



Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК Информатика и управление

КАФЕДРА ИУК4 Программное обеспечение ЭВМ, информационные технологии

КУРСОВАЯ РАБОТА

НА ТЕМУ:

Проектирование системы доставки готовой еды

Студент группы ИУК4-62Б

(подпись, дата)

М.А. Валявкин

(И.О. Фамилия)

Руководитель курсовой работы

(подпись, дата)

С.А. Глебов

(И.О. Фамилия)

Калуга, 2025

Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного
учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУК4
(Ю.Е. Гагарин)
« 10 » марта 2025г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине Проектирование систем хранения и обработки данных
Студент группы ИУК4-62Б Валявкин Максим Александрович
(фамилия, имя, отчество)

Тема курсовой работы Проектирование системы доставки готовой еды

Направленность КР учебная

Источник тематики кафедра ИУК4

Задание

Спроектировать структуры базы данных;

Разработать системное приложение;

Оформить графическую часть работы.

Оформление курсовой работы

Расчетно-пояснительная записка на 45 листах формата А4.

Перечень графического материала КР (плакаты, схемы, чертежи и т.п.):

– Структура базы данных – 1 лист формата А3;

– Визуальная часть приложения – 1 лист формата А3;

Дата выдачи задания « 10 » марта 2025 г.

Руководитель

10.03.2025

(подпись, дата)

С.А. Глебов

(И.О. Фамилия)

Студент

10.03.2025

(подпись, дата)

М.А. Валявкин

(И.О. Фамилия)

КАЛЕНДАРНЫЙ ПЛАН на выполнение курсовой работы

по дисциплине Проектирование систем хранения и обработки данных

Студент группы ИУК4-62Б Валявкин Максим Александрович
(фамилия, имя, отчество)

Тема курсовой работы Проектирование системы доставки готовой еды

№	Наименование этапов	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Руководитель	Куратор
1	Задание на выполнение	1-я нед.	1-я нед.		
2	Проектирование базы данных	4-я нед.	4-я нед.		
3	Разработка приложения и окончательное оформление графической части и расчетно-пояснительной записки	7-я нед.	7-я нед.		
4	Защита	11-я нед.	11-я нед.		

Студент _____ 10.03.2025г.
(подпись, дата)

Руководитель _____ 10.03.2025г.
(подпись, дата)

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ	4
ВВЕДЕНИЕ	5
1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ	6
1.1. Техническое задание	6
1.2. Анализ существующих аналогов	10
1.3. Перечень задач, подлежащих решению в процессе разработки.....	12
1.4. Обоснование выбора языков программирования.....	13
1.5. Обоснование выбора сред разработки.....	13
1.6. Описание реализуемой архитектуры	14
Выводы.....	15
2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА..	17
2.1. Разработка структуры системы	17
2.2. Разработка базы данных.....	17
2.3. Структура веб-приложения.....	27
Выводы.....	34
3. ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА	36
3.1. Тестирование системы	36
3.2. Руководство администратора	36
3.3. Руководство пользователя	39
Выводы.....	41
ЗАКЛЮЧЕНИЕ	42
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	43
Основная литература	43
Дополнительная литература	43

ВВЕДЕНИЕ

Тема курсовой работы актуальна в условиях стремительного роста популярности сервисов доставки готовой еды. В современном мире все большее внимание уделяется удобству, скорости и надежности услуг доставки. Разработка системы доставки готовой еды является важной задачей, так как она позволит обеспечить эффективную и безопасную организацию процесса доставки от ресторанов к потребителям.

Объектом курсовой работы является сфера сервисов доставки готовой еды.

Предметом исследования в рамках курсовой работы является разработка системы доставки готовой еды.

Целью работы является создание современной и надежной системы, способной обеспечить эффективную, быструю и безопасную доставку готовой еды от поставщиков к клиентам. Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ предметной области, определить текущие тенденции в сфере доставки еды и потребности клиентов.
2. Выполнить сравнительный анализ существующих систем доставки готовой еды.
3. Определить оптимальную структуру системы, учитывая требования к безопасности и надежности.
4. Выбрать средства и технологии для разработки системы доставки еды.
5. Реализовать функциональность системы.
6. Провести тестирование системы.
7. Разработать сопроводительную документацию, включая руководство пользователя и техническую документацию.

1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

1.1. Техническое задание

Наименование системы

Настоящее Техническое задание определяет требования и порядок создания «Системы доставки готовой еды».

Основания для разработки

Рост популярности сервисов доставки готовой еды и увеличение числа заказов через онлайн-платформы создают потребность в эффективной системе управления доставкой. В связи с этим возникает необходимость в создании универсальной платформы, способной обеспечить удобный и быстрый доступ к услугам доставки с любого устройства. Разработка такой системы предусматривает создание многофункционального веб-приложения, способного обрабатывать заказы, координировать логистику и обеспечивать надежную защиту данных пользователей. Такая система позволит эффективно управлять процессом доставки, а также обеспечит высокий уровень удобства и функциональности для клиентов и поставщиков.

Исполнитель

Исполнителем проекта является студент Калужского филиала МГТУ им. Н. Э. Баумана, факультета ИУК, группы ИУК4-62Б, Валявкин Максим Александрович.

Краткая характеристика области применения

Разрабатываемая система предназначена для применения в сфере доставки готовой еды. Ее целью является создание универсального веб-приложения, способного обеспечивать эффективное управление заказами и координацию доставки между ресторанами, клиентами и сотрудниками ресторана.

Целевая аудитория

Система доставки готовой еды ориентирована на пользователей, участвующих в заказе и получении еды. Она призвана обслуживать широкий круг клиентов, включая рестораны, их сотрудников и потребителей, с разнообразными уровнями технической оснащенности и предпочтениями.

Назначение системы

Разрабатываемая система доставки готовой еды предназначена для упрощения и эффективной организации процесса заказа и доставки еды. Она направлена на обеспечение простого доступа к услугам доставки, связанным с координацией заказов между клиентами, ресторанами и их сотрудниками.

Цели создания системы

Целью разработки системы доставки готовой еды является изучение процесса организации доставки и создание собственных механизмов для этого. Система будет направлена на повышение эффективности управления заказами и логистикой, обеспечивая удобство и прозрачность в процессе взаимодействия между клиентами, ресторанами и сотрудниками ресторана.

Плановые сроки начала и окончания работы по созданию системы

Планируемые сроки начала и окончания работы над работой: 12.03.2025 – 11.05.2025.

Требования к надежности

Надежность программного продукта для веб-приложения является критически важной для обеспечения устойчивого функционирования и защиты данных пользователей.

Надежное (устойчивое) функционирование системы должно быть обеспечено выполнением пользователем (заказчиком) совокупности организационно-технических мероприятий, перечень которых приведен ниже:

— организацией бесперебойного питания технических средств;

- использованием лицензионного программного обеспечения;
- регулярным выполнением рекомендаций Минтруда РФ, изложенных в Постановлении от 23 июля 1998 г. №28 «Об утверждении Межотраслевых типовых норм времени на работы по сервисному обслуживанию персональных электронно-вычислительных машин и организационной техники и сопровождению программных средств»;
- регулярным выполнением требований ГОСТ 51188-98 (защита информации, испытание компьютера на наличие компьютерных вирусов);

Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени восстановления операционной системы и восстановления работы сети.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Условия эксплуатации

Климатические условия эксплуатации системы должны соответствовать стандартам, предъявляемым к техническим средствам в сфере развлекательной индустрии и веб-технологий.

Требования к квалификации и численности персонала

Система требует наличия пользователей двух категорий – рядовых пользователей и администраторов, настраивающих бесперебойную работу веб-приложения, поддерживающих функционал и устраняющих неисправности в случае сбоя.

Требования к защите информации и программ

Защита информации осуществляется разграничением прав доступа – рядовые пользователи не должны иметь возможность вносить изменения, которые могут повлиять на функционирование системы.

