Министерство образования Иркутской области
Государственное бюджетное профессиональное
образовательное учреждение Иркутской области
«Иркутский авиационный техникум»
(ГБПОУИО «ИАТ»)

КР.09.02.07-5.24.222.13 ПЗ

ВЕБ-ПРИЛОЖЕНИЕ «КОНДИТЕРСКАЯ»

Председатель ВЦК: _____ (М.А. Кудрявцева)
(подпись, дата)

Руководитель: _____ (М.А. Кудрявцева)
(подпись, дата)

Студент: _____ (Е.М. Кривогорницына)
(подпись, дата)

Иркутск 2024

Содержание

Введение

В современном мире индустрия кондитерских изделий постоянно развивается, следуя за непрерывно меняющимися предпочтениями покупателей. Торты и десерты стали неотъемлемой частью жизни для каждого. На любом празднике уже невозможно представить себе стол без этих сладких изделий.

Кондитерская – это специализированный магазин, где представлен широкий ассортимент кондитерских изделий. В отличие от обычных магазинов, здесь можно найти более ограниченный выбор.

С каждым годом спрос на кондитерские изделия постоянно растет. Это связано не только с изменениями в питании, но и с повышением уровня жизни населения. Покупатели предъявляют высокие требования к качеству продукции и разнообразию ассортимента.

В связи с этим кондитерская индустрия не стоит на месте и постоянно развивается, предлагая новые вкусы и технологии. От классических рецептов до уникальных сочетаний вкусов – мир сладостей становится всё более интересным и доступным для всех.

Актуальность объясняется несколькими важными факторами.

В наше время, когда технологии развиваются быстрыми темпами, а интерес покупателей к онлайн-сервисам постоянно растёт, всё больше людей предпочитают заказывать продукты и услуги через интернет.

Качественный дизайн веб-приложения, вдохновлённый кондитерской тематикой, может сделать процесс заказа не только простым, но и приятным. Уникальные визуальные элементы и интуитивно понятная навигация помогут выделить проект среди конкурентов и оставить положительное впечатление у пользователей. Таким образом, создание веб-приложения для кондитерской – это шаг к успешной интеграции современных технологий, что в конечном итоге способствует не только росту продаж, но и формированию локальной клиентской базы.

Создание веб-приложения упростит процесс заказа для клиентов, и значительно оптимизирует управление ассортиментом кондитерской. Внедрение веб-

приложения позволит кондитерской автоматизировать такие процессы как: приём заказов, взаимодействия клиента с продукцией, обеспечить клиентам круглосуточный доступ к услугам кондитерской, независимо от их местоположения.

В современном мире, где время становится всё более ценным ресурсом, люди стремятся максимально упростить свои повседневные задачи. Веб-приложение для кондитерской становится не только удобным инструментом для покупателей, но и важным элементом в развитии кондитерской.

Кроме того, в условиях растущей конкуренции на рынке кондитерских изделий, наличие удобного и функционального веб-приложения может стать ключевым фактором успеха. Оно позволяет не только привлекать новых клиентов, но и удерживать уже существующих, предлагая им новые возможности и удобства.

Таким образом, внедрение веб-приложения в деятельность кондитерской – это не только удобство для клиентов, но и возможность повысить эффективность работы и укрепить позицию на рынке кондитерских изделий.

Цель: разработка программного продукта – веб-приложение «Кондитерская».

Для достижения поставленной цели служат следующие задачи:

- Разработка дизайна.
- Разработка базы данных.
- Разработка алгоритма веб-приложения.
- Разработка прототипов веб-приложения.
- Разработка программного продукта веб-приложения.
- Разработка каталога продукции.
- Написание руководства пользователя.

1 Описание предметной области

Предметной областью курсовой работы является автоматизация процесса оформления заказа в кондитерской.

Кондитерская представляет собой магазин по производству и продаже выпечки. Осуществляют прием и выполнение заказов на торты и другие изделия.

Основной целью веб-приложения является упрощение взаимодействия клиента с каталогом продукции и создание интуитивно понятного интерфейса.

На главной странице будет представлено: основная информацию об кондитерской, адрес заведения, контактная информация для обратной связи, поиск, переход к каталогу продукции и возможность регистрации и авторизации пользователя.

Каталог будет представлен в виде карточек с информацией об кондитерском изделии: название, изображение, характеристика и цена.

Предметная область охватывает следующие группы пользователей:

- Пользователь.
- Администратор.

