МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Специализация 1-40 01 01 10 «Программное обеспечение информационных технологий (программирование интернет-приложений)»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту на тему:**

Web-приложение «QR-меню»

Выполнил студент Левша Марк Сергеевич

(Ф.И.О.)

Руководитель проекта асс. Бернацкий П.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

И. о. заведующего кафедрой ст. препод. Блинова Е.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты асс. Бернацкий П.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер асс. Бернацкий П.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2023

**Реферат**

Пояснительная записка курсового проекта содержит 31 страницу пояснительной записки, 27 иллюстраций, 7 листингов, 5 источников литературы, 3 приложения.

NEST, JAVASCRIPT, REACT, REST, PRISMA, MSSQL.

Целью курсового проекта является web-приложение, которое предоставляет удобный интерфейс для составления QR-меню.

Пояснительная записка курсового проекта состоит из реферата, содержания, введения, пяти глав, заключения и списка использованных источников.

В первой главе проводится постановка задачи, обзор аналогов приложения, описаны достоинства и недостатки каждого из них.

Во второй главе изложено проектирование web-приложения: описана структура базы данных, серверной части web-приложения, приведены диаграмма вариантов использования и структурная схема приложения.

В третьей главе описывается структура серверной и клиентской части web-приложения, а также их разработка, приведены листинги с реализацией тех или иных частей приложения.

В четвёртой главе описывается проведенное ручное позитивное и негативное тестирование web-приложения.

В пятой главе изложено руководство пользователя по эксплуатации разработанного web-приложения.

В заключении приведены результаты проделанной работы, описан функционал получившегося web-приложения.

**Содержание**

[Введение 4](#_Toc153486366)

[1 Постановка задачи 5](#_Toc153486367)

[1.1 Анализ аналогов 5](#_Toc153486368)

[1.1.1 Relax.by 5](#_Toc153486369)

[1.1.2 Foodeon 6](#_Toc153486370)

[1.1.3 Выводы по разделу 8](#_Toc153486371)

[2 Проектирование web-приложения 9](#_Toc153486372)

[2.1 Проектирование архитектуры базы данных 9](#_Toc153486373)

[2.2 Проектирование структуры сервера 10](#_Toc153486374)

[2.3 Структурная схема приложения 11](#_Toc153486375)

[2.4 Диаграмма вариантов использования 12](#_Toc153486376)

[2.5 Выводы по разделу 12](#_Toc153486377)

[3 Разработка web-приложения 13](#_Toc153486378)

[3.1 Реализация серверной части 13](#_Toc153486379)

[3.1.1 Структура проекта 13](#_Toc153486380)

[3.1.2 Конкретные реализации 15](#_Toc153486381)

[3.2 Реализация клиентской части 18](#_Toc153486382)

[3.2.1 Структура проекта 18](#_Toc153486383)

[3.2.2 Компоненты 19](#_Toc153486384)

[3.3 Выводы по разделу 21](#_Toc153486385)

[4 Тестирование web-приложения 22](#_Toc153486386)

[4.1 Ручное тестирование 22](#_Toc153486387)

[4.2 Выводы по разделу 24](#_Toc153486388)

[5 Руководство пользователя 25](#_Toc153486389)

[5.1 Методика использования приложения 25](#_Toc153486390)

[5.2 Выводы по разделу 29](#_Toc153486391)

[Заключение 30](#_Toc153486392)

[Список используемых источников 31](#_Toc153486393)

[ПРИЛОЖЕНИЕ А 32](#_Toc153486394)

[ПРИЛОЖЕНИЕ Б 33](#_Toc153486395)

[ПРИЛОЖЕНИЕ В 34](#_Toc153486396)

# Введение

Web-приложения стали неотъемлемой частью нашей повседневной жизни, обеспечивая удобный и быстрый доступ к различным услугам и ресурсам. Одной из сфер, где web-приложения проявляются наиболее эффективно, является сфера ресторанного и кулинарного бизнеса. В рамках данного курсового проекта мы сосредоточимся на создании web-приложения QR-меню для ресторанов.

Приложение QR-меню охватывает широкую аудиторию, включая различные сферы бизнеса и категории пользователей, например, владельцы и посетители ресторанов и кафе, желающие удобно ознакомиться с меню без контакта с физическим меню или официантом.

В условиях современной ресторанной индустрии, важно обеспечить гостей удобством и быстрым доступом к информации о предлагаемых блюдах. Проект QR-меню решает эту задачу, предоставляя администраторам ресторанов интуитивно понятный интерфейс для составления и редактирования меню, а гостям — удобный доступ к нему через QR-код.

Внедрение QR-меню, особенно в сфере ресторанного бизнеса, актуально с учетом ряда проблем, связанных с экологией и вредом бумажных носителей информации. Традиционные бумажные меню могут привести к значительному потреблению бумаги и, следовательно, к вырубке деревьев, используемых для производства бумаги. Использование цифровых QR-меню уменьшает потребность в распечатке бумажных копий, способствуя уменьшению экологического воздействия. Также изменение бумажного меню может быть трудоемким процессом, особенно при внесении частых обновлений или внесении коррекций. Внедрение QR-меню выглядит как ответ на современные вызовы, связанные с экологической устойчивостью.

Таким образом, целью курсового проектирования является web-приложение, которое предоставляет удобный интерфейс для составления QR-меню.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* провести аналитический обзор литературы, аналогов приложения;
* спроектировать архитектуру приложения;
* спроектировать структуру базы данных;
* разработать приложение;
* протестировать приложение;
* написать руководство пользователя.

# 1 Постановка задачи

## **1.1 Анализ аналогов**

В данном разделе был проведен анализ конкурентов, предоставляющих услуги QR-меню. Анализ проводился, основываясь на информации из открытых источников.

**1.1.1 Relax.by**

Одной из ряда услуг, которых предоставляет Relax.by [1], является QR-меню. Чтобы получить возможности сервиса нужно оставить заявку на сайте, заполнив форму, где нужно указать контактные данные и указать в каких целях вы будете использовать сервис, после этого менеджер свяжется с вами в течении двадцати четырех часов, чтобы обговорить детали.

Также вы можете ознакомиться с возможностями, которые предлагает сервис. Пример приведен на рисунке 1.1.

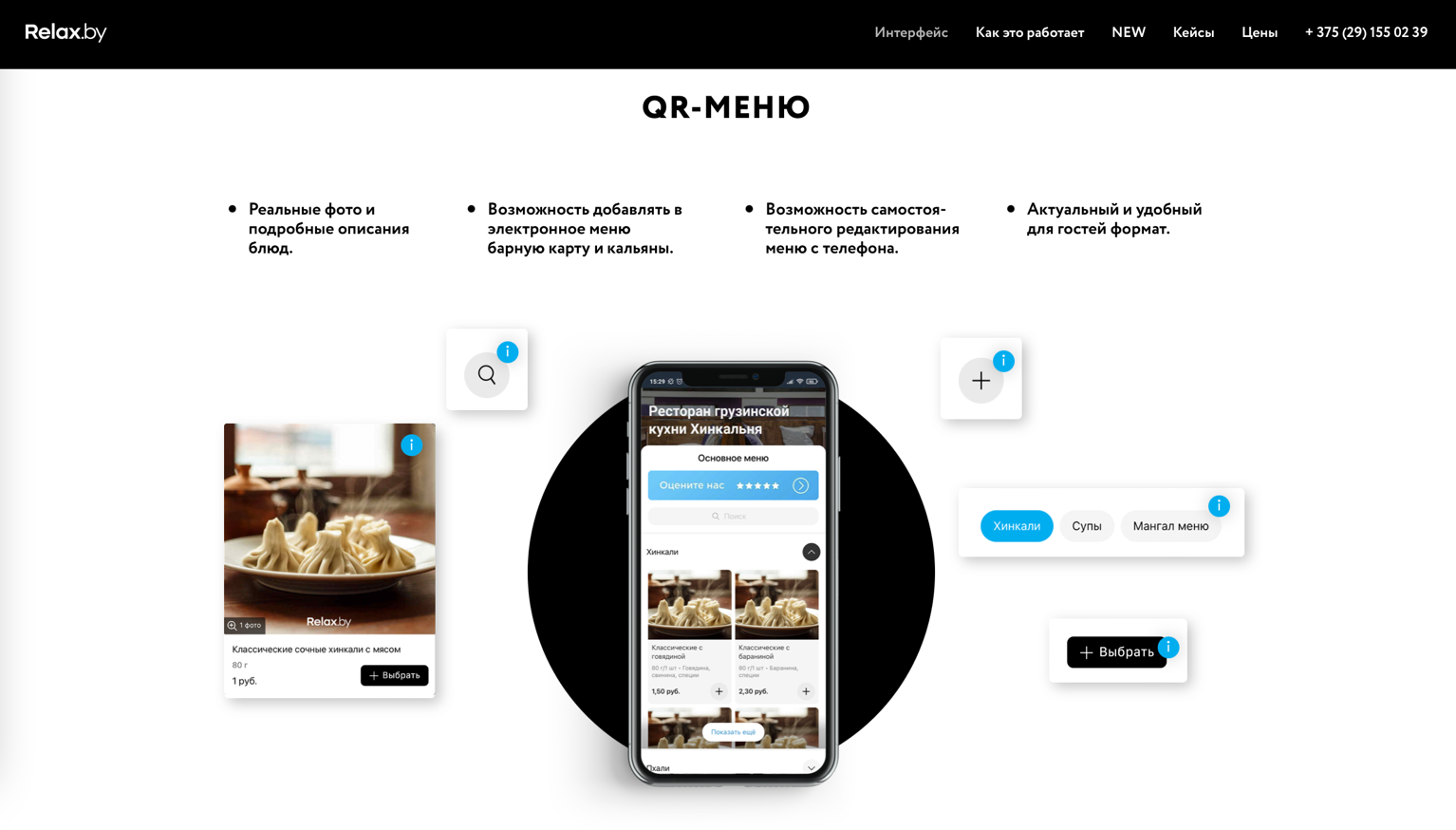


Рисунок 1.1 – Возможности предлагаемые QR-меню от Relax.by

На сайте представлены варианты размещения QR-кодов непосредственно в ресторане, важно отметить, что пользователь сервиса не может самостоятельно получить QR-код и разместить его в ресторане без взаимодействия с Relax.by. Примеры размещения QR-кодов представлены на рисунке 1.2.



Рисунок 1.2 – Примеры размещения QR-кодов в ресторанах

Также приведена статистика использования сервиса на примере ресторана «Хинкальня» и вы можете ознакомиться с тарифами и дополнительными услугами.

Данный сервис имеет ряд плюсов, таких как: реальные фото и подробные описания блюд, возможность редактирования меню с телефона, возможность осуществлять заказы. Однако есть и недостатки, чтобы получить физические QR-кода нужно взаимодействовать с владельцем сервиса, отсутствует защита от пользователей, которые хотят посмотреть меню за пределами ресторана, высокая цена за месячную подписку относительно аналогов в Европе, цена выше в 2 раза, отсутствие сортировки блюд.

**1.1.2 Foodeon**

Foodeon [2] – платформа, которая предоставляет возможность создания QR-меню.

Карточка блюда включает в себя фото блюда, цену, описание и ключевые слова, по которым пользователь может подчеркнуть важные особенности блюда, например, является блюдо вегетарианским или содержит ли оно лактозу. Пример представлен на рисунке 1.3.