Требования к программной документации

Должны быть разработаны следующие программные документы:

1. Расчетно-пояснительная записка:

- Техническое задание;
- Научно-исследовательская часть;
- Проектная часть;

Производственно-технологическая часть;

- Организационно-экономическая часть;

2. Графическая часть - 2 листа формата А1 включающие в себя:

- структурные схемы;
- основные алгоритмы;

Стадии разработки

Техническое задание

1. Обоснование перспективности реализуемого проекта:

- постановка задачи;
- сбор базовых материалов;
- установка критериев системы;
- необходимость проведения исследовательских работ;

2. Исследовательская работа:

- выбор оптимальных методов решения поставленной задачи;
- определение требований к техническим средствам;
- обоснование практической возможности реализации данного проекта;

3. Разработка и утверждение технического задания:

- определение требований к проекту;

- определение стадий, этапов и сроков разработки проекта и документации на нее;
- согласование и утверждение технического задания;

Технический проект

1. Разработка технического проекта:
 - определение формы представления входных и выходных данных;
 - определение конфигурации технических средств;
2. Утверждение технического проекта:
 - установка плана по разработке проекта;
 - создание пояснительной записки;
 - утверждение технического проекта;

Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии проектирования программы должен быть выполнен этап выборки программного обеспечения, сопутствующего ПО для разработки, библиотек для создания, этап проектирования системы в целом, разработка рабочей документации.

На стадии реализации производится разработка и тестирование спроектированной программы.

1.2. Анализ существующих аналогов

На текущий момент на рынке существует несколько аналогичных решений в области систем доставки готовой еды, однако, при проведении анализа выявлены некоторые ограничения и недостатки этих систем. В частности, основным аналогом является приложение «martipia». Некоторые аналоги предоставляют ограниченные возможности в управлении процессом доставки готовой еды, предлагая лишь базовый функционал или ограниченный набор инструментов. В отличие от них, разрабатываемая система обладает более широким спектром функций, позволяющих эффективно координировать заказы,

логистику и взаимодействие между клиентами, ресторанами и сотрудниками ресторана.

Существующие аналоги часто фокусируются на решении других задач, не связанных с основной функциональностью системы доставки еды. Например, они могут быть ориентированы на управление ресторанным бизнесом или общий логистический учет, что делает их менее эффективными для задач организации доставки. В отличие от них, наша система сосредотачивается исключительно на оптимизации процесса доставки готовой еды, обеспечивая удобство использования и максимальное удовлетворение потребностей пользователей в быстрой и надежной доставке.

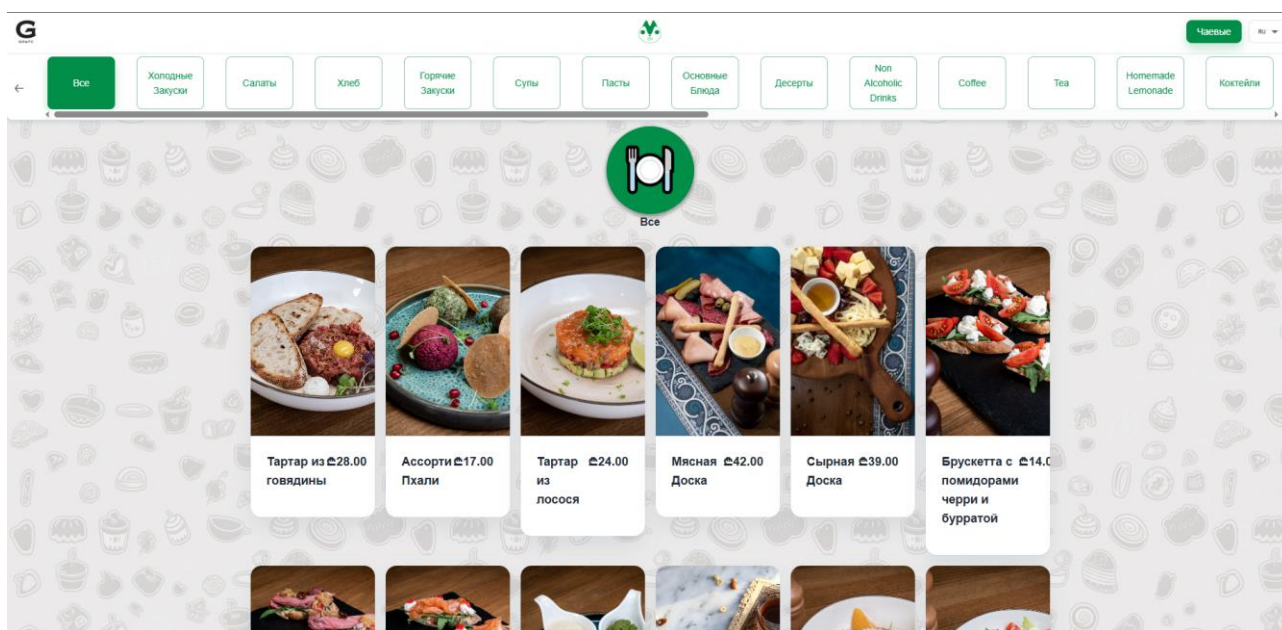


Рисунок 1.1 Приложение «martipia»

На основании проведенного анализа можно заключить, что существующие аналоги не всегда соответствуют требованиям данной системы доставки готовой еды. Наш проект нацелен на создание оптимального сочетания функционала, доступности и эффективности для пользователей, обеспечивая удобство и прозрачность процесса организации доставки между клиентами, ресторанами.

1.3. Перечень задач, подлежащих решению в процессе разработки

Для выполнения поставленной задачи необходимо определить оптимальную архитектуру для построения системы и средства реализации программного продукта, соответствующего данной архитектуре. Структура системы должна включать в себя клиентскую часть, написанную при использовании веб-фреймворка и серверную часть. Выбор средств реализации подразумевает выбор языков программирования и фреймворков для клиентской и серверной частей. Выбор инструментов разработки включает в себя выбор оптимальной операционной системы и среды разработки. Тестирование и отладка подразумевают проведение тестирования функционала и взаимодействия между пользователем и приложением, а также выявление и устранение возможных ошибок. Важным этапом является создание документации по коду и функционалу приложения и оптимизация системы для эффективного использования ресурсов.

Для реализации поставленной задачи выбраны следующие средства разработки. Для клиентской части использован фреймворк React, который обеспечивает удобную разработку интерфейса пользователя. В качестве серверной части выбран ExpressJS, веб-фреймворк на языке NodeJS, который обеспечивает гибкость и простоту в написании бэкенд-логики. Для работы с базой данных PostgreSQL были реализованы представления и хранимые функции для обеспечения целостности данных и выполнения бизнес-логики. Это позволяет эффективно управлять данными и обеспечить их корректность.

Важным этапом является создание подробной документации по коду и функционалу приложения, что обеспечивает понимание его работы и упрощает сопровождение. Оптимизация системы для эффективного использования ресурсов также была осуществлена, что позволяет обеспечить быструю и стабильную работу приложения на различных устройствах и операционных системах.

Важно определить оптимальную архитектуру системы и выбрать средства, соответствующие поставленным целям. Эффективное решение этих задач позволит создать качественное кроссплатформенное веб-приложение для осуществления доставки, который не только закрепит навыки веб-программирования, но и предоставит уникальный опыт использования для пользователей.

1.4. Обоснование выбора языков программирования

Для серверной части был выбран ExpressJS из-за его простоты и гибкости. ExpressJS позволяет быстро создавать RESTful API для взаимодействия с клиентской частью, а также обладает хорошей интеграцией с PostgreSQL, что важно для системы распределения платежей.

В качестве фронтенд-технологии был выбран React из-за его легкости в использовании и богатой экосистемы инструментов и библиотек. React позволяет создавать интерактивные пользовательские интерфейсы с минимальными усилиями, что важно для веб-приложения.

В качестве базы данных был выбран PostgreSQL из-за его надежности, производительности и богатого набора функций. PostgreSQL обеспечивает поддержку транзакций, целостность данных и возможность создания сложных запросов, что важно для системы распределения платежей.