Пользователь – это человек, который использует веб-приложение кондитерской для просмотра и оформления заказа продукции.

Пользователь обладает правом:

1. Регистрация/авторизации в системе.
2. Ознакомиться с доступными кондитерскими изделиями в каталоге.
3. Перейти к подробным характеристикам продукции и добавить в корзину.
4. Оформить заказ.
5. Редактировать личные данные.

Администратор – это лицо ответственное за управление ассортиментом кондитерской продукции.

Администратор обладает правом:

1. Добавление, изменение и удаление информации о продукции и категории (название, изображение, характеристика, цены).

2. Просматривать информацию о пользователях.

3. Изменять статус заказа.

Администратор, пользователь и веб-приложение взаимодействуют между собой для обеспечения функциональности и удобства использования каталога продукции кондитерской. Вот описание взаимодействия между этими объектами:

1. Администратор:

– Авторизуется в веб-приложения с помощью личных учетных данных.

– Имеет доступ к панели администратора, где управляет содержимым каталога продукции.

– Добавляет, и удаляет продукцию, управляет ее характеристиками, изображениями и ценами.

– Управляет категориями продукции.

– Просматривает информацию о пользователях.

– Изменяет статус заказов.

2. Пользователь:

– Регистрируется/авторизуется в веб–приложении.

– Просматривает основную информацию на главной странице.

– Просматривает информацию о продукции и может воспользоваться поиском.

– Переходит на страницы с подробной характеристики о продукции.

– Добавляет продукцию в корзину.

– Оформляет заказ.

– Редактирует личные данные.

3. Веб-приложение:

– Обеспечивает доступ к автоматизации работы кондитерской через веб-браузер.

– Авторизует администратора, позволяя ему входить в систему и использовать функциональность веб-приложения.

– Отображает каталог товаров, результаты поиска и страницы товаров с соответствующими характеристиками и изображениями.

– Предоставляет возможность поиска товаров.

– Обрабатывает запросы пользователей, включая добавление и удаление товаров.

– Взаимодействует с базой данных для хранения информации о сущностях.

В соответствии с предметной областью «Автоматизация работы Кондитерской» можно выделить базовые сущности проектируемого программного продукта:

1. Пользователи.

2. Товары.

3. Заказ.

4. Элемент заказа.

5. Категория.

Атрибуты пользователя: код пользователя, фамилия, имя, номер телефона, логин, пароль, роль.

Атрибуты продукции: код продукции, название, описание, изображение, вес, цена, доступность, количество.

Атрибуты категории: код категории, название, доступность.

Атрибуты заказа: код заказа, дата, итоговая сумма, статус.

Атрибуты элемента заказа: код элемента заказа, количество, цена.

2 Анализ инструментальных средств разработки

В ходе создания веб-приложения «Кондитерская» необходимо использовать язык программирования для разработки кода и определиться какая система управления базами данных (далее – СУБД) будет использована.

Дизайн веб-приложения удобно сделать через онлайн-сервис Figma.

Figma – онлайн-редактор, в котором удобно проектировать интерфейсы, создавать макеты сайтов, мобильных приложений, презентации, иллюстрации, логотипы и анимацию. В редакторе можно настроить совместную работу, вносить и обсуждать правки, причём как в браузере, так и через приложение на компьютере. Популярен для разработки прототипа и дизайна сайта. В проекте используется для создания наглядного ожидаемого дизайна программного продукта и создание логотипа.

Веб-приложение будет содержать в себе информацию о сущностях – её необходимо хранить, изменять, структурировать и использовать. Это реализуется благодаря базе данных. Были рассмотрены следующие варианты реализации СУБД:

1. MySQL.
2. PostgreSQL.

MySQL – это одна из самых популярных СУБД на основе языка SQL (Structured Query Language). Она используется для хранения, управления и извлечения данных в различных приложениях, от веб-сайтов до корпоративных систем.

PostgreSQL – это мощная и расширяемая система управления реляционными базами данных (СУБД), которая обеспечивает эффективное хранение, организацию и управление структурированными данными.

В ходе анализа инструментальных средств разработки было проведено сравнение СУБД. В таблице 1 представлено сравнение двух СУБД.

Таблица 1 – Сравнение средств реализации базы данных

Характеристики/название	MySQL	PostgreSQL
Поддержка репликации	+	-

Продолжение таблицы 1

Инструменты для администрирования	+	-
Большое кол-во типов данных	+	+
Простота использования	+	-
Расширяемость	-	+
Производительность	+	+

В ходе сравнительной характеристики была выбрана СУБД MySQL, это обусловлено теми фактами как простота в использовании и интуитивно понятный интерфейс, что делает его доступным для разработчиков и даже новичков.