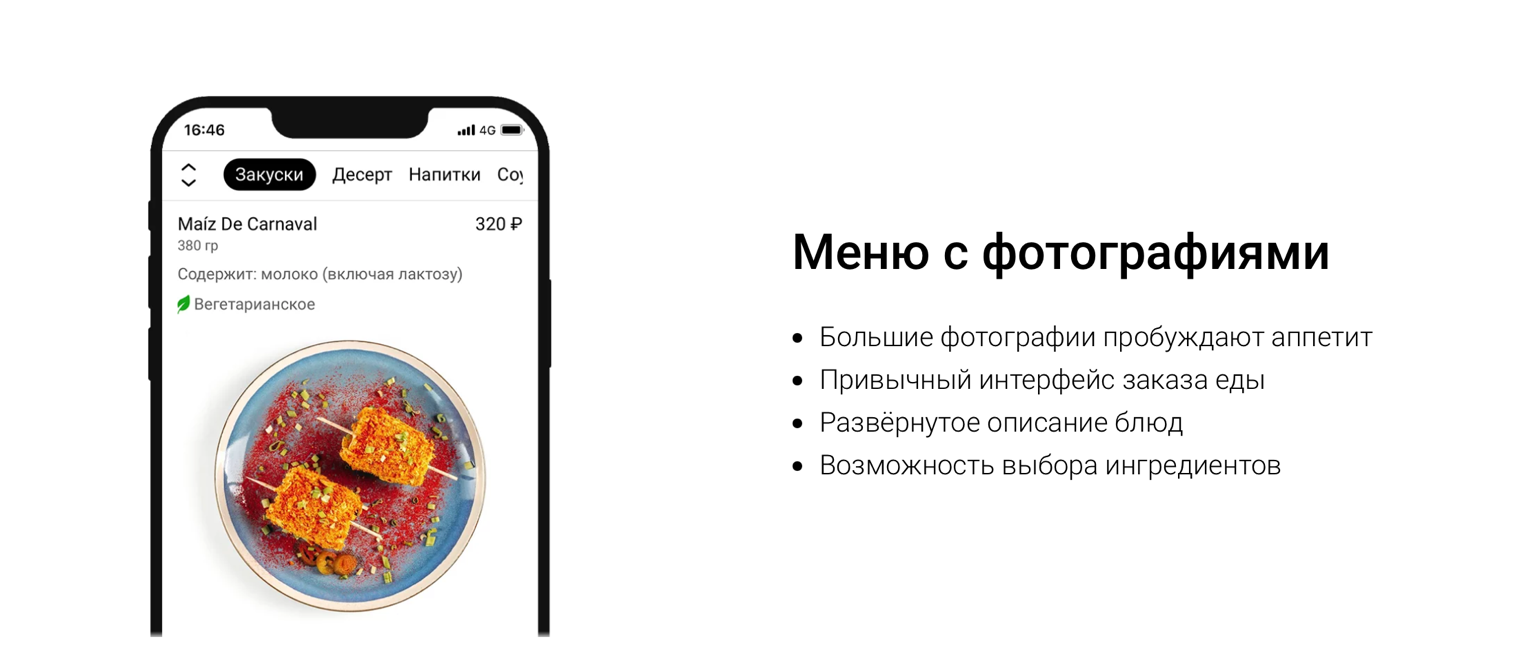


Рисунок 1.3 – Пример карточки блюда

В web-приложении можно добавлять блюда в корзину, но заказ нельзя сделать из приложения, для этого нужно позвать официанта. Пример корзины представлен на рисунке 1.4.



Рисунок 1.4 – Пример корзины

Также можно оставлять полноценные отзывы и бронировать столик из дома, несомненным плюсом является наличие акций и предложений. Сортировка блюд в приложении является отличным решением для такого вида приложения. Страница с бронированием, отзывами и акциями представлена на рисунке 1.5.

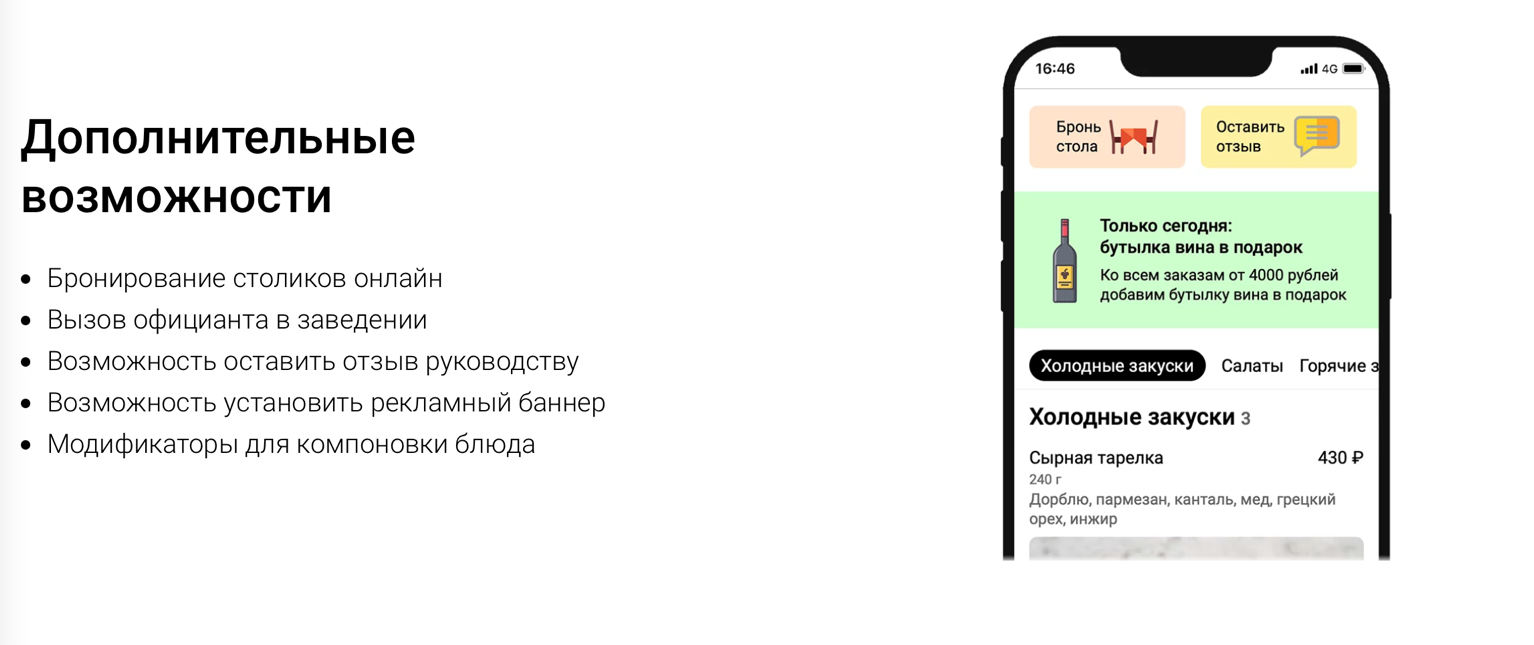


Рисунок 1.5 – Страница с бронированием, отзывами и акциями

Данный сервис имеет ряд плюсов, таких как: реальные фото и подробные описания блюд с ключевыми словами, возможность редактирования меню с телефона, возможность добавлять блюда в корзину, бронирование столов и возможность оставлять отзывы, а также наличие акций и предложений и сортировки. Однако есть и недостатки, отсутствует защита от пользователей, которые хотят посмотреть меню за пределами ресторана, отсутствие возможности заказать блюда из корзины без официанта.

**1.1.3 Выводы по разделу**

После проведенного тщательного анализа функциональных требований был сформирован обширный набор функций, которыми должно обладать разрабатываемое приложение. Эти функциональности включают в себя возможность добавления категорий и создания навигационной панели на их основе, добавление блюд с указанием цен и описанием, а также генерацию QR-кодов с опцией сохранения для дальнейшей печати. Пользователи смогут легко находить блюда по наименованию, осуществлять сортировку меню по ценам и алфавиту. Дополнительно предусмотрена функциональность сокрытия меню от посетителей, находящихся за пределами ресторана.

Особое внимание уделено внедрению двух ролей в системе: администратора и гостя. Администратор имеет полный контроль над управлением категориями, блюдами и информацией о ресторане. Также, администратор может генерировать QR-коды и сохранять их на физическом носителе. Гости, в свою очередь, могут легко просматривать меню, сортировать блюда и проводить поиск, что обеспечивает удобство использования приложения. Этот дифференцированный набор функций создает более гибкую и удовлетворяющую потребности систему для разных категорий пользователей.

# 2 Проектирование web-приложения

## **2.1 Проектирование архитектуры базы данных**

В качестве СУБД была выбрана MSSQL. Она предоставляет надежное и масштабируемое решение для хранения, управления и обработки больших объемов структурированных данных.

В базе данных находится 4 таблицы, схема базы данных представлена на рисунке 2.1.

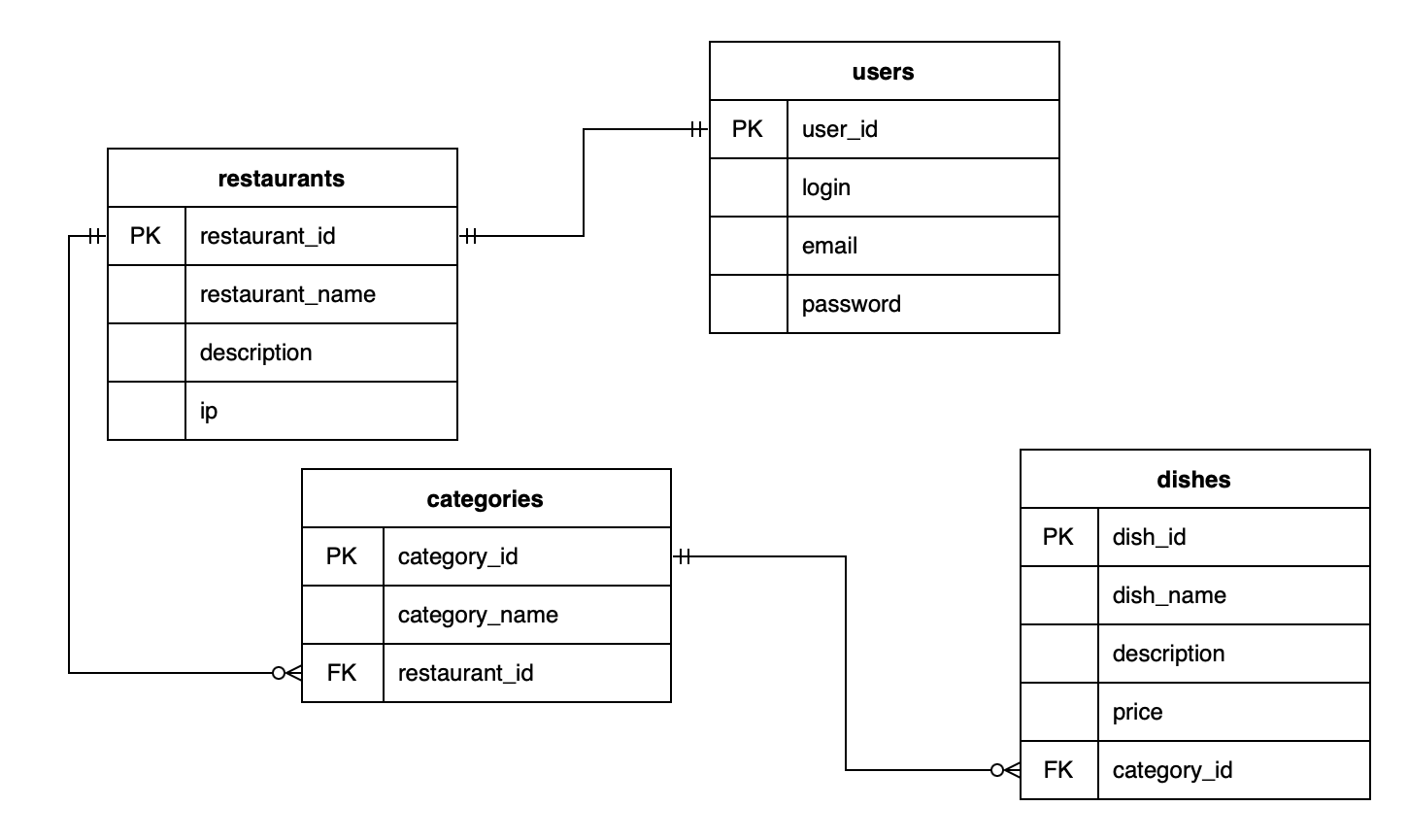


Рисунок 2.1 – Схема базы данных

Таблица users предназначена для хранения информации об администраторах.

