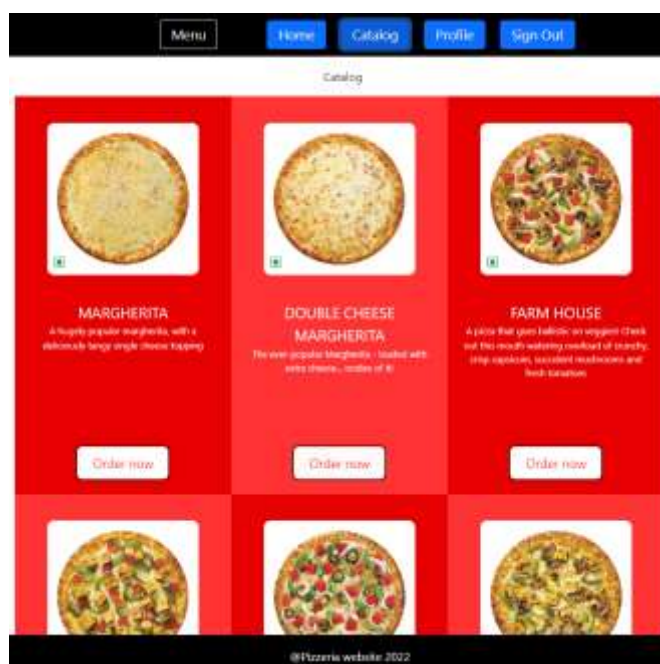


Рисунок 3.5 – Страница *Catalog* для клиента

Для оформления заказа пользователю необходимо выбрать желаемую пиццу из каталога и нажать кнопку «Order now». После чего откроется модальное окно, в котором пользователю будет необходимо ввести все необходимые данные для оформления заказа и последующей доставки, а именно выбрать количество пицц, размер пицц, указать адрес доставки и необходимость, в приложении предусмотрен вариант доставки самовывозом, а также дата и время доставки. Модальное окно оформления заказа представлено на рисунке 3.6.

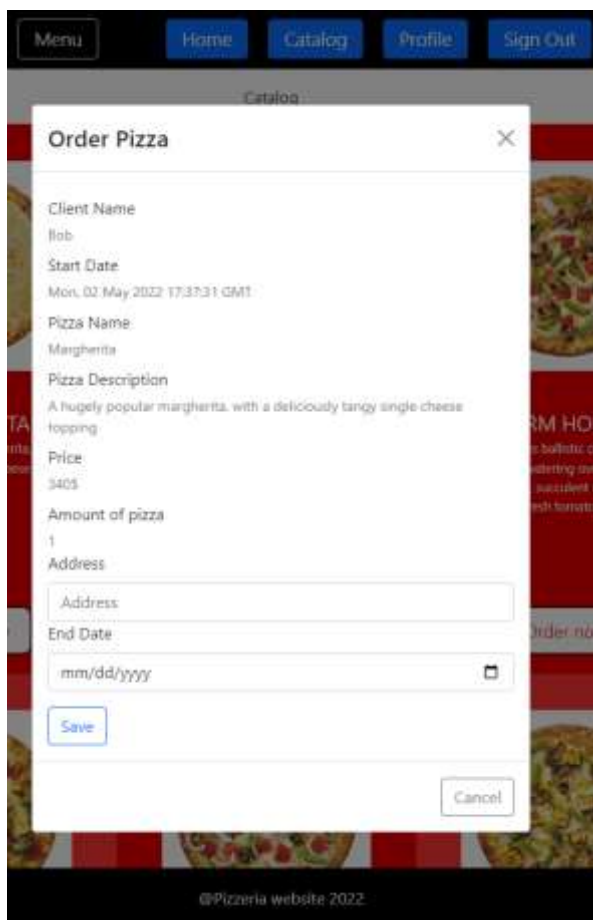


Рисунок 3.6 – Модальное окно для оформления заказа

После подтверждения заказа, то есть нажатия кнопки «Save», пользователь должен подтвердить заказ по телефону или электронной почте указанным при заполнении профиля на этапе регистрации.

Далее клиент может перейти во вкладку «*Menu*», где ему будут доступны история заказов за всё время и список последних заказов, которые ещё не выполнены или не отклонены.