Для работы с базой данных нужен браузер, который будет передавать на сервер все команды.

PHPMyAdmin – веб-приложение с открытым кодом, написанное на языке PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL.

Для разработки программного продукта серверной части были рассмотрены следующие языки программирования:

1. JavaScript.
2. Php.

JavaScript – это мультипарадигменный язык программирования, который поддерживает объектно-ориентированный, императивный и функциональный стили. Наиболее широкое применение JavaScript находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

PHP (Hypertext Preprocessor) – это популярный серверный язык программирования, который используется для создания динамических веб-страниц и веб-приложений.

PHP позволяет разработчикам встраивать код прямо в HTML, что упрощает процесс создания интерактивных и функциональных веб-сайтов.

В ходе анализа инструментальных средств разработки было проведено сравнение языков программирования. В таблице 2 представлено сравнение двух языков программирования.

Таблица 2 – Сравнение языков программирования

Характеристики/название	PHP	JavaScript
Фреймворки	+	+
Простой синтаксис	+	+
Простота развертывания	+	-
Поддержка баз данных	+	+

В ходе сравнительной характеристики был выбран язык программирования php, так как он удобен в использовании.

Для упрощения работы с php нужно использовать фреймворк. В данном проекте будет использоваться фреймворк Laravel.

Laravel – это PHP фреймворк, предназначенный для создания веб-приложений любой сложности. Он используется как начинающими, так и опытными разработчиками благодаря богатому набору инструментов и простоте в использовании. Laravel делает разработку более быстрой и эффективной, обеспечивая высокую производительность и безопасность приложений.

Так же в проекте будет использован фреймворк Bootstrap для создания пользовательского интерфейса.

Bootstrap – это набор инструментов для создания адаптивных и стильных веб-приложений, который включает готовые компоненты, шаблоны и утилиты. Он упрощает разработку, делая её быстрой и удобной, особенно для начинающих разработчиков.

В проекте необходимо использовать среду разработки. Для этого было проведено сравнение IDE.

1. Visual Studio Code.
2. Eclipse.
3. PyCharm

Visual Studio Code – это бесплатный и открытый кроссплатформенный текстовый редактор, разработанный корпорацией Microsoft. Он предназначен для написания и отладки кода на различных языках программирования, включая PHP.

Eclipse – это мощная и популярная интегрированная среда разработки (IDE) для программирования на языке Java и других языках. Она поддерживает разработку на PHP, C/C++, Python и других языках.

PyCharm – кроссплатформенная интегрированная среда разработки для языка программирования Python, разработанная компанией JetBrains. Предоставляет пользователю комплекс средств для написания кода и визуальный отладчик.

Таблица 3 – Сравнение IDE для разработки программного продукта

Характеристики/название	Visual Studio Code	Eclipse	PyCharm
Поддержка языков программирования	+	+	+
Интуитивно понятный интерфейс	+	-	-
Кроссплатформенность	+	+	+
Работа с большими проектами	+	+	+
Возможности настройки	+	+	+

В результате сравнения IDE для разработки программного продукта было принято решение использовать Visual Studio Code.

Visual Studio Code является наиболее быстрым редактором с отличной поддержкой множества языков программирования через расширения.

Eclipse не будет использоваться из-за своего менее удобного интерфейса, что может усложнить разработку. PyCharm не подходит, так как он ориентирован исключительно на Python и не предоставляет такой же универсальной поддержки других языков, как в Visual Studio Code.

CASE-средства – это методы и технологии, которые позволяют проектировать различные программные продукты (базы данных) и автоматизировать их создание.

Для данного проекта будет использован Draw.io и MySQL Workbench.

Draw.io – это кроссплатформенный онлайн-редактор для создания диаграмм и схем. Он позволяет пользователям создавать, например, блок-схемы, прототипы, UML-диаграммы, организационные схемы и сетевые диаграммы.

MySQL Workbench – инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в

единое бесшовное окружение для системы баз данных MySQL. В проекте используется для создания ER-диаграммы базы данных.