Столбец user\_id содержит уникальный идентификатор пользователя. В login хранится логин администратора, с помощью которого он аутентифицируется. Email содержит почту администратора и password хранит пароль администратора.

Таблица restaurants хранит в себе информацию о ресторанах.

Столбец restaurant\_id содержит уникальный идентификатор ресторана, а в столбце restaurant\_name хранится название ресторана. Описание ресторана содержится в столбце description и в столбце ip хранится ip-адрес роутера ресторана.

Таблица categories содержит информацию о категориях блюд.

Столбец category\_id содержит уникальный идентификатор категории. Название категории хранится в category\_name, а restaurant\_id указывает к какому ресторану относится данная категория.

Таблица dishes предназначена для хранения информации о блюдах.

Столбец dish\_id содержит уникальный идентификатор блюда. Dish\_name хранит наименование блюда, а описание блюда содержится в столбце description. В price находится цена блюда и category\_id хранит в себе идентификатор категории, к которой относится блюдо.

Таким образом, была рассмотрена каждая из таблиц базы данных и описаны ее столбцы.

## **2.2 Проектирование структуры сервера**

При разработке сервера был использован архитектурный шаблон MVC. Архитектурный шаблон MVC является широко используемым и давно устоявшимся подходом к организации кода в приложениях. В случае Nest.js [4], этот подход обеспечивает структурирование приложения, что облегчает его понимание, разработку, расширение и поддержку. Визуальное отображение данного архитектурного шаблона показано на рисунке 2.2.

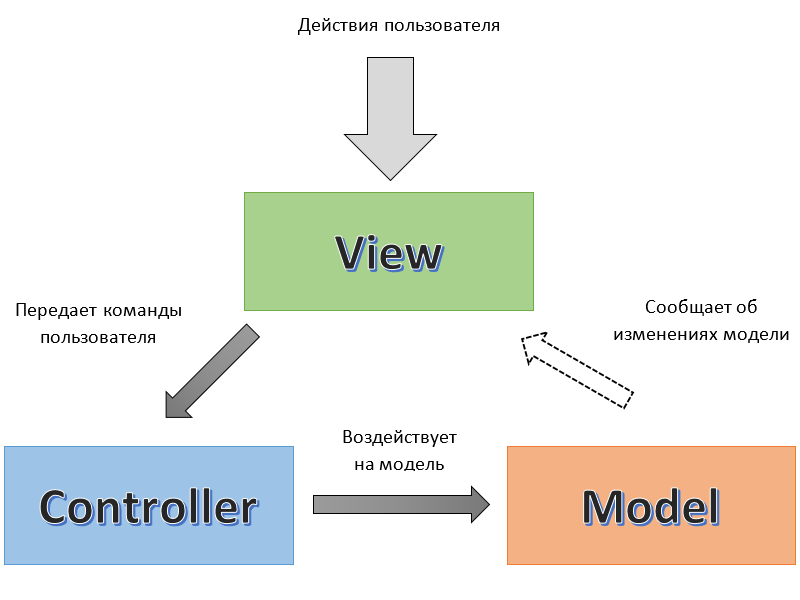


Рисунок 2.2 – Визуальное отображение архитектуры MVC

Модель представляет собой слой данных и бизнес-логики в приложении. В Nest.js, модель может быть представлена классами или сервисами, которые отвечают за работу с данными, валидацию и выполнение бизнес-логики.

В MVC-архитектуре на стороне сервера, как в Nest.js, представление обычно не представляет собой HTML-страницы, как в традиционном web-приложении. Вместо этого, это может быть формирование ответа в виде JSON или другого формата, представляющего данные, которые отправляются клиенту.

Контроллер обрабатывает входящие HTTP-запросы, взаимодействует с моделью для получения данных и возвращает ответ в виде представления. В Nest.js, контроллеры представлены классами с методами, которые обрабатывают запросы на различные маршруты.

## **2.3 Структурная схема приложения**

Структурная схема приложения — это диаграмма, которая отображает организацию и взаимосвязи различных компонентов или модулей внутри приложения. Эта схема предоставляет высокоуровневый обзор структуры приложения, обозначая ключевые элементы и связи между ними. Цель структурной схемы заключается в том, чтобы облегчить понимание организации кода и архитектурных решений. Структурная схема приложения представлена на рисунке 2.3.

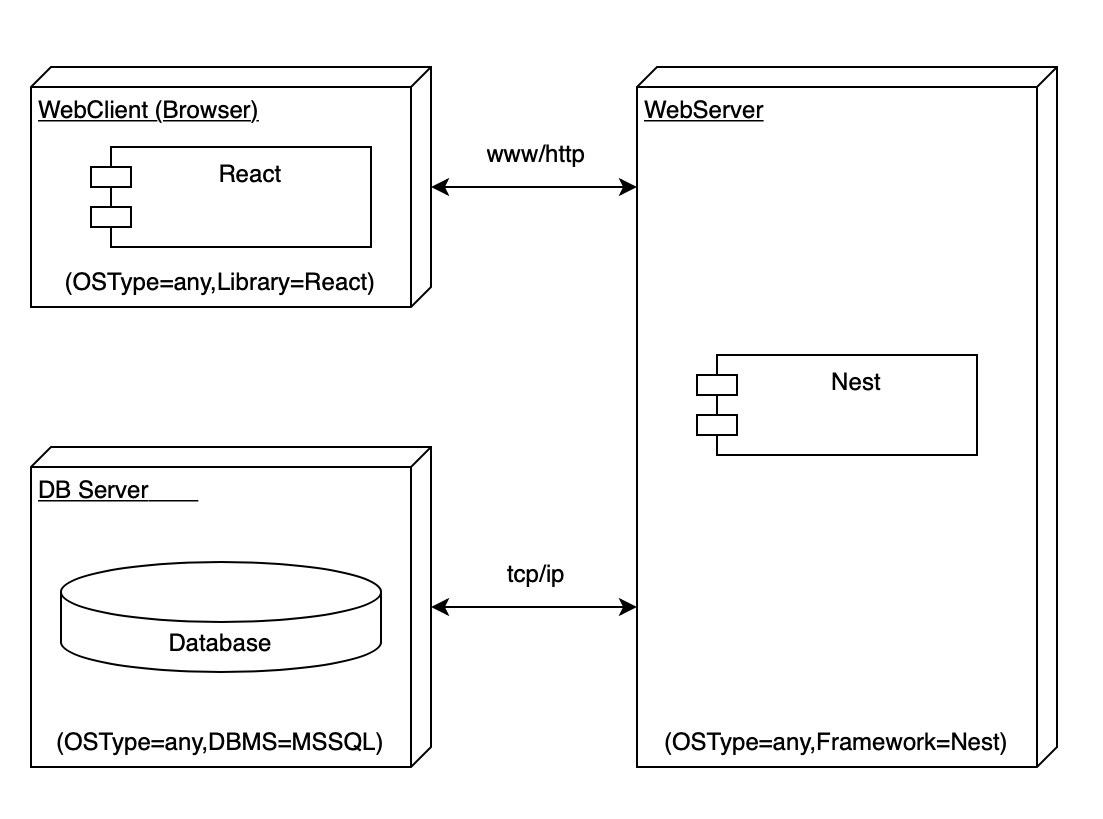


Рисунок 2.3 – Структурная схема приложения

Web-клиент отправляет HTTP-запросы к определенным маршрутам на web-сервере с использованием Fetch API. Как можно заметить, перед тем как данные дойдут от устройства – клиенту, им нужно пройти через цепочку сервисов, которые их преобразуют, обрабатывают и анализируют.

Web-сервер обрабатывает входящие запросы, вызывает соответствующие методы контроллеров. Сервисы взаимодействуют с базой данных для выполнения запросов SQL. Результаты операций базы данных возвращаются web-серверу. Web-сервер возвращает данные в формате JSON в ответ на запросы от web-клиента. Web-клиент получает данные и обрабатывает их с использованием React-компонентов.

Таким образом была рассмотрена структурная схема приложения.

## **2.4 Диаграмма вариантов использования**

Диаграмма вариантов использования – это тип диаграммы UML [3], который показывает взаимодействие между системой и ее окружением, описывая различные сценарии использования системы со стороны ее пользователей или других систем.

Web-приложение, в зависимости от роли, предоставляет пользователю функциональные возможности, которые показаны в use-case диаграмме, которая представлена в приложении А.

Для того, чтобы администратор мог использовать возможности приложения, он должен аутентифицироваться. После этого администратор может создать свой ресторан и, при необходимости, редактировать информацию о нем. Далее он может управлять категориями: создавать, редактировать и удалять. Также он может взаимодействовать с блюдами: создавать, редактировать и удалять. После заполнения меню, администратор может сгенерировать QR-код и скачать его на диск.

Гость при сканировании QR-кода попадает на страницу с меню и может сортировать блюда по алфавиту и ценам, а также осуществлять поиск по наименованием блюд.

## **2.5 Выводы по разделу**

В данном разделе осуществлено тщательное проектирование Web-приложения, включающее в себя выбор наилучшей архитектуры для разработки приложения и подробное описание структуры базы данных. В качестве системы управления базами данных (СУБД) была стратегически выбрана MSSQL, что обеспечивает надежную основу для хранения и управления данными.

Более того, в рамках проекта были разработаны и подробно описаны диаграммы вариантов использования, которые представляют собой наглядное изображение сценариев взаимодействия пользователей с приложением. Эти диаграммы обеспечивают более глубокое понимание функциональных возможностей приложения и его взаимодействия с конечными пользователями.

В дополнение к этому, разработана структурная схема web-приложения, которая визуализирует организацию ключевых компонентов и модулей внутри приложения. Такой детальный анализ и проектирование направлены на обеспечение оптимальной производительности, эффективной масштабируемости и высокого уровня функциональности web-приложения.

# 3 Разработка web-приложения

## **3.1 Реализация серверной части**

### **3.1.1 Структура проекта**

Проект был разработан с помощью фреймворка Nest. Для реализации MVC архитектуры файлы программы были распределены по соответствующим директориям и реализовали соответствующие функции. Разделение проекта на логические модули представлено на рисунке 3.1.

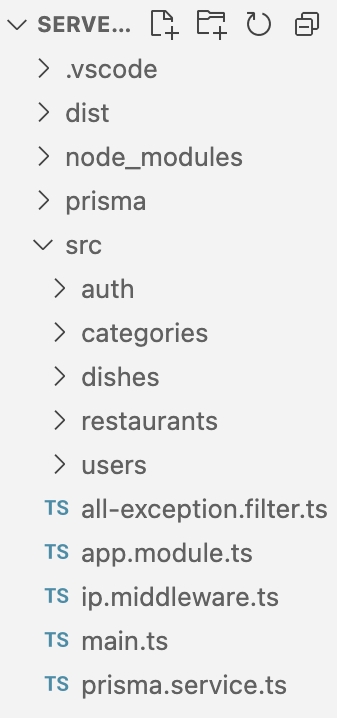


Рисунок 3.1 – Структура проекта

В директории auth находятся все сущности реализованные для аутентификации: декоратор в файле auth.guard.ts, который закрепляется за методами контроллера, требующих аутентификации, контроллер, который содержит в себе метод, обрабатывающий запрос на аутентификацию, сервис, который содержит в себе методы для генерации и верификации jwt-токена , содержимое папки представлено на рисунке 3.2.