В случае случайного оформления заказа, или изменении выбора до подтверждения заказа клиент может отменить его во вкладке «*Menu*» в пункте «*Pending orders*», отвечающем за заказы, находящиеся в состоянии ожидания подтверждения. Вкладка с текущими заказами клиента представлена на рисунке 3.7.

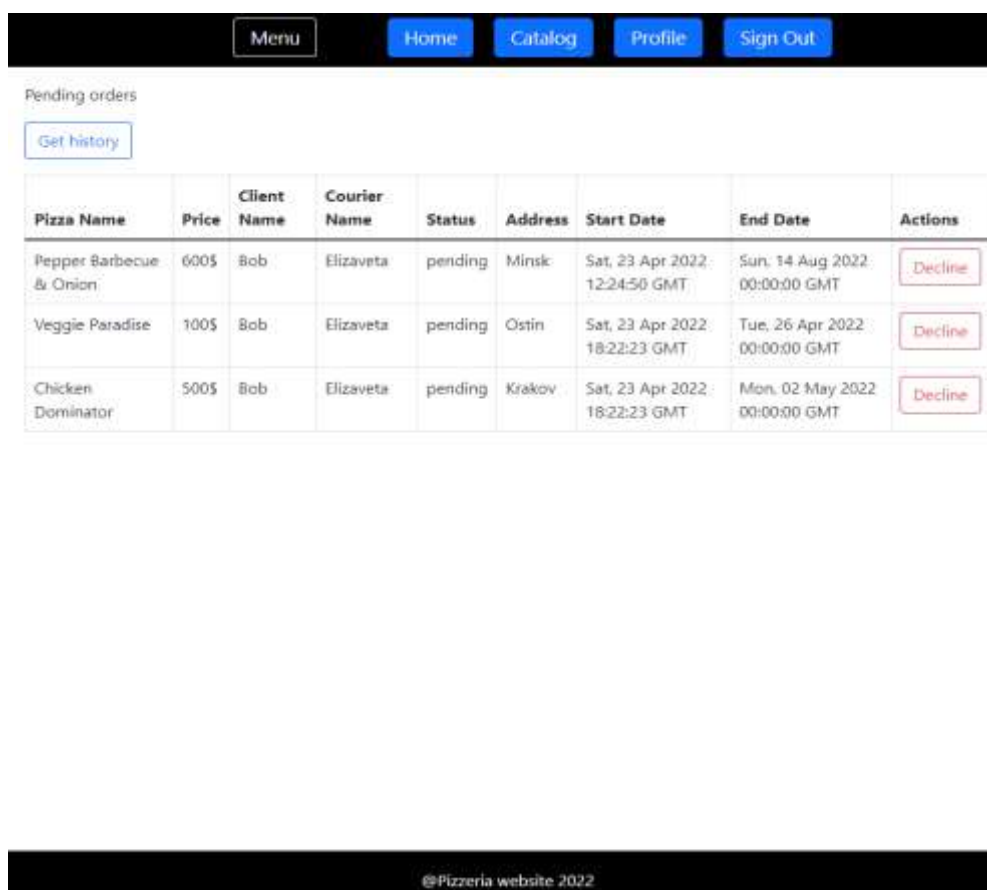


Рисунок 3.7 – Текущие заказы клиента

На этой же странице пользователь может произвести отмену заказа, если он его не подтвердил. В случае отмены заказа пользователю будет показано уведомление о том, что заказ был отменён, а также произойдёт обновление списка текущих заказов. Уведомление об отмене заказа представлено на рисунке 3.8.

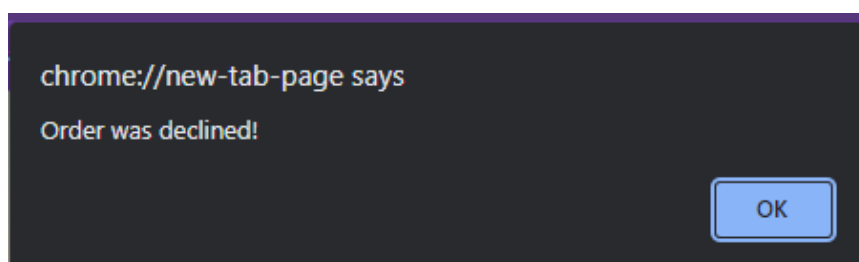


Рисунок 3.8 – Уведомление об отмене заказа

Пользователю также доступна история всех заказов во вкладке «*Menu*» в пункте «*History*», которая представлена на рисунке 3.9.