Выбор данных технологий обоснован исходя из требований к производительности, гибкости, простоты использования и совместимости с другими компонентами системы. Эти технологии позволяют эффективно реализовать поставленные задачи и обеспечить высокое качество программного продукта.

1.5. Обоснование выбора сред разработки

Данный выбор оказывает значительное влияние на удобство разработки, имеющийся инструментарий, возможности отладки и развертывания

приложений на различных платформах, эффективность и качество реализуемых приложений.

VSCode – интегрированная среда разработки была выбрана для написания бэкенд-кода на ExpressJS и фронтенд-кода на React. Этот выбор обусловлен широким набором инструментов для разработки веб-приложений на языке NodeJS, включая возможность работы с фреймворком ExpressJS для серверной части и поддержку разработки на React для клиентской части.

pgAdmin – для работы с базой данных PostgreSQL было принято решение использовать pgAdmin. Этот инструмент обеспечивает удобное управление данными, включая написание и отладку SQL-запросов, а также предоставляет возможности для визуализации структуры базы данных и мониторинга ее состояния.

Figma – для проектирования пользовательского интерфейса веб-приложения было выбрано приложение Figma. Этот инструмент обеспечивает возможность создания макетов, прототипов и визуализаций интерфейсов, а также позволяет командам совместно работать над дизайном.

Chromium – для тестирования веб-приложения используется браузер Chromium. Этот выбор обусловлен тем, что Chromium является открытым проектом и предоставляет средства для отладки веб-приложения, а также обеспечивает схожесть с другими браузерами на базе Chromium.

Выбор данных инструментов обеспечивает полный цикл разработки веб-приложения от написания кода до проектирования интерфейса. Инструменты удовлетворяют требованиям проекта, обеспечивая гибкость и эффективность в работе с различными аспектами приложения.

1.6. Описание реализуемой архитектуры

Разрабатываемое веб-приложение основано на клиент-серверной архитектуре, включающей в себя фронтенд, серверную часть и базу данных с представлениями и хранимыми функциями.

В качестве фронтенд-фреймворка для создания пользовательских интерфейсов используется React. Этот фреймворк обеспечивает динамичные и интерактивные пользовательские интерфейсы, что делает его идеальным выбором для веб-приложения.

Для управления навигацией внутри веб-приложения используется компонент маршрутизации React. Это обеспечивает переключение между различными страницами приложения без перезагрузки страницы.

Вся необходимая информация для функционирования приложения хранится в базе данных PostgreSQL. Для взаимодействия с базой данных используются представления и хранимые функции, которые обеспечивают автоматическое обновление данных и выполнение сложных запросов.

Серверная часть приложения реализуется с использованием фреймворка ExpressJS. Этот фреймворк обеспечивает простоту и гибкость разработки, что делает его идеальным выбором для веб-приложения. Серверная часть приложения реализует логику обработки запросов от клиента, взаимодействие с базой данных и предоставление данных клиенту.

Эта архитектура подходит для веб-приложения, требующего серверной части и базы данных, и ориентированного на обеспечение высокой производительности и современного интерфейса.

Выводы

Таким образом, исходя из требований к разрабатываемому веб-приложению и учитывая особенности его стека технологий, рассмотрения возможностей наиболее подходящих инструментов, вариантов разработки и последующего их сравнения было решено использовать следующие решения:

В связи с клиент-серверной архитектурой приложения и использованием базы данных, для хранения данных было решено выбрать PostgreSQL, React для клиентской части и ExpressJS для серверной части. Этот выбор обусловлен

гибкостью и простотой в использовании данных технологий, а также их хорошей интеграцией между собой.

React был выбран для разработки клиентской части приложения. Этот инструмент позволяет создавать интерактивные пользовательские интерфейсы с помощью компонентной архитектуры и однофайловых компонентов, что облегчает разработку и поддержку кода.

Для разработки серверной части приложения был выбран ExpressJS. ExpressJS обладает легковесной структурой и простотой в изучении, что делает его отличным выбором для быстрой разработки веб-приложений. Кроме того, NodeJS обеспечивает удобную интеграцию с PostgreSQL для работы с базой данных.

База данных PostgreSQL была выбрана для хранения данных приложения. PostgreSQL обладает надежностью, производительностью и богатым набором функций, что делает его подходящим выбором для различных типов веб-приложений.

Для разработки и отладки кода на React и ExpressJS была выбрана интегрированная среда разработки, которая поддерживает обе технологии. Это обеспечивает удобство и эффективность при разработке и отладке как клиентской, так и серверной частей приложения.

Эти решения соответствуют требованиям по созданию веб-приложения с клиент-серверной архитектурой и обеспечивают эффективное развертывание и поддержку приложения.

2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

2.1. Разработка структуры системы

Разрабатываемая система доставки готовой еды представляет собой клиент-серверную архитектуру, основанную на компонентном подходе на фронтенде.

Компоненты пользовательского интерфейса – эти компоненты обеспечивают пользовательский интерфейс веб-приложения, предоставляя визуальное взаимодействие пользователей с системой.

Модуль управления корзиной – модуль обеспечивает взаимодействие с корзиной, включая логику, управление состоянием и передачу данных. Модуль состоит из компонентов, отвечающих за различные типы платежей и операции с ними.

Навигационные компоненты – обеспечивают навигацию между различными разделами и компонентами веб-приложения. Представлены двумя составляющими: Навигационное меню (предоставляет ссылки на основные разделы приложения) и Компоненты навигации между разделами меню.

Модуль заказов – предоставляет функционал для обработки и создания заказов, включая, изменение статуса и т.д.

Модуль товаров и категорий – предоставляет функционал для управления товарами и категориями, включая создание, изменение, добавление товаров в категории и т.д.

База данных с представлениями и хранимыми функциями – обеспечивает хранение и обработку данных о заказах, продуктах, категориях и т.д.

Серверная часть – обеспечивает взаимодействие между клиентской частью и базой данных, а также предоставляет API для взаимодействия с внешними системами.

2.2. Разработка базы данных

База данных состоит из следующих таблиц:

- user_admin (Администратор)
- user_token (Токен администратора)
- product (Товар)
- category (Категория)
- category_product (Продукт в категории)
- order (Заказ)
- product_in_order (Продукт в заказе)

Ниже приведена структура базы данных.



Рисунок 2.1 Структура базы данных

Ниже приведено описание каждой из таблиц.

Таблица 1 – «user_admin» – хранит данные авторизации Админа

Наименование поля	Тип данных	Признак ключа	Примечание
Id	Integer	ПК	Уникальное значение

login	String(200)		Не может быть NULL
password	String(30)		Не может быть NULL

Таблица 2 – «user_token» – хранит токены доступа Администраторов

Наименование поля	Тип данных	Признак ключа	Примечание
Id	Integer	PK	Уникальное значение
user_id	Integer	FK	Не может быть NULL, ссылается на поле id таблицы user_admin, при удалении записи из таблицы user_admin соответствующая запись из таблицы user_token также будет удалена
token	String(256)		Не может быть NULL

Таблица 3 – «product» – хранит общую информацию о товарах

Наименование поля	Тип данных	Признак ключа	Примечание
Id	Integer	PK	Уникальное значение
name	String(100)		Не может быть NULL
is_sale	Bool		Не может быть NULL
sale_price	Numeric(10, 2)		Не может быть NULL

price	Numeric(10, 2)		Не может быть NULL
weight	Integer	FK	Не может быть NULL
description	String(300)		Не может быть NULL
img	String(120)		Не может быть NULL

Таблица 4 – «category» – хранит общую информацию о категориях

Наименование поля	Тип данных	Признак ключа	Примечание
Id	Integer	PK	Уникальное значение
name	String(100)		Не может быть NULL