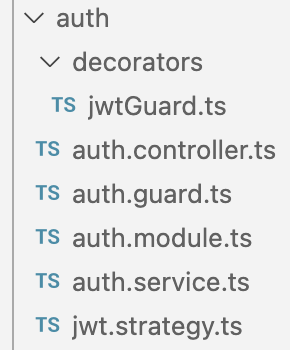
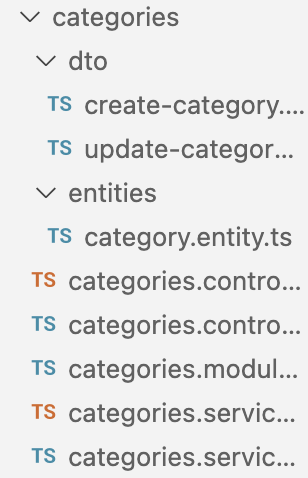


Рисунок 3.2 – Структура папки auth

Также в директории src находятся папки ресурсов: categories, dishes, restaurants. Содержимое ресурсов представлено на рисунке 3.3.



*a)*

*б)*

Рисунок 3.3 – Содержимое ресурсов: а – ресурс categories, б – ресурсы dishes и restaurants

В папке dto у ресурсов находятся классы используемые в качестве параметров методов контроллеров.

В директории entities находятся сущности, которые возвращаются из контроллеров.

Также содержатся контроллеры, которые принимают входящий запрос, обрабатывают его и формируют ответ.

В ресурсах содержатся сервисы, которые взаимодействуют с базой данных с помощью ORM Prisma.

Таким образом была рассмотрена структура серверного проекта.

### **3.1.2 Конкретные реализации**

Для того, чтобы скрыть меню за пределами ресторана, нужно считывать ip-адрес гостя и запрещать доступ к нему в том случае, если ip-адрес гостя не является приемлемым. Для решения этой проблемы был разработан промежуточный обработчик, который считывает ip-адрес гостя и запрещает ему доступ к меню в случае несовпадения ip-адреса с разрешенным, пример реализации промежуточного обработчика представлен в листинге 3.1.

@Injectable()

export class IpMiddleware implements NestMiddleware {

constructor(private readonly prisma: PrismaService) {}

async use(req: Request, res: Response, next: NextFunction) {

const ip = req.ip.split(':')[3].split('.').slice(0, 3).join('.');

const restaurantId = await this.prisma.restaurants.findFirst({

where: {

ip: {

contains: ip,

},

},

select: {

restaurant\_id: true,

},

});

if (!restaurantId)

throw new NotFoundException('Restaurant with this ip not found');

req['restaurant\_id'] = restaurantId.restaurant\_id;

next();

}

}

Листинг 3.1 – Реализация промежуточного обработчика IpMiddleware

Перед поступлением запроса в контроллер, он обрабатывается этим промежуточным обработчиком. Из объекта req извлекается ip-адрес клиента, сделавшего запрос, и берется необходимая подстрока для отправки запроса в базу данных с целью нахождения ресторана с данным ip-адресом, если ресторана с данным ip-адресом не найдено, то генерируется исключение, на которое реагирует глобальный фильтр исключений.

Для обработки ошибок приложения был разработан глобальный фильтр, обработчик исключительных ситуаций. Все исключения, возникающие в приложении, будут обработаны данным фильтром перед отправкой ответа клиенту. Реализация глобального фильтра представлена в листинге 3.2.

@Catch()

export class HttpExceptionFilter implements ExceptionFilter {

catch(exception: any, host: ArgumentsHost) {

const ctx = host.switchToHttp();

const response = ctx.getResponse<Response>();

if (!(exception instanceof HttpException)) {

response.status(HttpStatus.BAD\_REQUEST).json({

statusCode: HttpStatus.BAD\_REQUEST,

message: exception.message,

});

} else {

const status = exception.getStatus();

response.status(status).json({

statusCode: status,

message: exception.message,

});

}

}

}

Листинг 3.2 – Реализация глобального фильтра HttpExceptionFilter

Для взаимодействия с блюдами был реализован контроллер «DishesController». В котором находятся методы: findAll – для получение всех блюд в указанной категории, create – для создания нового блюда, update – для редактирования информации об указанном блюде и remove – для удаления указанного блюда. Реализация контроллера представлена в листинге 3.3.

@Controller('api/dishes')

export class DishesController {

constructor(private readonly dishesService: DishesService) {}

@Post() @UseGuards(AuthGuard)

async create(@Body() createDishDto: CreateDishDto) {

return await this.dishesService.create(createDishDto);

}

@Get(':id')

async findAll(@Param('id') id: string, @Query('like') like: string) {

return await this.dishesService.findAll(+id, like);

}

@Patch(':id') @UseGuards(AuthGuard)

async update(@Param('id') id: string, @Body() updateDishDto: UpdateDishDto) {

return await this.dishesService.update(+id, updateDishDto);

}

@Delete(':id') @UseGuards(AuthGuard)

async remove(@Param('id') id: string) {

return await this.dishesService.remove(+id);

}

}

Листинг 3.3 – Реализация контроллера DishesController

Этот контроллер в конструкторе инициализирует «DishesService». Этот сервис предназначен для управления блюдами в базе данных, посредством методов «PrismaService». Реализация сервиса представлена в листинге 3.4.

@Injectable()

export class DishesService {

constructor(private readonly prisma: PrismaService) {}

async create(createDishDto: CreateDishDto) {

return await this.prisma.dishes.create({

data: {...createDishDto,},

});

}

async findAll(id: number, like: string) {

return await this.prisma.dishes.findMany({

where: {

category\_id: id,

dish\_name: {contains: like,},

},

});

}

async update(id: number, updateDishDto: UpdateDishDto) {

return await this.prisma.dishes.update({

where: {

dish\_id: id,

},

data: {...updateDishDto,},

});

}

async remove(id: number) {

return await this.prisma.dishes.delete({

where: {dish\_id: id,},

});

}

}

Листинг 3.4 – Реализация сервиса DishesService

В данном сервисе реализованы следующие методы: create, который вызывает метод create у объекта класса «PrismaService», принимает параметром «CreateDishDto» и отправляет запрос на создание блюда в базе данных; findAll, который вызывает метод findMany у объекта класса «PrismaService», принимает параметрами id и like и отправляет запрос на извлечения блюд из базы данных в соответствии с фильтром, переданным параметром like; update – вызывает метод update у объекта класса «PrismaService», принимает параметрами id и «UpdateDishDto» и отправляет запрос в базу данных на изменение данных о блюде с идентификатором, переданным параметром id; remove – вызывает метод delete у объекта класса «PrismaService», принимает параметром id и отправляет запрос в базу данных на удаление блюда с идентификатором, переданным параметром id.

Главным модулем приложения является «AppModule». В нем подключаются все зависимости приложения: другие модули, провайдеры, фильтры и промежуточные обработчики.

@Module({

imports: [

AuthModule,

UsersModule,

RestaurantsModule,

CategoriesModule,

DishesModule,

],

controllers: [],

providers: [

{

provide: APP\_FILTER,

useClass: HttpExceptionFilter,

},

JwtService,

PrismaService,

],

})

export class AppModule implements NestModule {

configure(consumer: MiddlewareConsumer) {

consumer

.apply(IpMiddleware)

.exclude(

{ path: 'auth/login', method: RequestMethod.ALL },

{ path: 'restaurants', method: RequestMethod.POST },

)

.forRoutes('\*');

}

}

Листинг 3.5 – Реализация модуля AppModule

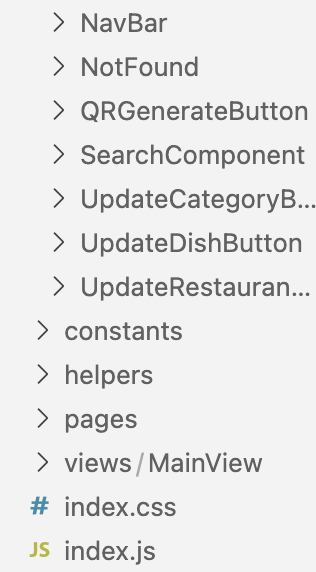
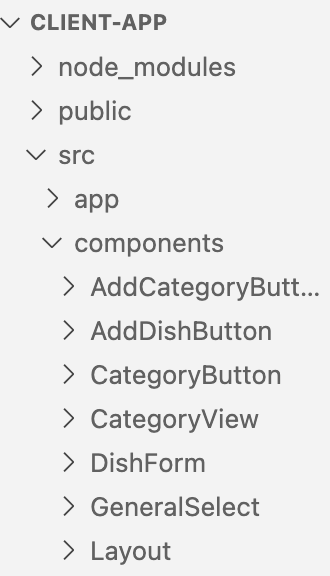
Таким образом была рассмотрена серверная часть приложения.

## **3.2 Реализация клиентской части**

Клиентская часть реализована с использованием библиотеки React [5]. React – это библиотека JavaScript, которая используется для создания пользовательских интерфейсов (UI) web-приложений. Она позволяет разработчикам создавать динамические и масштабируемые приложения с помощью компонентного подхода.

### **3.2.1 Структура проекта**

Проект был реализован с помощью JavaScript библиотеки React. При реализации клиентской части, использовался модульный подход, пользовательский интерфейс разбит на отдельные, переиспользуемые компоненты, которые находятся в папке components. Структура react-проекта client-app представлена на рисунке 3.6.



*б)*

*a)*

Рисунок 3.4 – Структура проекта client-app: а – первая часть структуры проекта (app, components), б – вторая часть структуры проекта (constants, helpers, pages, views)

В директории components находятся все react-компоненты, используемые в приложении, в constants находятся постоянные переменные приложения, в папке helpers находятся функции для отправки запросов на сервер, страницы хранятся в директории pages и в папке views находятся представления приложения.

### **3.2.2 Компоненты**

Для того, чтобы пользователь мог переходить на различные страницы приложения в файле App.js был настроен маршрутизатор с использованием пакета react-router-dom. При реализации приложения была проблема генерации динамических путей, при создании категории администратором, для удобной маршрутизации. В ходе разработки было найдено решение данной проблемы: для этого в компоненте App отправляется запрос в базу данных на извлечение всех категорий и создаются пути в соответствии с названиями категорий, что позволило создать динамическую навигационную панель, обновляющуюся в зависимости от полученных категорий. Реализация компонента App приведена в приложении Б.

Для навигационной панели был разработан компонент NavBar, который в свойствах принимает categories, массив с категориями из базы данных, и fetchCategories – функцию, которая содержит запрос на получение всех категорий. При добавлении новой категории данные из базы данных будут динамически подтягиваться и обновлять панель. Реализация компонента приведена в листинге 3.6.

const NavBar = ({ categories, fetchCategories }) => {

const navigate = useNavigate();

return (

<div className={styles.navBarContainer}>

<Button variant={window.location.href.includes('main') ? "warning" : "primary"} onClick={() => navigate("/main")}>

Main

</Button>

{categories?.length > 0 &&

categories.map((item, index) => (

<CategoryButton

name={item.category\_name}

key={index}

path={`/${item.category\_name.toLowerCase()}`}

selected={window.location.href.includes(item.category\_name.toLowerCase())}

/>

))}

{categories?.length < 9 && localStorage.getItem("token") && (

<AddCategoryButton fetchCategories={fetchCategories} />

)}

</div>

);

};

export default NavBar;

Листинг 3.6 – Реализация компонента NavBar

Отображение содержимого категории реализовано в компоненте CategoryView. Компонент принимает два свойства: category и fetchCategories, категорию и функцию на получение категорий от сервера соответственно. В CategoryView реализована сортировка, также он содержит компоненты для поиска, сортировки, добавления блюда, редактирования категории, удаления категории и отображает все блюда, которые принадлежат выбранной категории. Поиск осуществлен с помощью комбинации хуков useState и useEffect: сеттер setSearchValue передается в компонент поиска SearchComponent и там изменяется в зависимости от введенного значения. Тем временем выполнение useEffect в котором определен запрос на получение всех блюд от сервера из базы данных зависит от состояния SeacrhValue. Сортировка, в свою очередь, основана на стандартном методе массива sort, при получении от сервера массива с блюдами данной категории, он сортируется в зависимости от выбранного пользователем типа сортировки. Реализация компонента представлена в приложении В.