Для создания программного продукта было решено использовать следующие средства:

1. Для создания структурных схем, контекстной и диаграмм декомпозиции использовались CASE-средства – Draw.io.
2. Для создания ER-диаграммы базы данных используется MySQL Workbench.
3. Для работы с базой данных использовался браузер PHPMyAdmin.
4. Для создания серверной части программного продукта используется язык программирования php и фреймворк Laravel.
5. Для создания адаптивного пользовательского интерфейса используется Bootstrap.
6. Для разработки дизайна веб-приложения использовался онлайн-сервис – Figma.
7. Для разработки программного продукта было принято решение использовать IDE Visual Studio Code.

3 Техническое задание

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

1. Введение.
2. Основания для разработки.
3. Назначение системы.
4. Требования к системе.
5. Требования к техническому обеспечению.
6. Требования к программному обеспечению.
7. Организационно-технические требования.

Техническое задание на разработку веб-приложения представлено в приложении А.

4 Проектирование веб-приложения

4.1 Структурная схема веб-приложения

На этапе построения структурных схем веб-приложения были построенны диаграммы: прецедентов Uses CASE, диаграмма деятельности, диаграмма компонентов, диаграмма развертывания.

На рисунке 1 представленна диаграмма прецедентов Uses CASE.

Диаграмма прецедентов Uses CASE позволяет показать пользователей создаваемой системы и основные действия, которые актеры могут в этой системе выполнять.

Рисунок 1 – Диаграмма прецедентов Uses CASE

На рисунке 2 представленна диаграмма деятельности.

Диаграмма деятельности (activity diagram) – это графическое представление процессов и задач, выполняемых в рамках определенного проекта или деятельности.

В диаграмме деятельности представлен процесс оформления заказа.

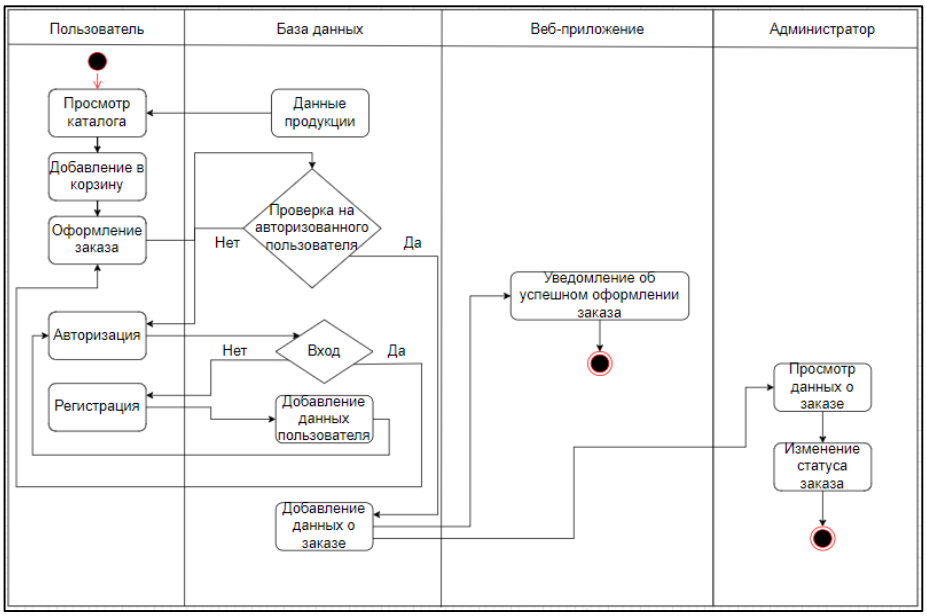


Рисунок 2 – Диаграмма деятельности

На рисунке 3 представленна диаграмма взаимодействия.

Диаграмма взаимодействия – это диаграмма, на которой представлено взаимодействие, состоящее из множества объектов и отношений между ними, включая и сообщения, которыми они обмениваются.