Таблица 5 – «category_product» – хранит связь продукт-категория

Наименование поля	Тип данных	Признак ключа	Примечание
Id	Integer	PK	Уникальное значение
category_id	Integer	FK	Не может быть NULL, Ссылается на поле id таблицы category, при удалении записи из таблицы category соответствующая запись из таблицы category_product также будет удалена
product_id	Integer	FK	Не может быть NULL, Ссылается на поле id таблицы

			product, при удалении записи из таблицы product соответствующая запись из таблицы category_product также будет удалена
--	--	--	--

Таблица 6 – «orders» – хранит общую информацию о заказах

Наименование поля	Тип данных	Признак ключа	Примечание
Id	Integer	РК	Уникальное значение
name	String(30)		Не может быть NULL
phone	String(30)		Не может быть NULL
delivery	String(30)		Не может быть NULL
address	String(410)		Не может быть NULL
flat	String(110)		Не может быть NULL
comment	String(410)		Не может быть NULL
tools	String(30)		Не может быть NULL
price	Numeric(10, 2)		Не может быть NULL
status	String(40)		Не может быть NULL
order_date	Date		Не может быть NULL
order_time	Time		Не может быть NULL

Таблица 7 – «product_in_order» – хранит общую информацию о одном продукте и его количестве в заказе

Наименование поля	Тип данных	Признак ключа	Примечание
Id	Integer	PK	Уникальное значение
order_id	Integer	FK	Не может быть NULL, Ссылается на поле id таблицы order, при удалении записи из таблицы order соответствующая запись из таблицы product_in_order также будет удалена
product_id	Integer	FK	Не может быть NULL, Ссылается на поле id таблицы product, при удалении записи из таблицы product соответствующая запись из таблицы product_in_order также будет удалена
count	Integer		Не может быть NULL
current_price	Numeric(10, 2)		Не может быть NULL

Ниже приведено описание хранимых функций и представлений.

save_product

Функция `save_product` предназначена сохранения продукта нового продукта в базу данных.

Функция принимает параметры `name`, `is_sale`, `sale_price`, `price`, `weight`, `description`, `img_path`, `img_exp` – соответствующие полям в таблице `product`.

Внутри функции объявляется переменная: `_id` – индификатор, который получит продукт при сохранении.

С помощью запроса `INSERT` продукт сохраняется.

Затем, с помощью запроса `UPDATE`, сохраняется, сформированный в функции, при помощи `_id`, путь к изображению.

Особенности функции:

- Функция использует язык PL/pgSQL.
- Функция использует подзапросы `INSERT INTO` для добавления новой записи в таблицу `product` и получения значения переменной `_id`, и подзапросы `UPDATE` для добавления пути изображения в таблицу `product`.
- Функция возвращает `id` продукта.

category_with_all_products

Представление создается для предоставления информации о категориях товаров, включая связанные с ними продукты и продукты, которые не связаны с данной категорией. Представление использует SQL-запросы с общими табличными выражениями, агрегацию `JSON` и условные объединения для формирования результата.

Общие табличные выражения:

`distinct_products`:

Формирует список уникальных пар `category_id` и `product_id` из таблицы `category_product`, представляющих связь между категориями и продуктами.

Используется SELECT DISTINCT для исключения дубликатов.

distinct_other_products:

Формирует список пар category_id и product_id для продуктов, которые не связаны с данной категорией.

Используется CROSS JOIN между таблицами category и product для создания всех возможных комбинаций.

Условие NOT EXISTS исключает пары, которые уже есть в таблице category_product.

Основной запрос:

Выбирает данные из таблицы category (C.id, C.name).

Для каждой категории формирует два JSON-массива:

products: Список продуктов, связанных с категорией.

Используется подзапрос с JSON_AGG и JSON_BUILD_OBJECT для создания JSON-объектов с полями id, name, price, is_sale, sale_price, description для продуктов, связанных с категорией через таблицу category_product.

Продукты сортируются по id в порядке убывания (ORDER BY P1.id DESC).

Если продуктов нет, возвращается пустой массив '[]' с помощью COALESCE.

other: Список продуктов, не связанных с категорией.

Аналогично формируется JSON-массив для продуктов из CTE distinct_other_products.

Продукты также сортируются по id в порядке убывания.

Если таких продуктов нет, возвращается пустой массив '[]'.

Группировка:

Данные группируются по C.id и C.name для обеспечения уникальности категорий в результате.

Особенности представления:

Использование CTE: Позволяет разбить сложную логику на более читаемые части, избегая дублирования кода.

JSON-агрегация: Формирует структурированные данные в формате JSON для удобной работы с результатами (например, в приложениях).

Обработка пустых данных: Использование COALESCE гарантирует, что вместо NULL возвращаются пустые массивы '[]'.

Фильтрация данных: Условие NOT EXISTS в distinct_other_products эффективно исключает уже связанные продукты.

Сортировка: Продукты в массивах products и other упорядочены по убыванию id, что обеспечивает предсказуемый порядок вывода.

Динамическая структура: Представление автоматически адаптируется к изменениям в таблицах category, product и category_product.

Итоговый результат:

Каждая строка представления содержит:

id: Идентификатор категории.

name: Название категории.

products: JSON-массив продуктов, связанных с категорией.

other: JSON-массив продуктов, не связанных с категорией.

category_with_included_products

Представление создается для предоставления информации о категориях товаров и связанных с ними продуктах. Оно формирует структурированный вывод, включая JSON-массив продуктов для каждой категории, с использованием общих табличных выражений и агрегации JSON.

Общие табличные выражения:

distinct_products:

Извлекает уникальные пары category_id и product_id из таблицы category_product, представляющих связь между категориями и продуктами.

Применяется SELECT DISTINCT для исключения дубликатов.

distinct_other_products:

Формирует список пар category_id и product_id для продуктов, которые не связаны с данной категорией.

Используется CROSS JOIN между таблицами category и product для создания всех возможных комбинаций.

Условие NOT EXISTS исключает пары, уже присутствующие в таблице category_product.

Основной запрос:

Выбирает данные из таблицы category (C.id, C.name).

Для каждой категории формирует JSON-массив products:

Используется подзапрос с JSON_AGG и JSON_BUILD_OBJECT для создания JSON-объектов, содержащих поля:

id: Идентификатор продукта.

name: Название продукта.

price: Цена продукта.

is_sale: Флаг, указывающий, находится ли продукт на распродаже.

sale_price: Цена со скидкой.

description: Описание продукта.

weight: Вес продукта.

image: Путь или имя изображения продукта (поле img).

Продукты сортируются по id в порядке убывания (ORDER BY P1.id DESC).

Если продуктов нет, возвращается пустой массив '[]' с помощью COALESCE.

Группировка:

Данные группируются по C.id и C.name для обеспечения уникальности категорий в результате.

Особенности представления:

Использование CTE: Разбивает логику на модульные части, упрощая чтение и поддержку кода.

JSON-агрегация: Формирует данные в формате JSON, удобном для использования в приложениях.

Обработка пустых данных: COALESCE гарантирует возврат пустого массива '[]' вместо NULL.

Расширенные атрибуты продукта: Включает дополнительные поля weight и image, в отличие от предыдущего представления category_with_all_products.

Сортировка: Продукты упорядочены по убыванию id, обеспечивая предсказуемый порядок.

Динамическая структура: Представление автоматически обновляется при изменениях в таблицах category, product или category_product.

Итоговый результат:

Каждая строка представления содержит:

id: Идентификатор категории.

name: Название категории.

products: JSON-массив продуктов, связанных с категорией, с полями id, name, price, is_sale, sale_price, description, weight, image.

2.3. Структура веб-приложения

Веб-приложение представляет собой многофункциональную платформу, сосредоточенную на доставке. Ключевые компоненты и функции приложения включают:

1. Создание и управление товарами:
 - Возможность добавления товаров
 - Возможность изменение товаров
 - Возможность удаления товаров
2. Создание и управление категориями:
 - Возможность добавления категорий
 - Возможность изменение категорий
 - Возможность удаления категорий
 - Возможность добавления товаров в категорию
 - Возможность удаления товаров из категории

3. Управление заказами:

- Формирование новых заказов.
- Изменение и отслеживание статуса заказа.