Кнопка навигационной панели, содержащая название категории реализована в компоненте CategoryButton. В свойствах компонент принимает name, название категории, path, для формировании динамического пути категории, и selected для отображения нажатия кнопки. Реализация компонента приведена в листинге 3.7.

const CategoryButton = ({ name, path, selected }) => {

const navigate = useNavigate();

return (

<Button

variant={selected ? "warning" : "primary"} onClick={() => navigate(path)}>

{name}

</Button>

);

};

export default CategoryButton;

Листинг 3.7 – Реализация компонента CategoryButton

Компонент Button взят из библиотеки react-bootstrap, стили кнопки динамически изменяются по нажатию, с помощью атрибута variant.

## **3.3 Выводы по разделу**

В процессе разработки было создано web-приложение, полностью соответствующее установленным в предшествующих разделах требованиям. Для серверной части проекта была подробно изложена структура, а также осуществлена разработка основных компонентов, необходимых для обеспечения полноценного функционирования web-приложения. Дополнительно к этому, внимание уделено деталям взаимодействия серверной части с клиентской.

В контексте клиентской части проекта представлено детальное описание структуры, включая разработку компонентов, эффективно взаимодействующих между собой и осуществляющих запросы на сервер для получения необходимых данных. Сфокусировавшись на максимальной интеграции компонентов, обеспечивается гармоничное взаимодействие всей системы.

Особое внимание также уделено подробному описанию реализации ключевых функций web-приложения. Здесь освещены технические детали взаимодействия между сервером и клиентской частью, что обеспечивает стабильную и эффективную работу всех функций приложения. Таким образом, в результате проведенной работы было успешно создано web-приложение, полностью соответствующее заявленным требованиям и обеспечивающее высокий уровень функциональности.

# 4 Тестирование web-приложения

### **4.1 Ручное тестирование**

Для проверки взаимодействия клиентской и серверной части приложения было проведено ручное тестирование. Будут предоставлены скриншоты с описанием проведенных тестов.

При попытке входа под несуществующим пользователем выводится модальное окно с сообщением, приведенным на рисунке 4.1.

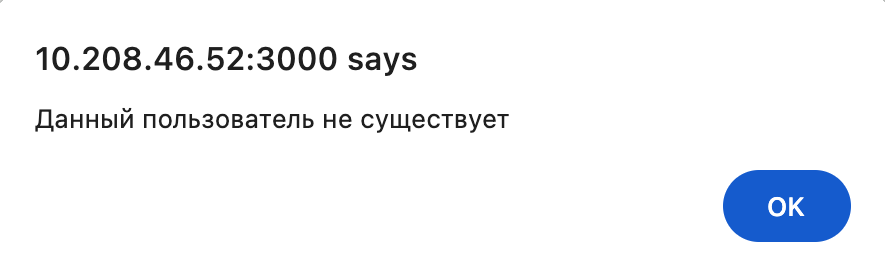


Рисунок 4.1 – Результат попытки входа под несуществующим пользователем

У администратора есть возможность создать ресторан, при этом ему нужно заполнить необходимые поля, пока они не будут заполнены, он не сможет сохранить ресторан, пример приведен на рисунке 4.2.



Рисунок 4.2 – Результат ввода не всех полей в форму для создания ресторана

После создания ресторана пользователь попадает на главную страницу, там он может редактировать информацию о ресторане, для этого ему нужно заполнить поля формы, иначе он не сможет сохранить изменения, пример приведен на рисунке 4.3.

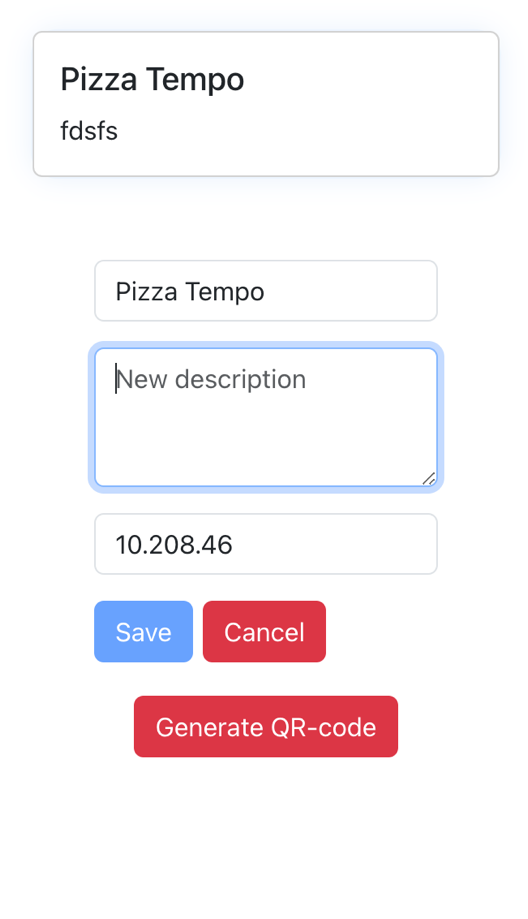
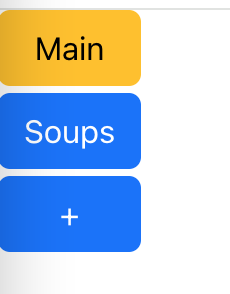
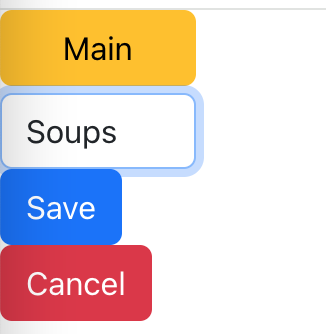


Рисунок 4.3 – Результат ввода не всех полей в форму редактирования информации о ресторане

Поля формы заполняются автоматически уже существующими данными, чтобы пользователь мог изменить описание ресторана, а не писать его с нуля, если оно содержит много информации.

Для создания новой категории нужно нажать на плюс и заполнить появившееся поле. Кнопка сохранения Результат позитивного теста на добавление новой категории, приведен на рисунке 4.4.



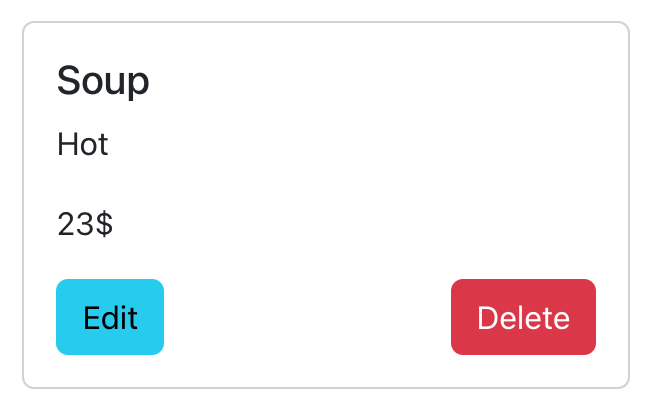
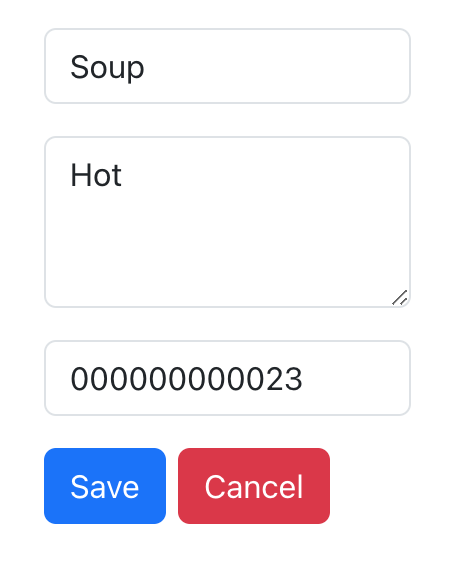
*б)*

*a)*

Рисунок 4.4 – Позитивный тест добавления новой группы: а – заполнение формы, б – результат работы

Можно увидеть, что категория была добавлена успешно, об этом свидетельствует появившаяся категория.

Также была протестирована ситуация, когда администратор пытается добавить блюдо написав много нулей перед самой ценой. Результат работы приложения в этом случае приведен на рисунке 4.5.



*б)*

*a)*

Рисунок 4.5 – Попытка ввода множества нулей перед ценой: а – заполнение формы, б – результат работы

Отрицательные числа и символы нельзя ввести в поле цены. Таким образом мы предотвращаем ввод некорректных данных.

## **4.2 Выводы по разделу**

В данном разделе подробно описывается процесс ручного тестирования, проведенного для разработанного web-приложения. В рамках этой проверки были освещены ключевые функциональности административной панели, такие как создание и изменение ресторана, категорий и блюд, включая проверку валидации полей.

Важно отметить, что тестирование не ограничивалось только финальным этапом работы над приложением. Еще на стадии разработки проводились тщательные проверки каждого созданного компонента, а также их взаимодействия между собой. Это позволило удостовериться в корректности работы приложения на более ранних этапах.

С учетом всех проведенных проверок и тестирований можно утверждать, что приложение функционирует стабильно и корректно. Полученные результаты подтверждают успешную работу каждого компонента, а также эффективное взаимодействие между ними. Таким образом, тестирование приложения в различных этапах разработки обеспечивает высокий уровень его качества и надежности.

# 5 Руководство пользователя

## **5.1 Методика использования приложения**

Первое, что встречает администратор, – страница для входа в приложение, приведена на рисунке 5.1.

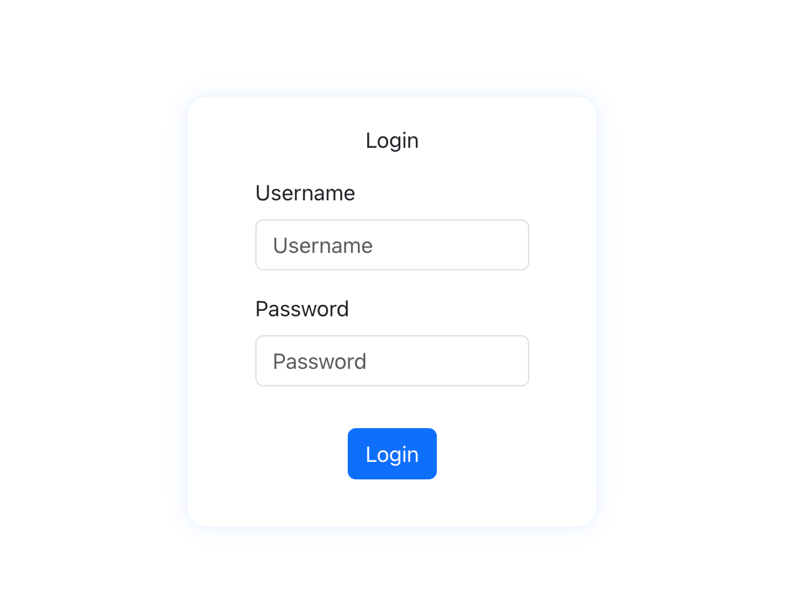


Рисунок 5.1 – Страница входа в приложение

Чтобы администратор попал в систему, ему необходимо ввести свой логин и пароль, которые были выданы владельцем приложения. После входа в приложение, пользователь попадает на приветственную страницу, на которой он должен создать свой ресторан, заполнив необходимые поля формы: название ресторана, описание и ip-адрес ресторана, этот ip-адрес может быть адресом сети Wi-Fi, к которой имеют доступ гости ресторана. Исходное состояние страницы представлено на рисунке 5.2.