<a href="#">Menu</a> <a href="#">Home</a> <a href="#">Catalog</a> <a href="#">Profile</a> <a href="#">Sign Out</a>							
History							
<a href="#">Get pending orders</a>							
Pizza Name	Price	Client Name	Courier Name	Status	Address	Start Date	End Date
Farm House	360\$	Bob	Dmitriy	declined	Gomel	Sat, 23 Apr 2022 09:39:58 GMT	Sun, 03 Apr 2022 00:00:00 GMT
Margherita	340\$	Bob	Elizaveta	completed	Mozyr	Sat, 23 Apr 2022 09:40:34 GMT	Sat, 30 Apr 2022 00:00:00 GMT
Pepper Barbecue & Onion	600\$	Bob	Dmitriy	completed	Dublin	Sat, 23 Apr 2022 09:41:38 GMT	Sun, 03 Apr 2022 00:00:00 GMT
Farm House	360\$	Bob	Elizaveta	completed	Moscow	Sat, 23 Apr 2022 09:45:35 GMT	Mon, 18 Apr 2022 00:00:00 GMT
Non Veg Supreme	700\$	Bob	Dmitriy	completed	Brest	Sat, 23 Apr 2022 12:24:50 GMT	Thu, 30 Jun 2022 00:00:00 GMT
Indi Chicken Tikka	270\$	Bob	Elizaveta	completed	New York City	Sat, 23 Apr 2022 12:24:50 GMT	Sun, 24 Apr 2022 00:00:00 GMT
Chicken Golden Delight	600\$	Bob	Dmitriy	completed	London	Sat, 23 Apr 2022 15:47:47 GMT	Sat, 29 Oct 2022 00:00:00 GMT

Рисунок 3.9 – История всех заказов

Единственная роль в приложении, которая может добавлять комментарии – клиент. Комментарии добавляются после получения и употребления пиццы или хорошего опыта использования веб-сайта. Страница с формой добавления комментария представлена на рисунке 3.10.

<a href="#">Menu</a> <a href="#">Home</a> <a href="#">Catalog</a> <a href="#">Profile</a> <a href="#">Sign Out</a>	
Home	
Comments: <div> ClientName: Eduard  Content: Yllllll  Date: Thu, 21 Apr 2022 07:20:30 GMT </div> <div> ClientName: Alex  Content: 213231  Date: Thu, 21 Apr 2022 07:35:25 GMT </div> <div> ClientName: Alex </div>	Your comment <div> Comment </div> <div>Submit comment</div>
@Pizzeria website 2022	

Рисунок 3.10 – Страница с формой добавления комментариев



Клиент имеет возможность изменить свой комментарий, для этого ему необходимо удалить прошлый и составить новый. Механизм удаления комментариев представлен на рисунке 3.11.

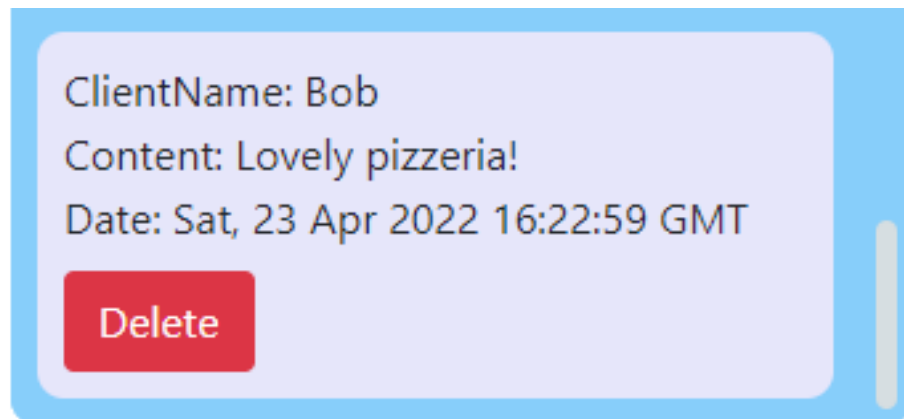


Рисунок 3.11 – Механизм удаления комментариев

Для проверки правильности данных введенных при регистрации пользователь может посетить страницу «*Profile*», всегда находящуюся вверху страницы, которая представлена на рисунке 3.12.