4. Авторизация администраторов:

- Вход администратора в систему.
- Проверка на подлинность сессии администратора.

Структура бэкенда включает файлы:

- AuthRouter.js
- CategoryRouter.js
- OrderRouter.js
- ProductRouter.js
- AuthController.js
- CategoryController.js
- OrderController.js
- ProductController.js
- index.js
- database.js

Рассмотрим подробнее:

- AuthRouter.js - файл, содержащий маршруты для аутентификации и авторизации администраторов.
- CategoryRouter.js - файл, содержащий маршруты для управления категориями.
- OrderRouter.js - файл, содержащий маршруты для управления заказами.
- ProductRouter.js - файл, содержащий маршруты для управления товарами.
- index.js - файл, содержащий основной код приложения, включая создание и настройку экземпляра приложения ExpressJS.

- database.js - файл, содержащий настройки подключения к базе данных.
- AuthController.js – файл обработки запросов на управление авторизацией.
- CategoryController.js - файл обработки запросов на управление категориями.
- OrderController.js - файл обработки запросов на управление заказами.
- ProductController.js - файл обработки запросов на управление товарами.

Рассмотрим основные функции:

- login(req, res, next) - функция авторизации пользователя.
- checkAuth(req, res, next) - функция для получения проверки токена авторизации пользователя.
- logout(req, res, next) - функция обработки выхода пользователя.
- getCategories(req, res, next) - функция для получения информации о всех категориях.
- addCategory(req, res, next) - функция для добавления категории.
- deleteCategory(req, res, next) - функция для удаления категории с указанным идентификатором id.
- setCategory(req, res, next) - функция для изменения категории с указанным идентификатором id.
- getCategory(req, res, next) - функция для получения категории с указанным идентификатором id.
- addProductToCategory(req, res, next) - функция для добавления товара в категорию с указанными идентификаторами id.
- deleteProductFromCategory(req, res, next) - функция для удаления товара из категории с указанными идентификаторами id.
- getMenu(req, res, next) - функция для получения меню ресторана.
- addOrder(req, res, next) - функция для формирования заказа.

- `getOrder(req, res, next)` - функция для получения заказа с указанным идентификатором `id`.
- `setStatus(req, res, next)` - функция для изменения статуса заказа с указанным идентификатором `id`.
- `parseFormData(req)` - функция для обработки объекта `FormData` и получения `json`-объекта и изображений.
- `addProduct(req, res, next)` - функция для добавления товара.
- `getProduct(req, res, next)` - функция для получения товара с указанным идентификатором `id`.
- `deleteProduct(req, res, next)` - функция для удаления товара с указанным идентификатором `id`.
- `setProduct(req, res, next)` - функция для изменения товара с указанным идентификатором `id`.
- `getProducts(req, res, next)` - функция для получения информации о всех товарах.
- `getPrices(req, res, next)` - функция для получения информации о цене на товары.

Структура фронтенда включает:

- `node_modules`
- `assets`
- `scss`
- `AdminHeader.jsx`
- `CheckAuth.jsx`
- `Footer.jsx`
- `Header.jsx`
- `ItemCard.jsx`
- `Loading.jsx`
- `Navigation.jsx`
- `Popup.jsx`

- AddCategory.jsx
- AddProduct.jsx
- Cart.jsx
- Categories.jsx
- Login.jsx
- MenuMain.jsx
- Orders.jsx
- ProductList.jsx
- ProductPage.jsx
- SetCategory.jsx
- SetProduct.jsx
- CartCounterProvider.jsx
- FetchService.jsx
- ValidationService.jsx
- CartService.jsx
- App.jsx
- index.js
- package.json
- package-lock.json

Рассмотрим подробнее:

- node_modules - директория, содержащая зависимости приложения, установленные с помощью npm.
- assets - директория, содержащая файлы ресурсов.
- scss - директория, содержащая файлы стилей.
- AdminHeader.jsx - компонент, отображающий «шапку» сайта для администратора.
- CheckAuth.jsx - компонент, проводящий проверку авторизации администратора.
- Footer.jsx - компонент, отображающий «подвал» сайта.

- Header.jsx - компонент, отображающий «шапку» сайта.
- ItemCard.jsx - компонент, отображающий карточку товара.
- Loading.jsx - компонент, отображающий колесо загрузки.
- Navigation.jsx - компонент, отображающий навигационное меню по разделу «Меню».
- Popup.jsx - компонент, отображающий всплывающее окно «popup».
- AddCategory.jsx - компонент, отображающий страницу с функционалом добавления категории.
- AddProduct.jsx - компонент, отображающий страницу с функционалом добавления товара.
- Cart.jsx - компонент, отображающий страницу с функционалом обработки корзины.
- Categories.jsx - компонент, отображающий страницу с функционалом обработки категорий.
- Login.jsx - компонент, отображающий страницу с функционалом обработки авторизации.
- ProductList.jsx - компонент, отображающий страницу с функционалом обработки товаров.
- Orders.jsx - компонент, отображающий страницу с функционалом обработки заказов.
- MenuMain.jsx - компонент, отображающий страницу с меню.
- ProductPage.jsx - компонент, отображающий страницу продукта.
- SetCategory.jsx - компонент, отображающий страницу с функционалом изменения категории.
- SetProduct.jsx - компонент, отображающий страницу с функционалом изменения продуктов.
- CartCounterProvider.jsx - провайдер, хранящий и обрабатывающий состояние счетчика товаров в корзине.

- `FetchService.jsx` - сервис, обрабатывающий обращения к серверу, по средствам `http`-запросов.
- `ValidationService.jsx` - сервис, осуществляющий функционал валидации форм.
- `CartService.jsx` - сервис, обрабатывающий состояние корзины пользователя.
- `App.js` - главный компонент `React`, содержащий общую структуру приложения.
- `index.js` - главный `JavaScript`-файл приложения, инициализирующий приложение `React`.
- `package.json` - файл, содержащий информацию о приложении и его зависимостях.
- `package-lock.json` - файл, содержащий информацию о версиях установленных зависимостей.

Маршруты приложения включают:

- `'/'` (компонент: `MenuMain`)
- `'/menu'` (компонент: `MenuMain`)
- `'/product/:id'` (компонент: `ProductPage`)
- `'/cart'` (компонент: `Cart`)
- `'/adpn'` (компонент: `ProductList`, требуется аутентификация и роль администратора)
- `'/adpn-products'` (компонент: `ProductList`, требуется аутентификация и роль администратора)
- `'/adpn-add'` (компонент: `AddProduct`, требуется аутентификация и роль администратора)
- `'/adpn-product-set/:id'` (компонент: `SetProduct`, требуется аутентификация и роль администратора)
- `'/adpn-login'` (компонент: `Login`, требуется аутентификация и роль администратора)

- '/adpn-categories' (компонент: Categories, требуется аутентификация и роль администратора)
- '/adpn-categories-add' (компонент: AddCategory, требуется аутентификация и роль администратора)
- '/adpn-category-set/:id' (компонент: SetCategory, требуется аутентификация и роль администратора)
- '/adpn-orders' (компонент: Orders, требуется аутентификация и роль администратора)

Выводы

В результате был реализован программный продукт, включающий в себя клиентскую часть, серверную часть и базу данных. Компоненты пользовательского интерфейса, модуль обычного пользователя, администратора и навигационные компоненты были разработаны в соответствии с функциональными требованиями технического задания.

Фронтенд был реализован с использованием фреймворка React и включает в себя компоненты для отображения и взаимодействия с данными, а также навигационные компоненты для перемещения между различными разделами приложения.

Бэкенд был реализован с использованием фреймворка ExpressJS и включает в себя модули для обработки HTTP-запросов, аутентификации и авторизации пользователей, а также взаимодействия с базой данных.

База данных была реализована с использованием СУБД PostgreSQL и включает в себя таблицы для хранения данных о пользователях, товарах, категориях и заказах. Для взаимодействия с базой данных используются pgAdmin и SQL-запросы.