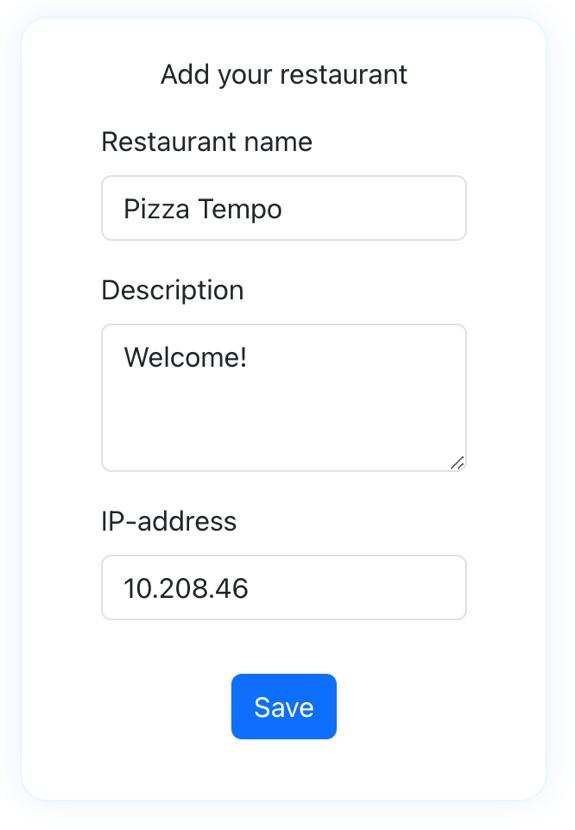


Рисунок 5.2 – Приветственная страница

Ip-адрес заполняется автоматически, считывая ip-адрес, с которого зашел администратор. Это сделано дабы избежать недопонимания у неопытного пользователя. После нажатия на кнопку save, администратор попадает на главную станицу меню, на которой размещается информация о ресторане, отображение главной страницы представлено на рисунке 5.3.

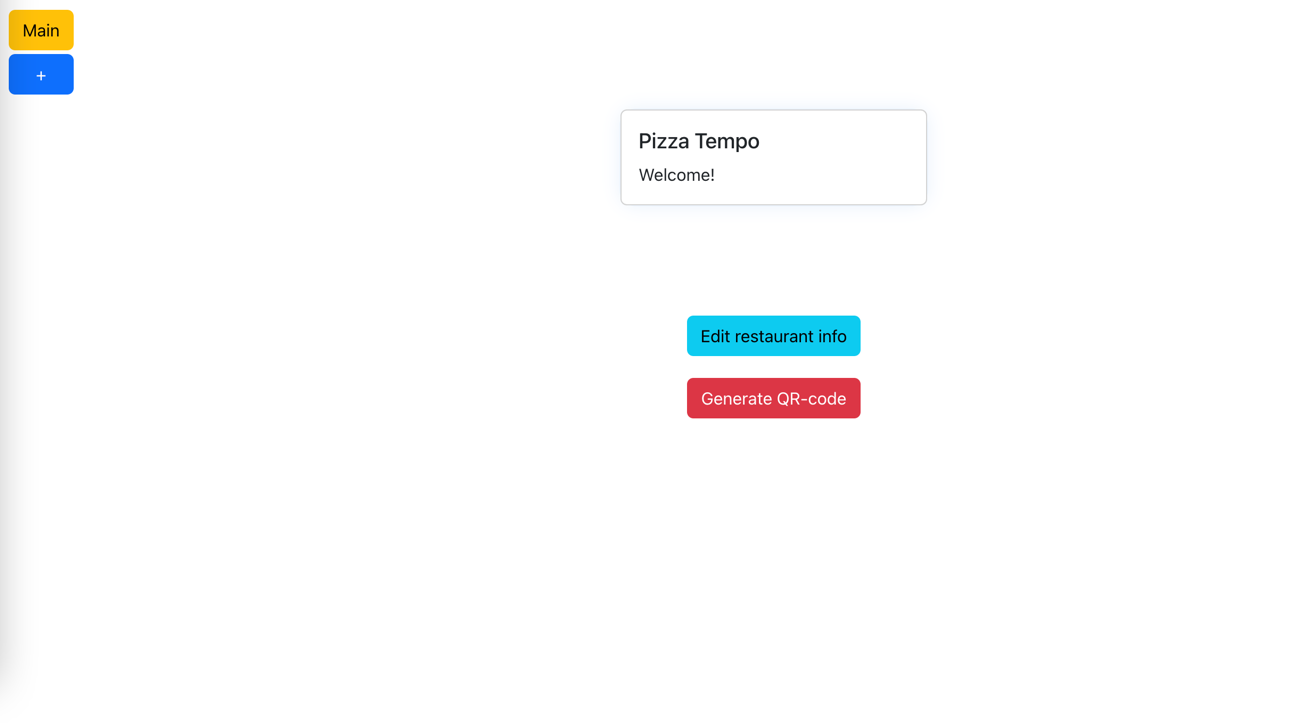


Рисунок 5.3 – Отображение главной страницы

На главной странице есть доступ к навигационной панели, а также есть возможность редактировать информацию о ресторане и сгенерировать QR-код. Демонстрация главной страницы после изменения информации о ресторане на рисунке 5.4.

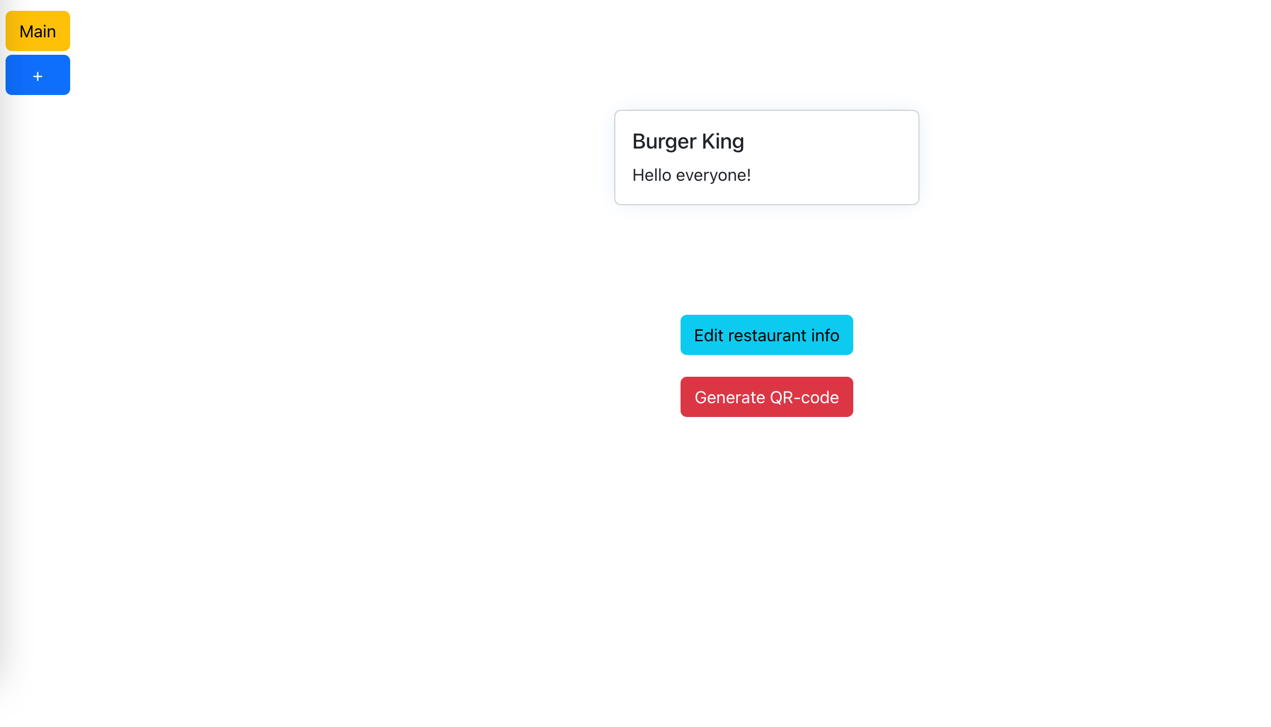


Рисунок 5.4 – Демонстрация главной страницы после изменения информации о ресторане

Для создания категории администратору нужно нажать на плюс в навигационной панели и ввести название категории, после этого кликнув по категории он может попасть на страницу этой категории, страница созданной категории представлена на рисунке 5.5.

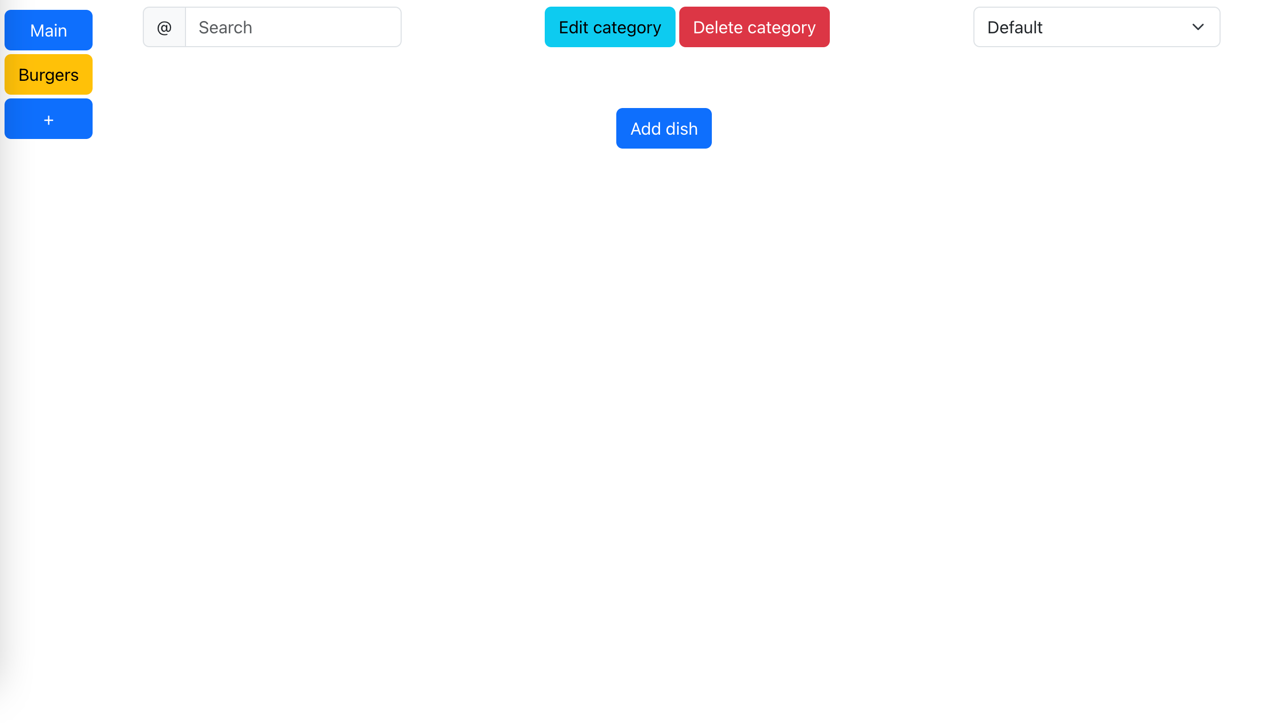


Рисунок 5.5 – Страница категории

На странице категории администратор может редактировать название категории, удалить категорию, добавить блюдо, редактировать блюдо, удалить блюдо, сортировать блюда по алфавиту и по ценам, а также осуществлять динамический поиск по наименованиям блюд. Для создания блюда нужно нажать на кнопку добавления блюда и заполнить необходимые поля. Пример добавленного блюда приведен на рисунке 5.6.