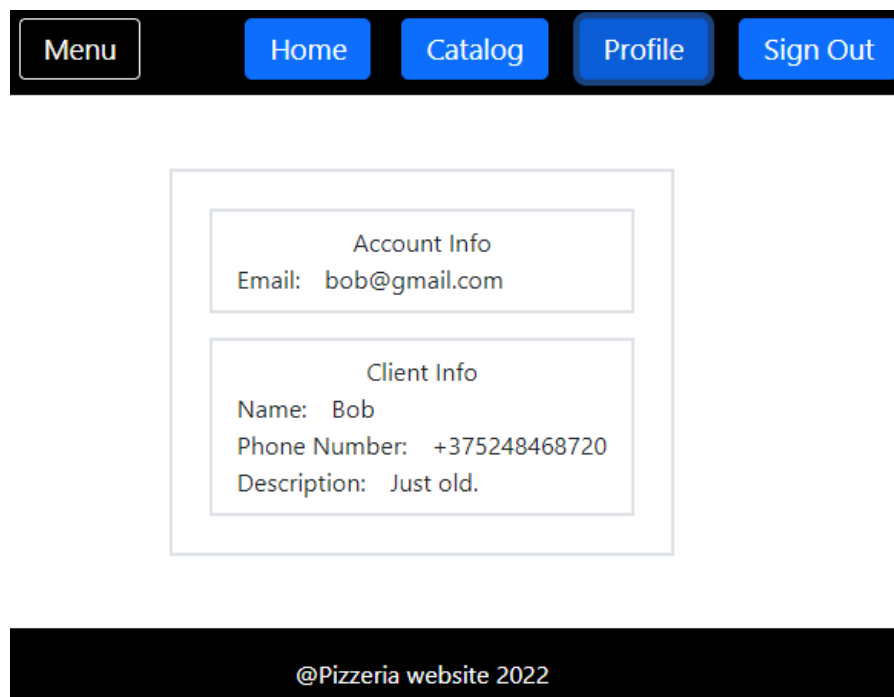


Рисунок 3.12 – Страница «*Profile*»

В приложении также описаны роли курьера и менеджера, для полноценного функционирования службы доставки.

Начинается использование приложения для роли курьера, как и для всех остальных ролей, всё также с авторизации, после которой пользователю доступны те же вкладки, что и клиенту, а именно «Menu», «Home», «Catalog» и «Profile». Вкладка «Catalog» повторяет вкладку для неавторизованного пользователя, потому что никто, кроме клиента не имеет возможности заказывать пиццу. Вкладка «Profile» повторяет вкладку для клиента и отображает данные пользователя, указанные при регистрации. Вкладка «Home» отображает комментарии, как и для гостя, без возможности добавления или удаления комментариев.

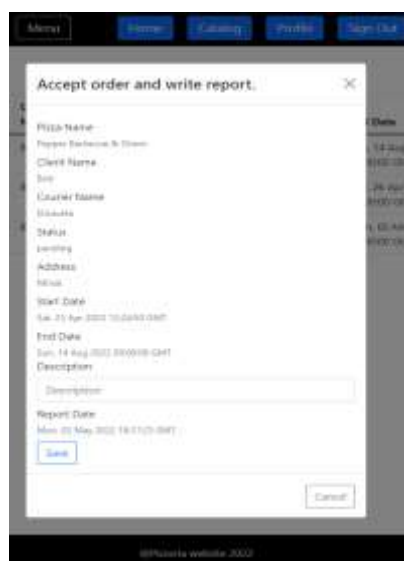
Во вкладке «Menu» курьер может ознакомиться с заказами, которые ему необходимо выполнить. Страница с текущими заказами курьера представлена на рисунке 3.13.



Pizza Name	Price	Client Name	Courier Name	Status	Address	Start Date	End Date	Actions
Pepper Barbecue & Onion	600\$	Bob	Elizaveta	pending	Minsk	Sat, 23 Apr 2022 12:24:50 GMT	Sun, 14 Aug 2022 00:00:00 GMT	Accept
Veggie Paradise	100\$	Bob	Elizaveta	pending	Ostin	Sat, 23 Apr 2022 18:22:23	Tue, 26 Apr 2022 00:00:00	Accept

Рисунок 3.13 – Страница с текущими заказами курьера

На этой же странице курьер может подтвердить заказ и написать отчёт по доставке. Модальное окно для написания отчёта по заказу представлено на рисунке 3.14.