Также были реализованы функции для управления товарами, категориями, заказами, аутентификации и авторизации пользователей, а также взаимодействия с базой данных.

В целом, программный продукт реализует систему управления доставкой готовой еды, предоставляя пользователям возможность заказывать еду, а администраторам отслеживать статус доставки, а также управлять заказами и взаимодействием между клиентами, ресторанами и их сотрудниками.

3. ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

3.1. Тестирование системы

Тестирование системы производилось с использованием персонального компьютера и мобильных устройств на базе ОС Android. Тестирование проводилось в различных веб-браузерах и операционных системах на персональных компьютерах с использованием инструментов для разработчиков с акцентом на кроссплатформенность приложения. На мобильных устройствах производилось тестирование адаптивности веб-приложения и его производительности.

Требования к веб-приложению

Для работы веб-приложения необходимо, чтобы компьютер обладал следующими минимальными характеристиками:

- процессор с тактовой частотой не менее 1 ГГц;
- ОЗУ 1 Гб;
- 2 Гбайт свободного пространства на диске;
- Microsoft Windows, MacOS, Linux

Для взаимодействия с приложением необходимо наличие мыши, монитора и клавиатуры.

Требования к программному обеспечению

Для работы приложения необходимо наличие веб-браузера и доступа к сети Интернет.

3.2. Руководство администратора

Запуск и поддержка веб-приложения

Убедитесь, что на сервере установлены необходимые компоненты, такие как Node.js, npm (или yarn), база данных (PostgreSQL) и другие зависимости, указанные в документации.

Склонируйте репозиторий проекта с GitHub.

Откройте терминал или командную строку и перейдите в каталог проекта.

Выполните команду `npm install` или `yarn install`, чтобы установить все зависимости, указанные в файле `package.json`.

Выполните команду `npm run backend` для запуска серверной части приложения.

Убедитесь, что сервер успешно запущен и не возникло ошибок подключения к базе данных.

Откройте новое окно терминала или командной строки.

Перейдите в каталог клиентской части проекта.

Выполните команду `npm start` или `yarn start` для запуска клиентской части приложения.

После успешного запуска, откройте браузер и перейдите по адресу, указанному в выводе команды (обычно это `http://localhost:3000`).

Перейдите на страницу входа в систему и используйте учетные данные администратора для входа.

После успешного входа в систему вы получите доступ к административной панели, где сможете управлять пользователями, мероприятиями и другими функциями приложения.