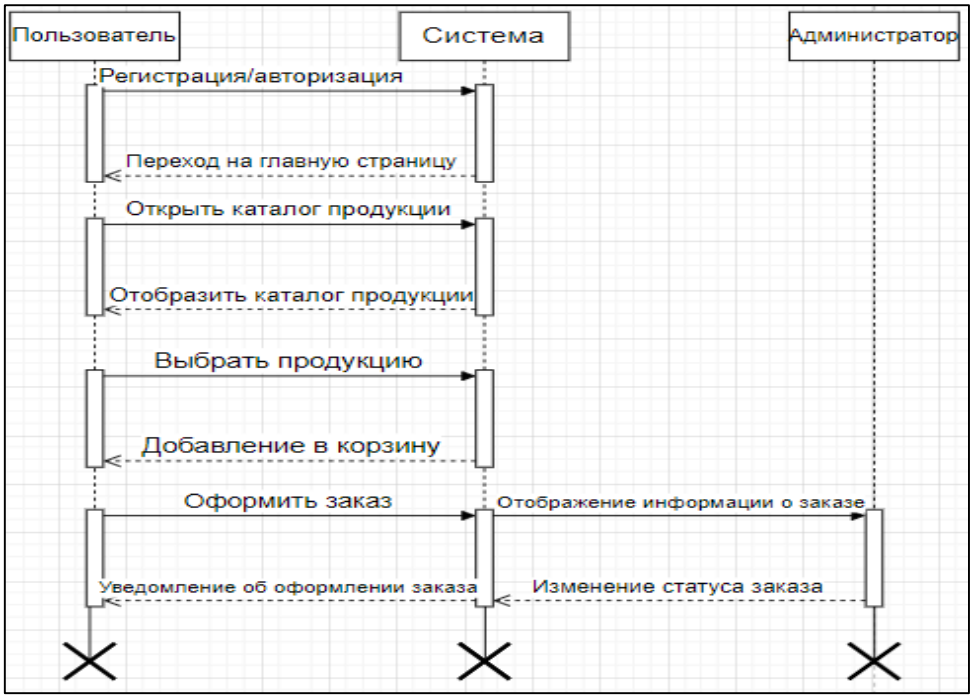


Рисунок 3 – Диаграмма взаимодействия

На рисунке 4 представленна диаграмма компонентов.

Диаграмма компонентов в UML используется для визуализации структуры системы, отображая её компоненты, их интерфейсы и взаимосвязи. Она помогает понять, как различные части системы взаимодействуют друг с другом и как они организованы в рамках архитектуры.