Accept order and write report.

Pizza Name: Pepper Barbecue & Onion

Client Name: Bob

Courier Name: Elizaveta

Status: pending

Address: Minsk

Start Date: Sat, 23 Apr 2022 12:24:50 GMT

End Date: Sun, 14 Aug 2022 00:00:00 GMT

Description:

Report Date: Mon, 25 May 2022 18:51:25 GMT

Save

Cancel

Рисунок 3.14 – Модальное окно добавления отчёта

После авторизации за аккаунт с ролью менеджера пользователю доступны все стандартные вкладки, как и у предыдущих ролей. Единственным значимым отличием, является содержимое вкладки «*Menu*».

Во вкладке «*Menu*» менеджер может ознакомиться со списком курьеров, их отчётами и деталями как отчётов, так и заказов, по которым они были составлены. Вкладка «*Menu*» для менеджера представлена на рисунке 3.15.

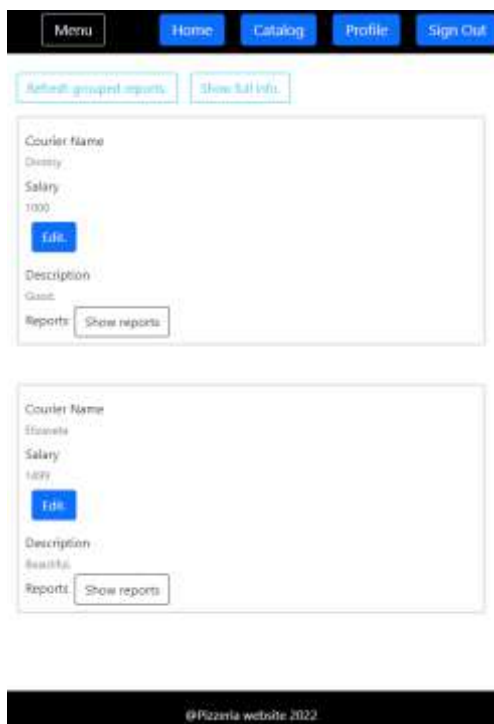


Рисунок 3.15 – Список курьеров

После ознакомления с отчётами по выполненным заказам, менеджер может поощрить или покарать курьера, вышеперечисленные действия осуществляются через форму изменения заработной платы, которая представлена на рисунке 3.16.

Рисунок 3.16 – Форма изменения заработной платы

Таким образом, можно считать, что интерфейс пользователей был полностью описан.

### 3.2 Описание классов

Путем анализа предметной области можно выделить следующие доменные классы, соответствующие сущностям базы данных, описание которых представлено в таблице 3.1.

Таблица 3.1 – Описание доменных классов

Название класса	Назначение	Соответствующая сущность базы данных
<i>Account</i>	Хранение информации о аккаунтах	Таблица 2.2
<i>Client</i>	Хранение информации о клиентах	Таблица 2.3
<i>Courier</i>	Хранение информации о курьерах	Таблица 2.5
<i>Manager</i>	Хранение отзывов о менеджерах	Таблица 2.6
<i>Pizza</i>	Хранение информации о пиццах	Таблица 2.8
<i>Status</i>	Хранение информации о статусах заказа	Таблица 2.10
<i>Order</i>	Хранение информации о заказах	Таблица 2.7
<i>Comment</i>	Хранение информации о комментариях	Таблица 2.4
<i>Report</i>	Хранение информации о отчётах по заказам	Таблица 2.9

Каждой таблице из базы данных соответствует один класс модели из библиотеки *Sequelize*, который описывает операции *CRUD* для доступа к базе данных, из этого следует что для каждой вышеописанной сущности должен существовать класс модели, который будет реализовывать *CRUD* операции.

Реализация классов доступа к базе данных находится на *DataAccess*, содержащий классы типа *Entity*, *Model*.

Классы *Entitis* полностью повторяют таблицы из базы данных и служат для хранения информации из таблиц на слое *DataAccess*.

Классы *Models* предназначены для обращения к базе данных от соответствующих классов *DTO*, реализуют интерфейс *IModel<T>*. Описание интерфейса представлено в таблице 3.2.