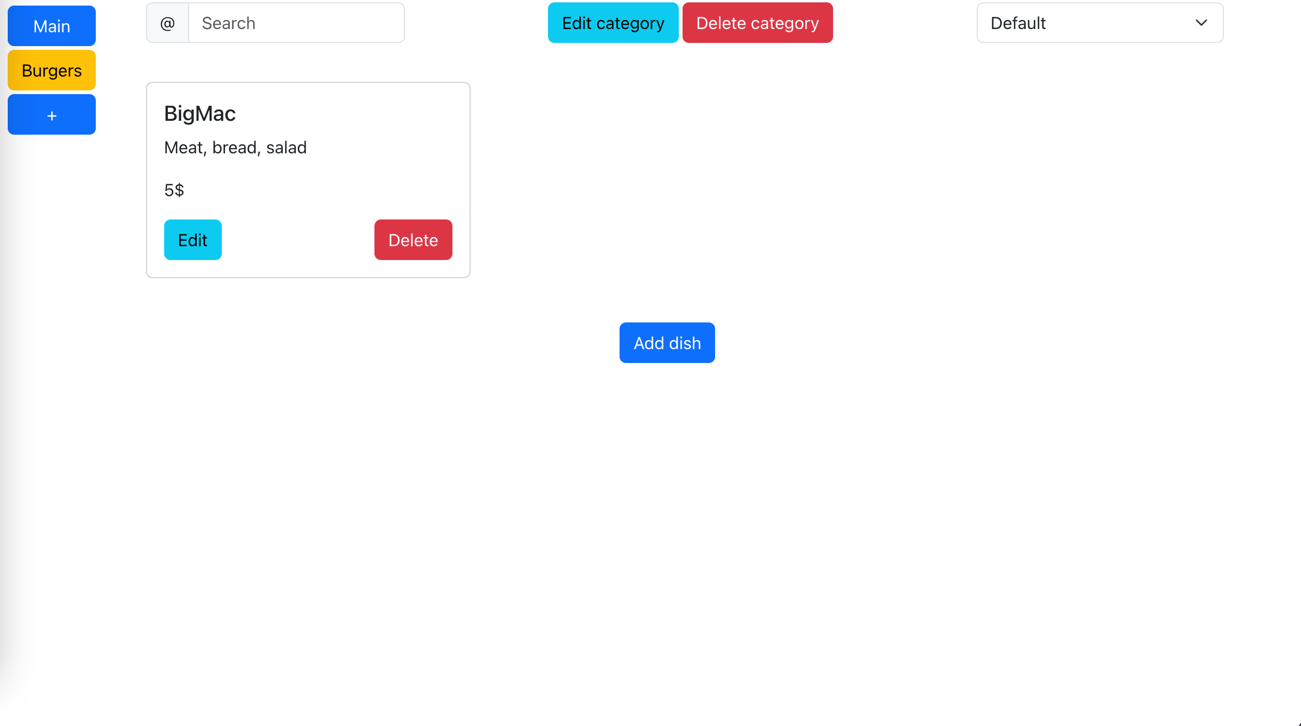


Рисунок 5.6 – Пример добавленного блюда

Администратор может изменить информацию о блюде и удалить блюдо. Пример измененного блюда представлен на рисунке 5.7.

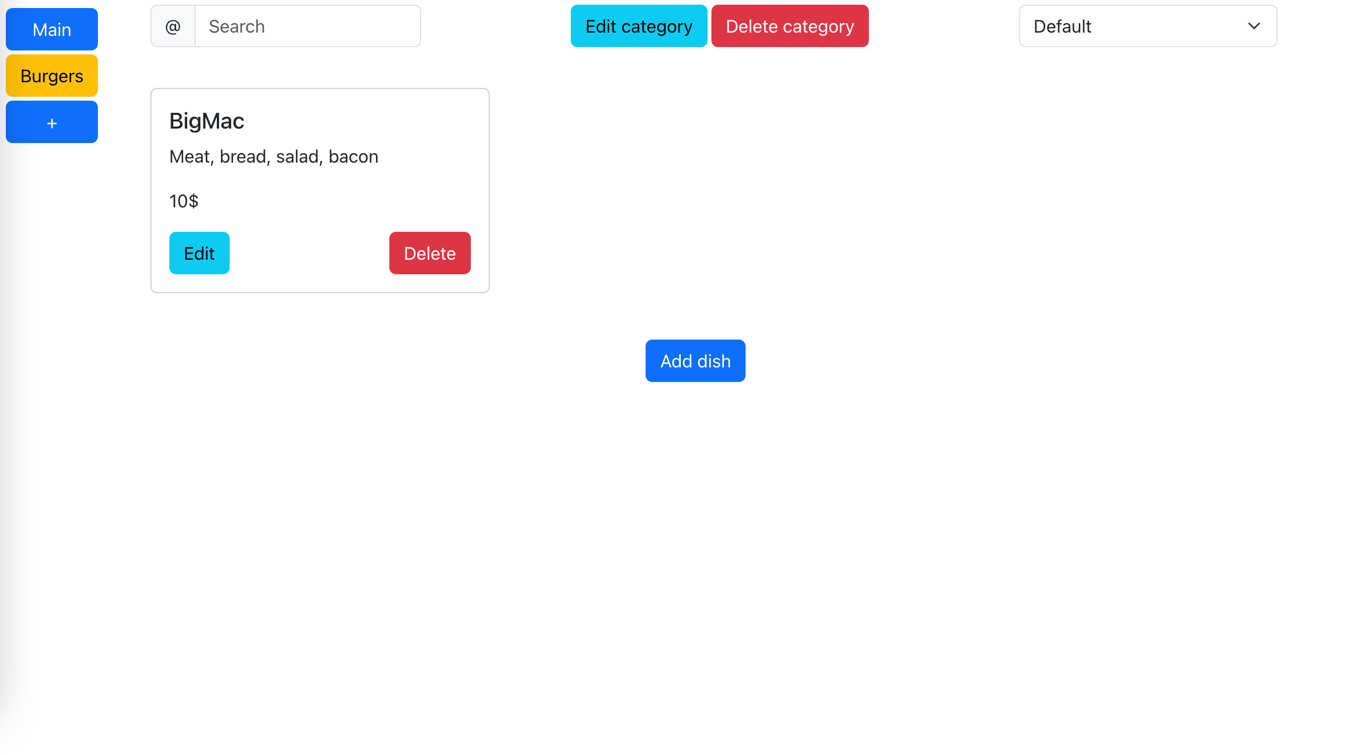


Рисунок 5.7 – Пример измененного блюда

Поиск является динамическим и срабатывает при вводе каждого символа. Блюда, которые не соответствуют введенному шаблону, не отображаются. Результат поиска представлен на рисунке 5.8.

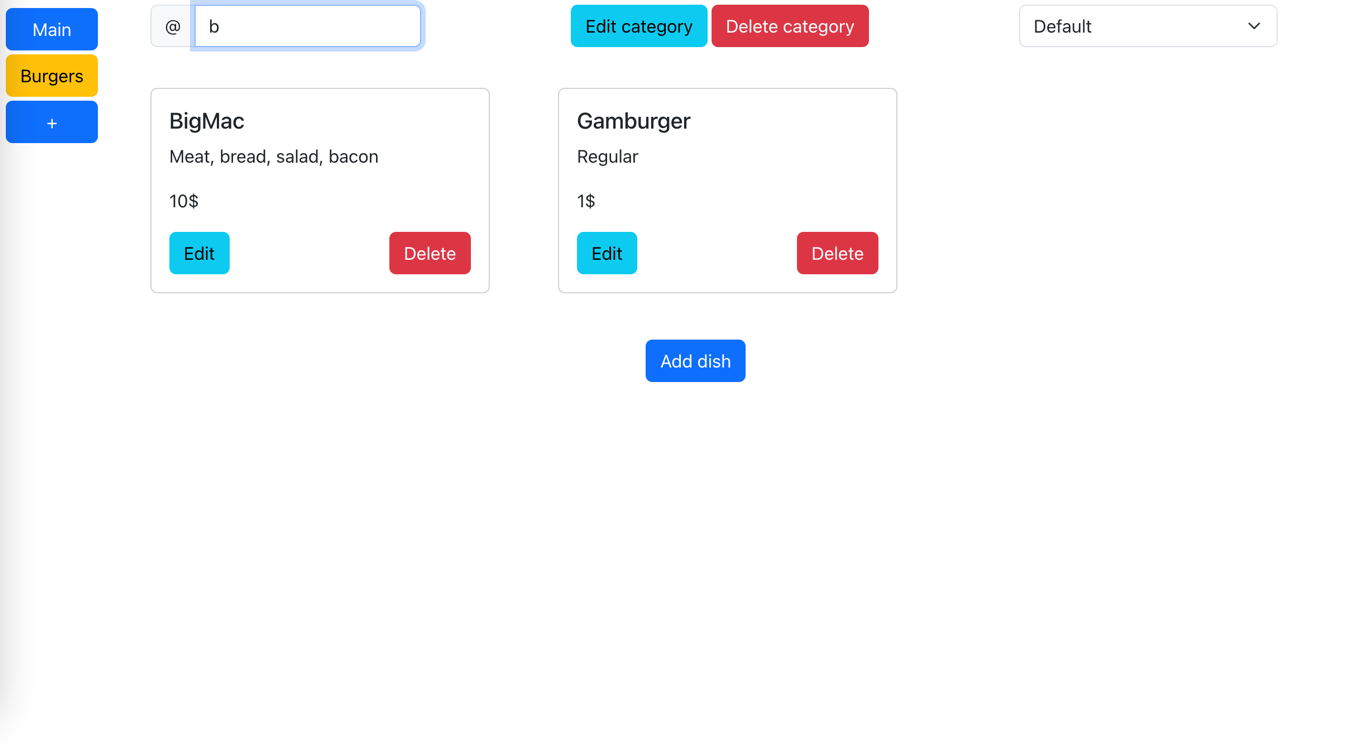


Рисунок 5.8 – Результат поиска

Сортировка возможна по четырем позициям: по убыванию цен, по возрастанию цен, по алфавиту, по алфавиту в обратном порядке. Пример сортировки по возрастанию цен представлен на рисунке 5.9.

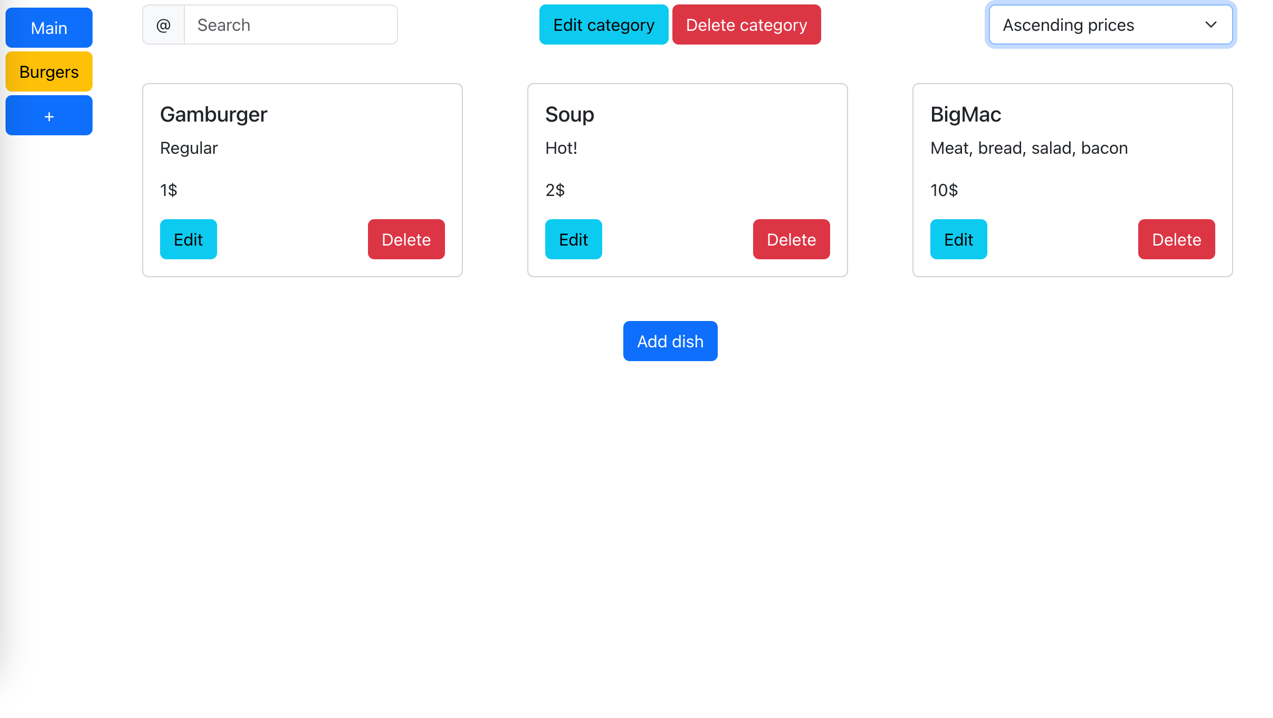


Рисунок 5.9 – Пример сортировки по возрастанию цен

Генерация QR-кода происходит на главной странице по нажатию на кнопку Generate QR-code. Затем на странице появляется сам QR-код и кнопка по нажатию на которую происходит сохранение его на диске. Пример приведен на рисунке 5.10.