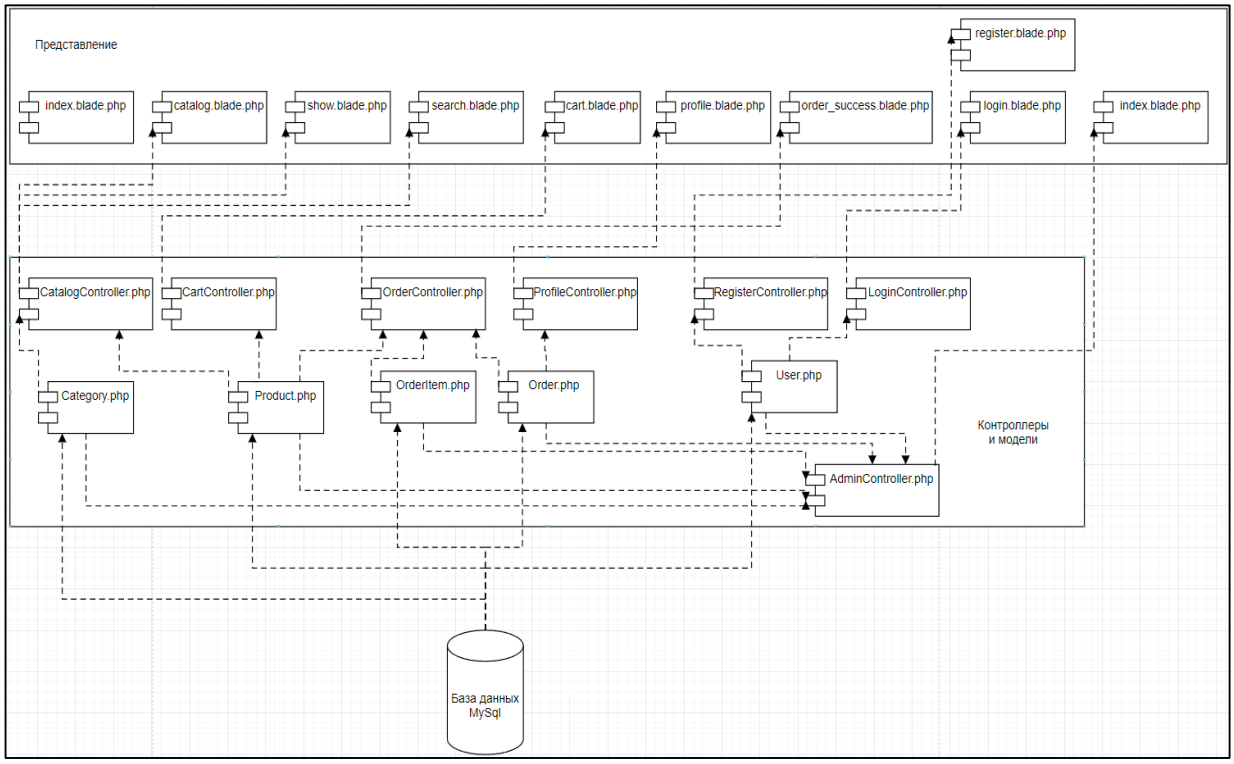


Рисунок 4 – Диаграмма компонентов

На рисунке 5 представленна диаграмма развертывания.

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения.

Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы.

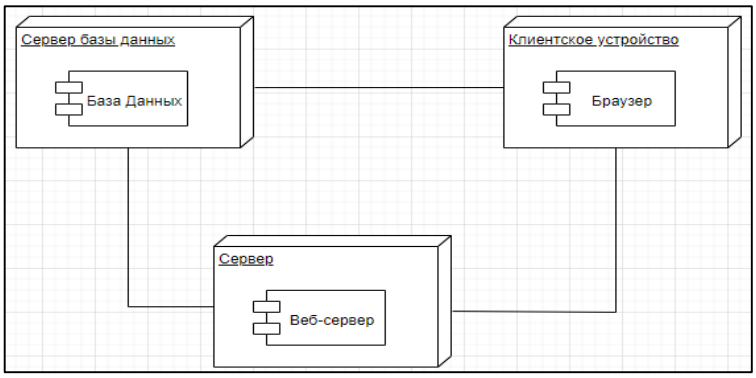


Рисунок 5 – Диаграмма развертывания

4.2 Функциональная схема веб-приложения

На этапе построения функциональных схем веб-приложения были построенны диаграммы: контекстная диаграмма созданной в нотации IDEF0 (A0), диаграмма декомпозиций (A1), диаграмма классов, диаграмма потоков данных DFD.

Контекстная диаграмма представляет общее описание системы и её взаимодействия с внешней средой.

На рисунке 6 контекстная диаграмма представляет собой процесс оформления заказа.

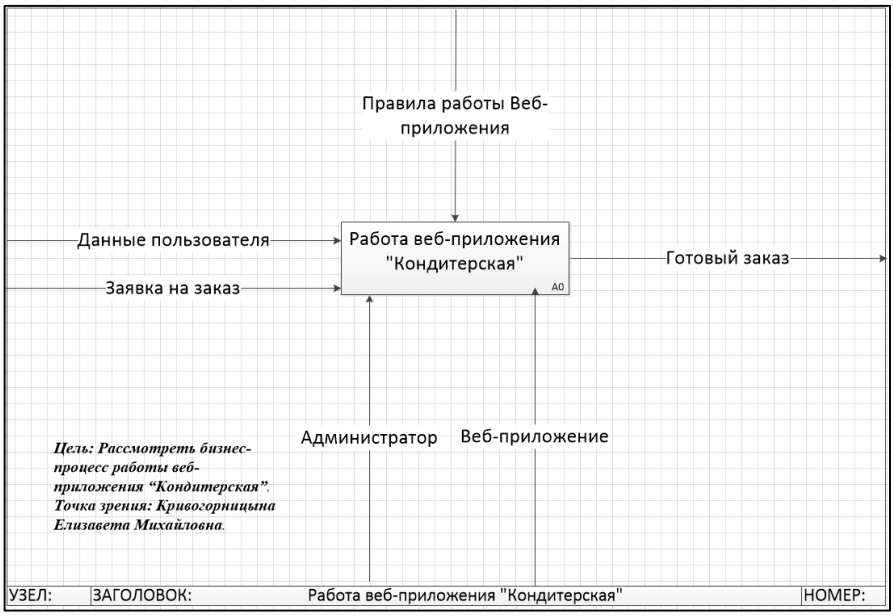


Рисунок 6 – Контекстная диаграмма созданной в нотации IDEF0 (A0)

На рисунке 7 представленна диаграмма декомпозиции.

Диаграмма декомпозиции детализирует отдельные элементы системы и связи между ними.