Таблица 3.2 – Описание интерфейса *IModel*

Название метода	Назначение
<i>findAll</i>	Обращение к указанной таблице и возврат всех существующих данных
<i>findOne</i>	Обращение к элементу таблицы по его переданному фильтру и возврат данных о элементе
<i>create</i>	Создание новой записи в таблице базы данных
<i>update</i>	Обновление информации о записи в таблице из базы данных
<i>delete</i>	Удаление записи из таблицы базы данных

Интерфейсы и классы реализации пользовательских функций располагаются в слое *BusinessLogic*, они необходимы для обработки данных полученных с пользовательского интерфейса. Так, нагрузка на приложение будет разделена на серверную и пользовательскую часть, где на пользовательской части будут создаваться запросы, которые будут отправляться на сервер, где они и будут обрабатываться.

Классы *DTOs* повторяют классы *Entities* на слое *DataAccess*.

Классы *Routers* реализуют методы для обработки пользовательских запросов, в данном случае это обработчики *clientRouter*, *courierRouter*, *managerRouter*, *catalogRouter*, *commentsRouter*, *signInRouter*, *signUpRouter*, реализующие методы необходимые для выполнения поставленных задач.

Рассмотрим *clientRouter* из *Routers*, он отвечает за методы, необходимые для работы с заказами, в том числе оформление, отмена и просмотр. Реализуемые классом методы представлены в таблице 3.5.

Таблица 3.5 – Описание методов, реализуемых *clientRouter*

Название	Передаваемые значения	Возвращаемые значения	Описание метода
1	2	3	4
<i>getOrdersByClientId</i>	<i>int, boolean</i>	<i>FullOrder[]</i>	Обращение к модели заказов для получения всех либо текущих заказов, оформленных клиентом

Продолжение таблицы 3.5

1	2	3	4
<i>declineOrder</i>	<i>int</i>	<i>void</i>	Обращение к модели заказов для отмены заказа по номеру

Обработчик *courierRouter* работает с запросами роли курьера, описание которых представлено в таблице 3.6.

Таблица 3.6 – Описание методов, реализуемых *courierRouter*

Название	Передаваемые значения	Возвращаемые значения	Описание метода
<i>getOrdersBy-CourierId</i>	<i>int, boolean</i>	<i>FullOrder[]</i>	Обращение к модели заказов для получения всех или текущих заказов для курьера по номеру
<i>acceptOrder</i>	<i>int</i>	<i>void</i>	Обращение к модели заказов для подтверждения доставки

Обработчик *managerRouter* работает с запросами роли менеджера, описание которых представлено в таблице 3.7.

Таблица 3.7 – Описание методов, реализуемых *managerRouter*

Название	Передаваемые значения	Возвращаемые значения	Описание метода
<i>getGroupedReports</i>	—	<i>GroupedReports</i>	Обращение к модели отчётов для получения всех отчётов
<i>editCourierSalary</i>	<i>int, int</i>	<i>void</i>	Обращение к модели курьеров для редактирования зарплаты по номеру курьера

Обработчик *catalogRouter* работает с запросами, связанными с каталогом пицц, описание которых представлено в таблице 3.8.

Таблица 3.8 – Описание методов, реализуемых *catalogRouter*

Название	Передаваемые значения	Возвращаемые значения	Описание метода
<i>getAll</i>	—	<i>Pizza[]</i>	Обращение к модели пицц для получения всех пицц
<i>addOne</i>	<i>PizzaWithoutId</i>	<i>Pizza</i>	Обращение к модели пицц для добавления пиццы

Обработчик *commentsRouter* работает с запросами, связанными с комментариями, описание которых представлено в таблице 3.9.

Таблица 3.9 – Описание методов, реализуемых *commentsRouter*

Название	Передаваемые значения	Возвращаемые значения	Описание метода
<i>getAll</i>	—	<i>Comment[]</i>	Обращение к модели комментариев для получения всех комментариев
<i>addOne</i>	<i>CommentWithoutId</i>	<i>Comment</i>	Обращение к модели комментариев для добавления комментария
<i>deleteOne</i>	<i>int</i>	<i>void</i>	Обращение к модели комментариев для удаления комментария по номеру

Обработчик *signInRouter* работает с запросами, связанными со входом, описание запросов представлено в таблице 3.10.