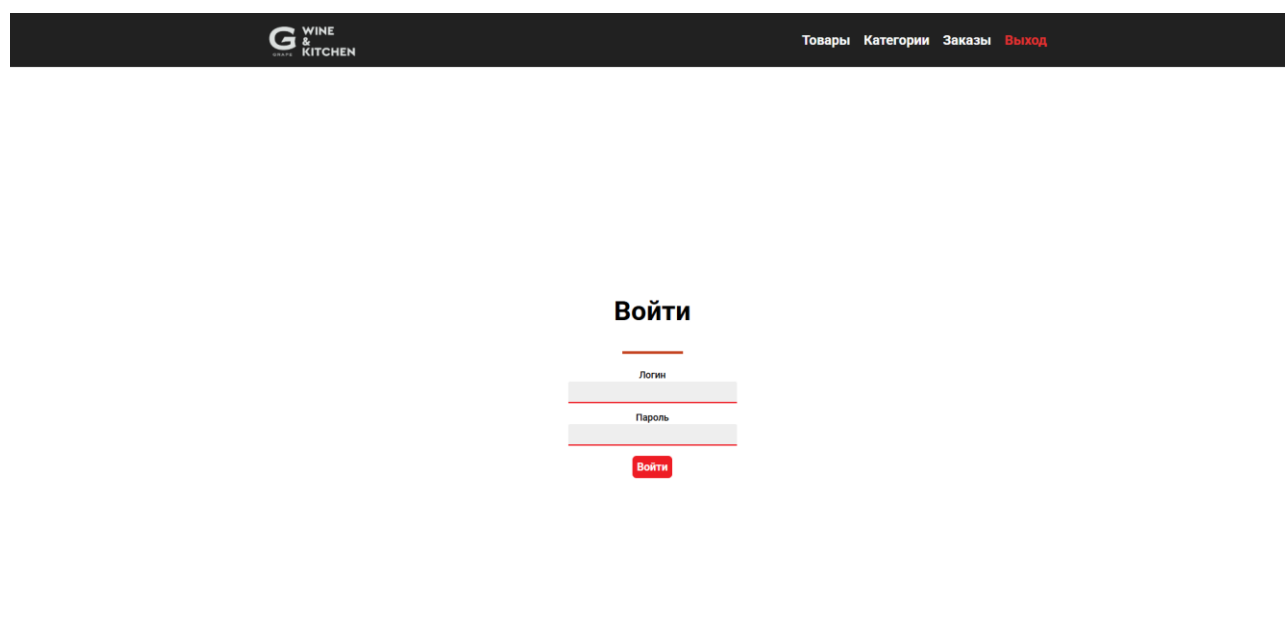


Рисунок 3.1. Страница авторизации

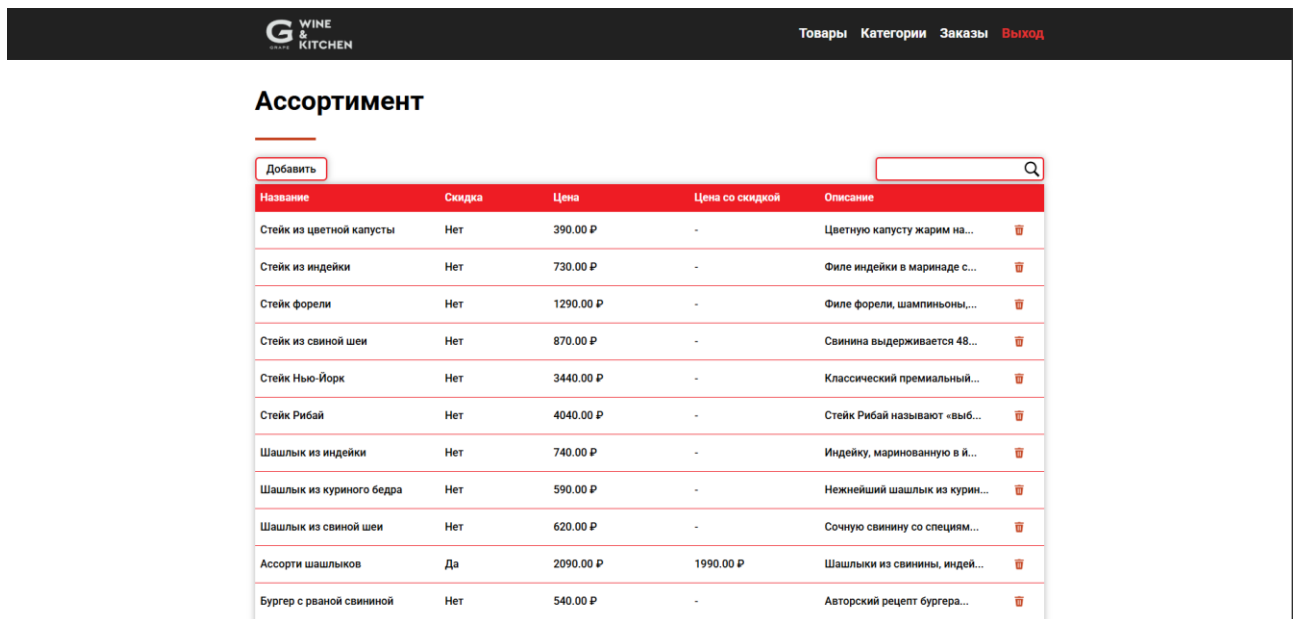


Рисунок 3.2. Главная страница панели администратора, информация о товарах

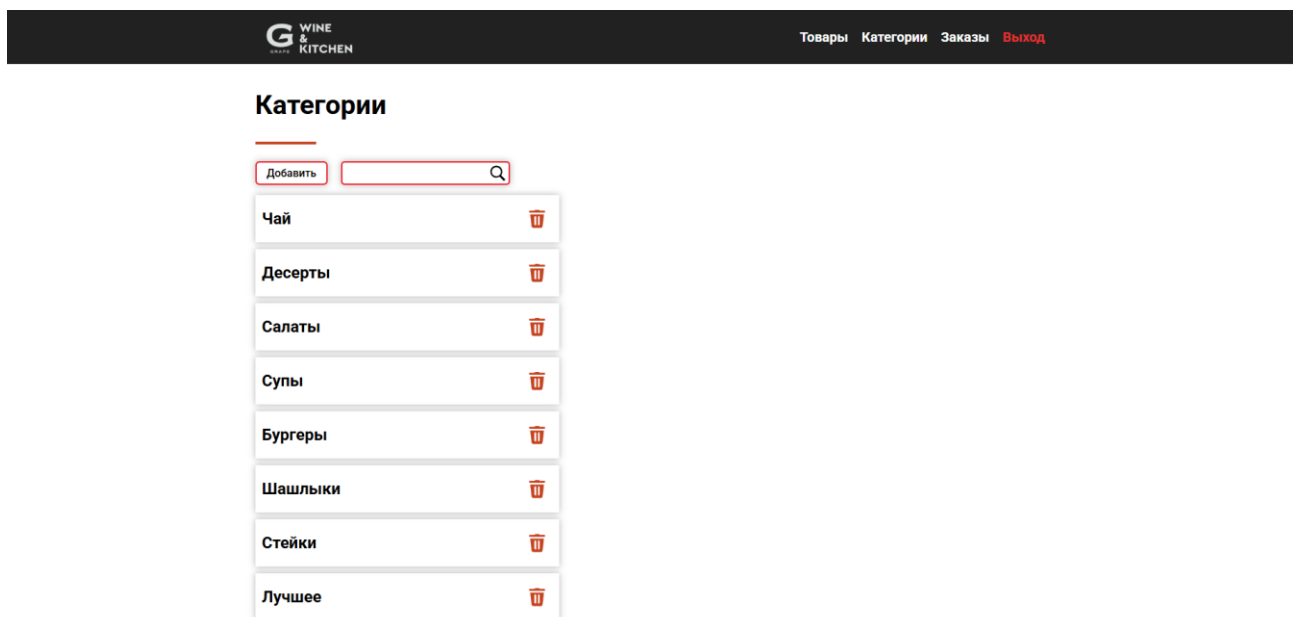


Рисунок 3.3. Вкладка с информацией о категориях

<div> <div>G WINE & KITCHEN</div> <div> Товары Категории Заказы Выход </div> </div>	
Заказы	
Настя(23323)	Получен
Володя(23324)	Получен
Аня(23325)	Отдан в доставку
Марго(23326)	Отдан в доставку
Александр(23327)	Отдан в доставку
Полина(23328)	Доставлен
Коля(23329)	Доставлен
Коля(23330)	Доставлен
Володя(23331)	Доставлен

Рисунок 3.4. Вкладка с историей заказов

3.3. Руководство пользователя

Руководство к веб-приложению

После успешного запуска, откройте браузер и перейдите по адресу <http://localhost:3000>.

В навигационном меню выберите нужную категорию «меню» или перейдите к ней колесиком мыши. Выберите нужное блюдо и нажмите на него.

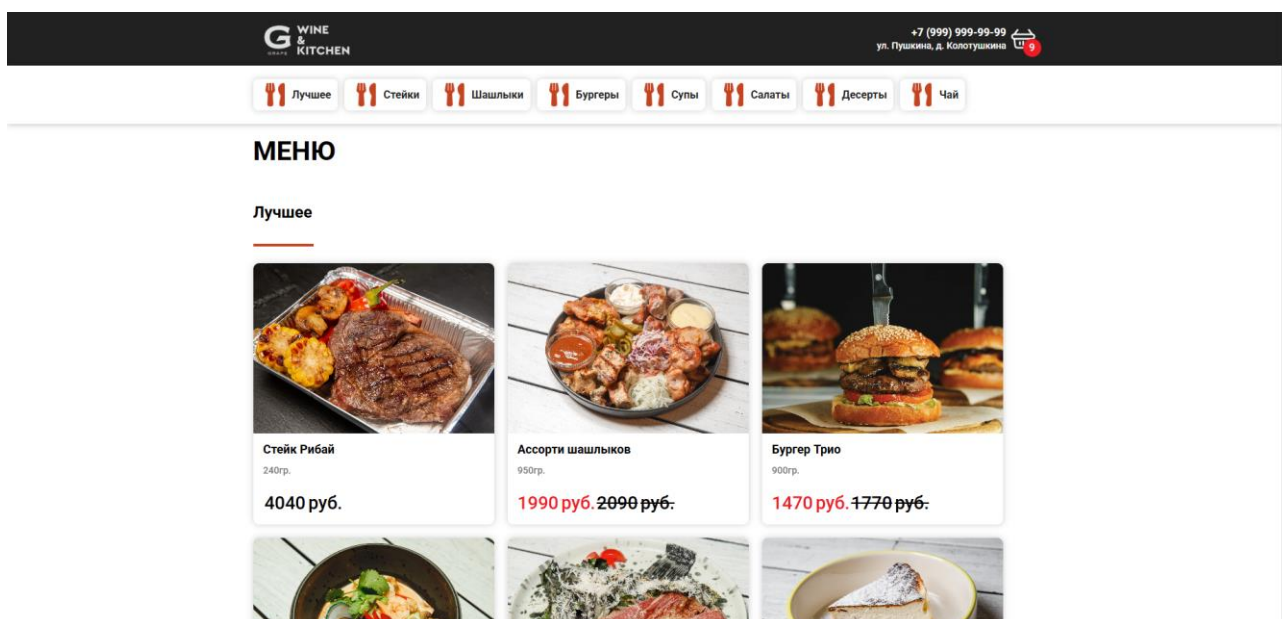


Рисунок 3.5. Страница меню

Прочитайте описание блюда и, если оно вам подходит, добавьте его в корзину, нажав на кнопку «в корзину». Тем же образом добавьте в корзину остальные понравившиеся блюда.



Рисунок 3.6. Страница товара

Затем, кликом по значку корзины на «шапке» сайта перейдите в корзину. Кнопками «плюс» и «минус» укажите количество каждого блюда и убедитесь, что цена вам подходит. После введите данные для доставки и нажмите кнопку «Оформить заказ».