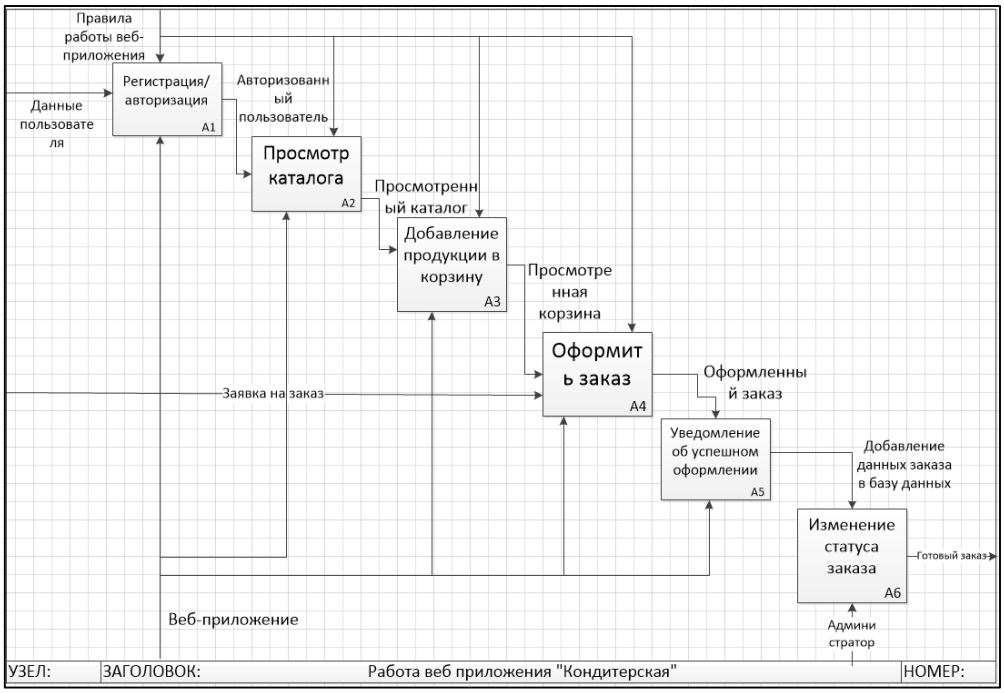


Рисунок 7 – Диаграмма декомпозиции

На рисунке 8 представленна диаграмма потоков данных DFD.

Диаграмма потоков данных (DFD) – это графическое представление потока данных в веб-приложении. С помощью DFD можно описывать входящие и выходящие потоки данных и хранилища этих данных.

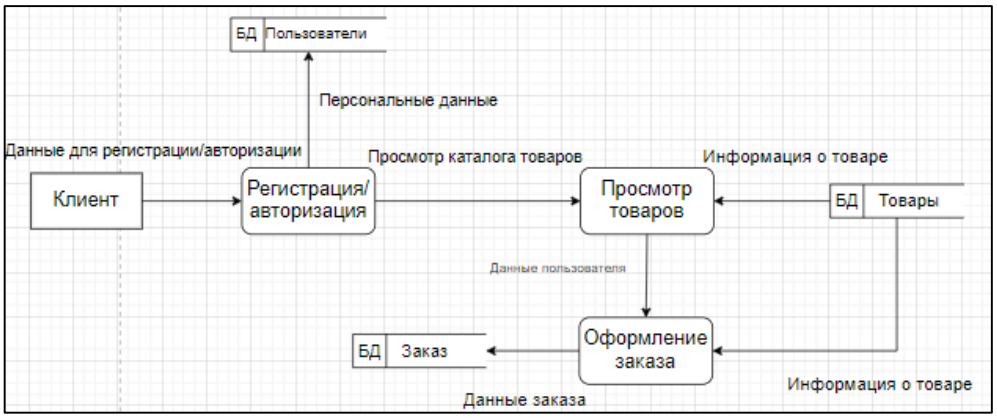


Рисунок 8 – Диаграмма потоков данных DFD

На рисунке 9 представленна диаграмма классов.

Диаграмма классов в UML представляет структуру системы, отображая классы, их атрибуты, методы и отношения между ними. Она является основным инструментом для моделирования объектно-ориентированного проектирования.