Рисунок 5.10 – Пример генерации QR-кода

Далее QR-код может быть обработан в фоторедакторе в сочетании со стилистикой заведения и напечатан.

## **5.2 Выводы по разделу**

В разделе представлены основные моменты работы с web-приложением. Были приведены скриншоты самого приложения для большей наглядности использования с подробным описанием алгоритмов взаимодействия пользователя с web-приложением. Описано то, что пользователь может наблюдать на различных страницах и функционал, который ему доступен.

Интерфейс приложения достаточно прост, поэтому у пользователя не должно возникнуть трудностей с его эксплуатацией.

# Заключение

В данном курсовом проекте были рассмотрены основные аналоги приложения, были рассмотрены их преимущества и недостатки, изучены возможности рассматриваемых web-приложений.

При проектировании web-приложения была спроектирована структура базы данных, выбрана архитектура для web-приложения, разработаны диаграмма развертывания приложения и структурная схема приложения.

Web-приложение выполнено с использованием асинхронного программирования. Оно реализовано на языке JavaScript на платформе Node.js, что делает его кроссплатформенным. В качестве системы управления базой данных использована MSSQL. Для разработки клиентской части использовалась библиотека React совместно с библиотекой react-bootstrap.

Разработанное web-приложение предоставляет следующий функционал:

‒ поддержку ролей: администратор, гость;

‒ возможность управления содержимым меню администратором (добавление, редактирование, удаление конкретных блюд и категорий блюд);

‒ генерацию QR-кода администратором;

‒ возможность просмотра меню гостем при сканировании QR-кода;

‒ поиск по наименованием блюд гостем и администратором;

‒ сортировка блюд по алфавиту и ценам гостем и администратором.

Было проведено ручное тестирование, показавшее, что приложение работает правильно и соблюдает все ранее установленные требования.

При составлении руководства пользователя была подробно описана работа с web-приложением.

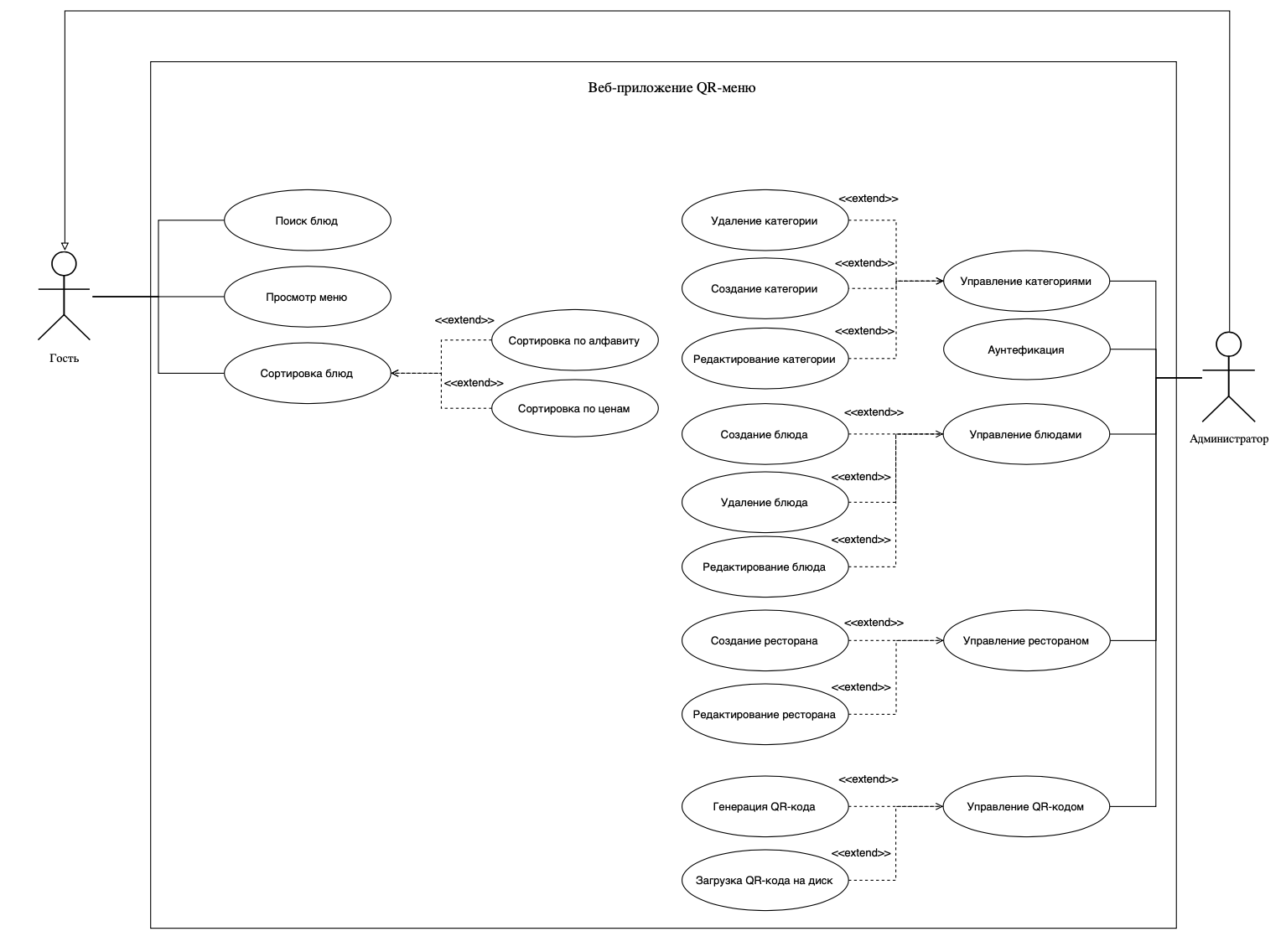
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно. Таким образом, поставленная цель достигнута, а задачи выполнены в полном объеме.

# Список используемых источников

1. Relax.by [Электронный ресурс] / info.relax.by/menu. – Режим доступа: https://info.relax.by/menu. – Дата доступа: 11.10.2023.
2. Foodeon [Электронный ресурс] / foodeon.com. – Режим доступа: https://foodeon.com/ru/menu. – Дата доступа: 12.10.2023.
3. Фаулер, М. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. / М. Фаулер. – 3-е изд. – СПб: Символ-Плюс, 2004. – 192с.
4. Nest documentation [Электронный ресурс] / nestjs.com. – Режим доступа: https://nestjs.com. – Дата доступа: 25.10.2023.
5. Бэнкс, А. React и Redux. Функциональная веб-разработка. / А. Бэнкс, Е. Порселло. – СПб: Питер, 2020. – 336с.

# ПРИЛОЖЕНИЕ А

Диаграмма вариантов использования



# ПРИЛОЖЕНИЕ Б

Реализация компонента App

function App() {

const [categories, setCategories] = useState([]);

async function fetchCategories() {

const response = await getCategories();

const body = await responseHandler(response);

setCategories(body);

}

useEffect(() => {

fetchCategories();

}, []);

return (

<>

<Router>

<Routes>

<Route element={<LoginPage />} path="/login" />

<Route

element={<WelcomePage fetchCategories={fetchCategories}/>}

path="/welcome"

/>

<Route element={ <Layout

categories={categories}

fetchCategories={fetchCategories}

/>}

path="/"

>

<Route index element={<Navigate to="/main" />} />

<Route

element={<MainView fetchCategories={fetchCategories}/>}

path="main"

/>

{categories?.length > 0 &&

categories.map((item, index) => (

<Route

key={index}

element={

<CategoryView

category={item}

fetchCategories={fetchCategories}

/>}

path={`/${item.category\_name.toLowerCase()}`}

/>

))}

</Route>

<Route path="\*" element={<NotFound />} />

</Routes>

</Router>

</>);

}

export default App;

# ПРИЛОЖЕНИЕ В

Реализация компонента CategoryView

const CategoryView = ({ category, fetchCategories }) => {

const navigate = useNavigate();

const [dishes, setDishes] = useState([]);

const defaultDishes = useRef([]);

const [searchValue, setSearchValue] = useState("");

const [orderValue, setOrderValue] = useState("");

const handleDeleteCategory = async () => {

const response = await deleteCategory(category.category\_id);

responseHandler(response);

fetchCategories();

navigate("/main");

};

async function fetchDishes() {

const response = await getDishes(category.category\_id, searchValue);

const body = await responseHandler(response);

setDishes(body);

defaultDishes.current = body;

}

useEffect(() => {

fetchDishes();

}, [category, searchValue]);

useEffect(() => {

setDishes(sortDishes());

}, [orderValue]);

function compareByName(a, b) {

const nameA = a.dish\_name.toUpperCase();

const nameB = b.dish\_name.toUpperCase();

if (nameA < nameB) {

return -1;

}

if (nameA > nameB) {

return 1;

}

return 0;

}

function compareByPrice(a, b) {

const priceA = parseFloat(a.price);

const priceB = parseFloat(b.price);

if (priceA < priceB) {

return -1;

}

if (priceA > priceB) {

return 1;

}

return 0;

}

const sortDishes = () => {

let result = [...dishes];

switch (orderValue) {

case "1":

result = result.sort(compareByPrice);

break;

case "2":

result = result.sort(compareByPrice).reverse();

break;

case "3":

result = result.sort(compareByName);

break;

case "4":

result = result.sort(compareByName).reverse();

break;

default:

result = defaultDishes.current;

break;

}

return result;

};

return (

<div className={styles.categoryViewContainer}>

<div className={styles.contentContainer}>

<div className={styles.functionalButtons}>

<div className={styles.searchWidthFix}>

<SearchComponent

searchValue={searchValue}

setSearchValue={setSearchValue}

/>

</div>

{localStorage.getItem("token") && (

<div className={styles.categoriesButtons}>

<div className={styles.updateButton}>

<UpdateCategoryButton

fetchCategories={fetchCategories}

category={category}

/>

</div>

<div className={styles.deleteButtton}>

<Button variant="danger" onClick={handleDeleteCategory}>

Delete category

</Button>

</div>

</div>

)}

<div className={styles.selectWidthFix}>

<GeneralSelect

orderValue={orderValue}

setOrderValue={setOrderValue}

/>

</div>

</div>

<div className={styles.dishesContainer}>

{dishes?.length > 0 &&

dishes.map((item, index) => (

<DishForm key={index} dish={item} fetchDishes={fetchDishes} />

))}

</div>

{localStorage.getItem("token") && (

<div className={styles.addDishContainer}>

<AddDishButton

categoryId={category.category\_id}

fetchDishes={fetchDishes}

/>

</div>

)}

</div>

</div>

);

};

export default CategoryView;