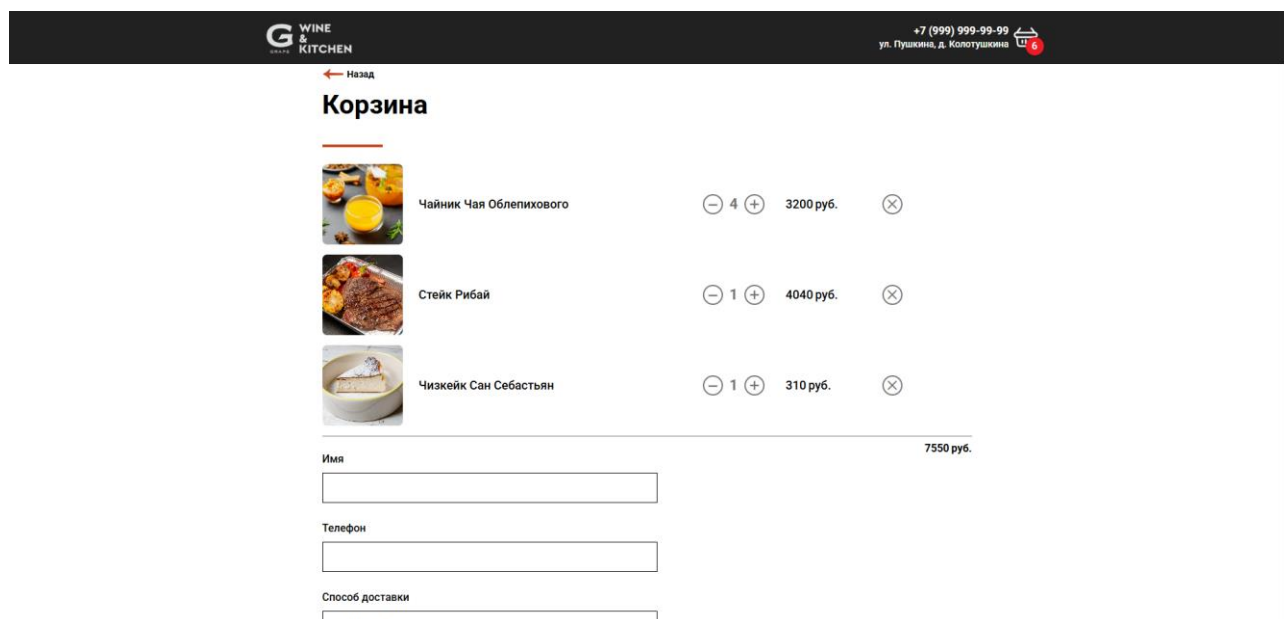
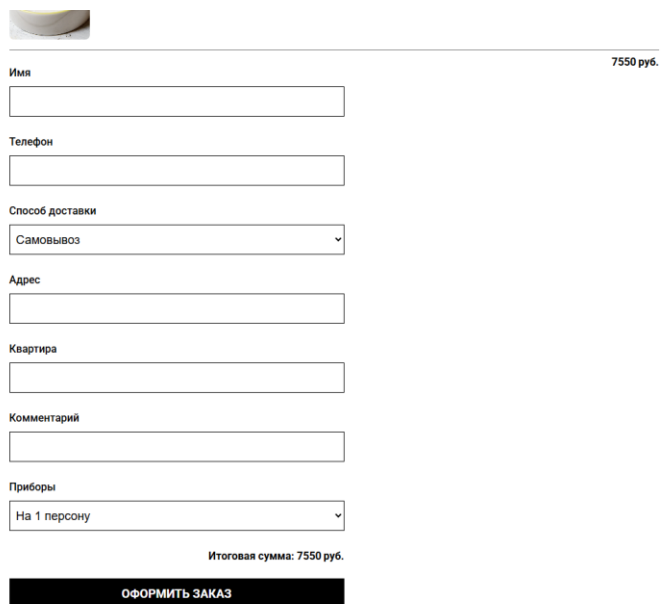


Рисунок 3.7. Корзина пользователя



The screenshot shows a web form for placing an order. At the top left is a small circular logo. The form fields are arranged vertically: 'Имя' (Name) with a text input; 'Телефон' (Phone) with a text input; 'Способ доставки' (Delivery method) with a dropdown menu showing 'Самовывоз' (Self-pickup); 'Адрес' (Address) with a text input; 'Квартира' (Apartment) with a text input; 'Комментарий' (Comment) with a text input; and 'Приборы' (Instruments) with a dropdown menu showing 'На 1 персону' (For 1 person). In the top right corner, the text '7550 руб.' is displayed. Below the form fields, the text 'Итоговая сумма: 7550 руб.' (Total amount: 7550 rub.) is shown. At the bottom of the form is a black button with the white text 'ОФОРМИТЬ ЗАКАЗ' (Place order).

Рисунок 3.8. Форма отправки заказа

Выводы

В результате были выполнены тестирование компонентов программного продукта. Разработана техническая документация.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была создана система управления доставкой готовой еды. Результат соответствует всем заранее определенным требованиям технического задания. Разработанная система обладает гибкостью, многофункциональностью и удобством в эксплуатации. Внедрение данной системы предоставит уникальные возможности для улучшения пользовательского опыта и повышения эффективности взаимодействия между клиентами, ресторанами и сотрудниками ресторана. Благодаря минимизации времени на поиск необходимой информации, пользователи смогут более полно контролировать статус заказов, не отвлекаясь на лишние шаги. Экономия времени и отсутствие необходимости в сложных процессах координации создадут удобные условия для взаимодействия с системой. Такие улучшения приведут к повышению общей производительности и удовлетворенности пользователей, что, в свою очередь, способствует росту популярности и успеху данной системы доставки готовой еды.

Поскольку поставленная задача была выполнена в полном объеме, дальнейшее ее расширение не планируется. Однако разработанная архитектура позволяет при необходимости улучшить и расширить данную систему.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Основная литература

1. Романенко, В.В. Объектно-ориентированное программирование : учебное пособие / В.В. Романенко ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Томский государственный университет систем управления и радиоэлектроники, 2014. - 475 с. : ил. - Библиогр.: с. 442. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=480517>.
2. Николаев, Е.И. Объектно-ориентированное программирование : учебное пособие / Е.И. Николаев ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северо-Кавказский федеральный университет». - Ставрополь : СКФУ, 2015. - 225 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=458133>.
3. Суханов, М.В. Основы Microsoft .NET Framework и языка программирования C# : учебное пособие / М.В. Суханов, И.В. Бачурин, И.С. Майоров ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессионального образования Северный (Арктический) федеральный университет им. М.В. Ломоносова. - Архангельск : ИД САФУ, 2014. - 97 с. : схем., табл., ил. - Библиогр. в кн. - ISBN 978-5-261-00934-4 ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=312313>.

Дополнительная литература

1. Мейер, Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс] / Б. Мейер. — 3-е изд. — Электрон.

- текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 285 с. — 978-5-4486-0513-0. — Режим доступа: <http://www.iprbookshop.ru/79706.html>
2. Моделирование систем[Текст]: учебник для вузов / С.И. Дворецкий, Ю.Л. Муромцев, В.А. Погонин, А.Г. Схиртладзе. – М.: Академия, 2009. – 320 с.
 3. Щербаков, А. Интернет-аналитика [Электронный ресурс]: поиск и оценка информации в web-ресурсах: практическое пособие / А. Щербаков. - М.: Книжный мир, 2012. - 78 с. - URL: <http://biblioclub.ru/index.php?page=book&id=89693>.
 4. Маюрникова, Л. А. Основы научных исследований в научно-технической сфере [Электронный ресурс]: учебно-методическое пособие / Л. А. Маюрникова, С. В. Новосёлов. — Кемерово: Кемеровский технологический институт пищевой промышленности, 2009. — 123 с. — Режим доступа: <http://www.iprbookshop.ru/14381.html>
 5. Вайнштейн, М. З. Основы научных исследований [Электронный ресурс] : учебное пособие / М. З. Вайнштейн, В. М. Вайнштейн, О. В. Кононова. — Йошкар-Ола: Марийский государственный технический университет, Поволжский государственный технологический университет, ЭБС АСВ, 2011. — 216 с. — Режим доступа: <http://www.iprbookshop.ru/22586.html>
 6. Дэвид Флэнаган, Еннет Фралин. JavaScript. Подробное руководство. - Перевод с английского. - СПб.: Питер, 2018. - 872 с.
 7. Смит Дж., Джонсон А. React: Полное руководство. - Перевод с английского. - М.: БХВ-Петербург, 2021. - 480 с.
 8. Долин А. Node.js. Разработка веб-приложений. - Перевод с английского. - М.: БХВ-Петербург, 2020. - 400 с.
 9. Коул Р. Node.js для профессионалов. - Перевод с английского. - М.: Вильямс, 2019. - 320 с.
 10. Мерфи М. Node.js в действии. - Перевод с английского. - М.: Питер, 2021. - 480 с.

11. Фаина Ю. PostgreSQL. Базы данных на максималках. - СПб.: Питер, 2020.
- 432 с.
12. Карнаух И. И. PostgreSQL. Администрирование и оптимизация. - СПб.: БХВ-Петербург, 2016. - 496 с.
13. Альбитро М. RESTful Web APIs. Создание масштабируемых веб-сервисов.
- Перевод с английского. - К.: Диалог, 2016. - 304 с.