Таблица 3.10 – Описание методов, реализуемых *signInRouter*

Название	Передаваемые значения	Возвращаемые значения	Описание метода
<i>trySignIn</i>	<i>string, string</i>	<i>Account // null</i>	Обращение к моделям аккаунтов, клиентов, менеджеров и курьеров для получения информации профиля по заданным логину и паролю

Обработчик *signUpRouter* работает с запросами, связанными со входом, описание запросов представлено в таблице 3.11.

Таблица 3.11 – Описание методов, реализуемых *signUpRouter*

Название	Передаваемые значения	Возвращаемые значения	Описание метода
<i>trySignUp</i>	<i>Client &amp; Account</i>	<i>(Account &amp; Client) // null</i>	Обращение к моделям аккаунтов и клиентов для создания новой учётной записи

Выше были описаны обработчики со стороны сервера, отвечающего за базу данных, однако в приложении был реализован ещё один сервер, для работы с пользовательским интерфейсом, он обрабатывает *URL* адреса *html* страниц, которые доступны пользователю. Вся разметка хранится в файлах с расширением *.tsx* – файлы, которые совмещают *TypeScript* код и *html* разметку. Пользовательский интерфейс реагирует на любые изменения состояний, от ввода данных в формы, до перехода на другие страницы, что было достигнуто использованием библиотеки *React.js*.

### 3.3 Тестирование и верификация

В качестве примеров верификации будут рассмотрены функции создания заказа, его оплаты, подтверждение и выполнение его курьером, возможность оставить отзыв, просмотр каталога пицц.

#### 1. Создание и обработка заказа.



При нажатии на кнопку создания заказа, приложению необходимо сделать запросы в базу данных по пиццам, заказам, статусам заказов и курьерам, чтобы подготовить данные для заполнения.

Для выбора курьера, программе необходимо выгрузить из базы данных список всех курьеров.

Далее, на выборе статуса заказа, программе необходимо найти статус, который отвечает за неподтвержденный заказ.

В качестве даты оформления заказа программа использует текущую дату, получаемую с помощью встроенных классов.

После изменения состояние, программа так же меняет его и у курьера, после чего он может подтвердить оплаченный заказ.

Как видно из рисунков, представленных в 3.1, программа справляется с поставленной задачей. Таким образом, тест прошел успешно.

## 2. Просмотр каталога пицц.

На главной странице каждого пользователя существует кнопка «*Catalog*» при нажатии на которую открывается новая страница с каталогом.

После нажатия на данную кнопку программа запрашивает все существующие пиццы и их описание.

Как видно из рисунков, представленных в 3.1, программа справляется в поставленной задачей, из чего следует – тест прошел успешно.

## 3. Возможность оставить отзыв.

На странице «*Home*» клиент может увидеть все комментарии, а также форму для создания нового.

После ввода текста комментария необходимо нажать кнопку «*Submit comment*», после чего комментарий будет успешно добавлен и форма очистится.

Как видно из рисунков, представленных в 3.1, программа справляется в поставленной задачей, из чего следует – тест прошел успешно.

Для комфортной работы приложения необходимо предусмотреть обработку исключительных ситуаций. В связи с этим были обработаны следующие исключительные ситуации. Обработанные исключительные ситуации были описаны в таблице 3.12.

Таблица 3.12 – Тестирование приложения

Исключительная ситуация	Результат
1	2
Отправка пустого поля в случае, когда поле должно быть заполнено	Сообщение о том, что поле не может быть пустым

Продолжение таблицы 3.12

1	2
Попытка оплатить заказ, когда на балансе недостаточно средств	Сообщение о предложении пополнить счет
Попытка обратиться к функционалу приложения, не находясь на соответствующей роли	Сообщение о недоступности вызываемого функционала
Попытка выбора даты заказа, которая уже занята	Невозможность выбора, т.к. кнопка будет неактивна
Вход с несуществующим логином или паролем	Вывод сообщения о том, что следует проверить корректность вводимых данных

Также было проведено тестирование слоя *DataAccess*, оно заключается в проверке доступа к данным, необходимо проверить какие данные возможно отправить в базу данных, какие невозможно, так же работоспособность каждого метода доступа. В таблице 3.13 приведен пример тестирования класса модели *OrderModel*, осуществляющего доступ к данным таблице *Orders*.

Таблица 3.13 – Валидационное тестирование класса *OrderModel*

Название теста	Поступаемые данные	Описание теста	Ожидаемый результат
1	2	3	4
<i>getOrders_WhenDatabaseIsCorrect_ShouldReturnCollectionOfOrders</i>	-	Вызывается метод <i>GetAll()</i> и данные полученные из него проверяются на соответствие с данными в базе данных	Полное соответствие данных
<i>getOrderById_WhenTheItemIsInTheDatabase_ShouldReturnItem</i>	<i>int index</i> , в значении 1	Вызывается метод <i>findOne(index)</i> , по заранее переданному значению. Полученный элемент проверяется на соответствие с базой данных	Полное соответствие данных
<i>CreateNewOrder_WhenTheAllItemsIsCorrect_ShouldReturnNewItem</i>	-	Вызывается метод <i>create</i> , в который передается заранее созданный объект класса <i>Order</i> . Затем, созданный элемент проверяется на соответствие с тем, что передали в метод	Полное соответствие элементов

Продолжение таблицы 3.13

1	2	3	4
<i>create-NewOrder_WhenAllDateIsNotCorrect_ShouldReturnDateException</i>	<i>int pizzaId, int clientId, string address, string startDate</i> , в не валидных значениях каждого передаваемого элемента	Вызывается метод <i>Create</i> , куда передается объект, созданный по передаваемым данным. Далее проверяется созданный элемент	Ошибка <i>DataException</i>
<i>create-NewOrder_WhenAllDateIsBoundary_ShouldAddNewItem</i>	<i>int pizzaId, int clientId, string address, string startDate</i> , в граничных значениях каждого передаваемого элемента	Вызывается метод <i>Create</i> , куда передается объект, созданный по передаваемым данным. Далее проверяется созданный элемент	Полное соответствие элементов
<i>deleteOrder_WhenIdIsCorrect_ShouldDeleteOrder</i>	-	Вызывается метод <i>Delete</i> , после чего данные базы данных сверяются	В базе данных не должно остаться удаляемого элемента
<i>getOrderById_WhenAnItemIsNotInTheDatabase_ShouldReturnNull</i>	<i>int index</i> в значении 2, -1, null	Вызывается метод <i>GetById(index)</i> , по заранее переданному значению.	Полученный элемент должен быть <i>null</i>

Таким образом приложение можно считать полностью протестированным.

## ЗАКЛЮЧЕНИЕ

В результате курсового проектирования был создан продукт, цель которого – автоматизация работы службы доставки пицц. Созданная программа обладает функциями, необходимыми для формирования клиентом заказа, его продвижения и организации работы сотрудников, сбора статистики за определённый период, создания отчётов и их последующего экспортирования в новый документ текстового редактора *MS Word*. В процессе создания продукта были пройдены такие этапы разработки, как постановка задачи, составление логической модели предметной области, создание и реализация алгоритма работы программы с последующим тестированием.

В процессе разработки были рассмотрены платформа для исполнения *Javascript* кода *Node.js*, библиотека для построения пользовательских интерфейсов *React.js*, была разработана архитектура приложения на основе паттерна *MVC*, что позволило создать веб-приложение с использованием самых современных технологий.

Продукт был разработан, отлажен и протестирован. Поставленные в курсовом проекте задачи были полностью выполнены, однако всё ещё есть некоторые возможности, реализовав которые можно добиться более углубленного моделирования предметной области.

## **Список использованных источников**

1. Модели баз данных, системы управления базами данных [Электронный ресурс]. – Интернет-Технологии.ру, 2020. – Режим доступа: <https://www.internet-technologies.ru/articles/modeli-baz-dannyh-sistemy-upravleniya-bazami-dannyh.html>. – Дата доступа – 20.04.2022.
2. Руководство по Node.js [Электронный ресурс]. METANIT.COM Сайт о программировании, 2019 – Режим доступа: <https://metanit.com/sharp/express/>. – Дата доступа: 21.04.2022.
3. В.А. Биллиг. Основы программирования на TypeScript (TypeScript 4.0, Jet Brains Rider). – М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2010. – 584 с.
4. Стефанов, Ст. Программирование на платформе Node.js 12 на языке JavaScript. 3-е изд. – СПб.: Питер, 2016 – 896 с.

**ПРИЛОЖЕНИЕ А**  
**(обязательное)**  
**Листинг программного кода**