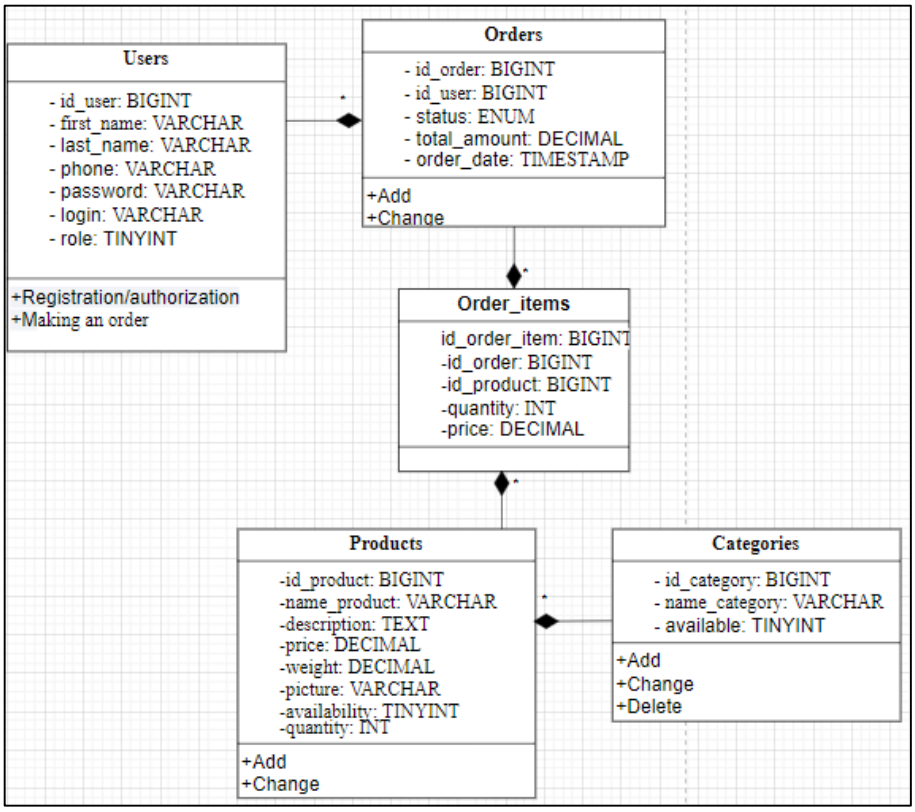


Рисунок 9 - Диаграмма классов

4.3 Проектирование базы данных

На этапе проектирования базы данных были постренны модели: инфологическая модель и ER модель базы данных.

На рисунке 10 представленна инфологическая модель базы данных. В модели схематично отображены сущности веб-приложения, их атрибуты и связи между ними. Так, в прямоугольник отображены сущности, в овалах отображены атрибуты сущностей, робами изображены связи между сущностями.

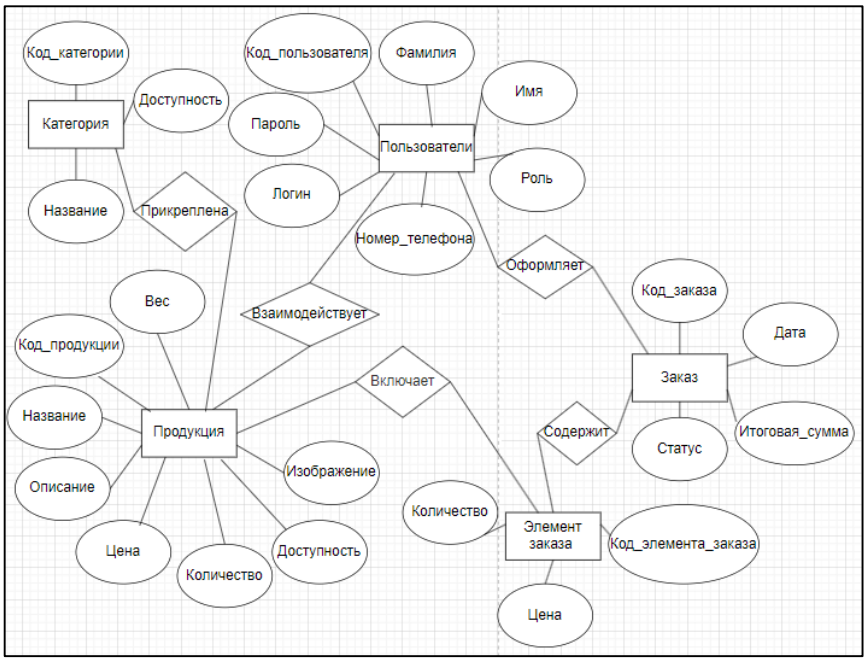


Рисунок 10 - Инфологическая модель базы данных

На рисунке 11 ER-модели базы данных представлены 5 таблиц, связи и типы данных.

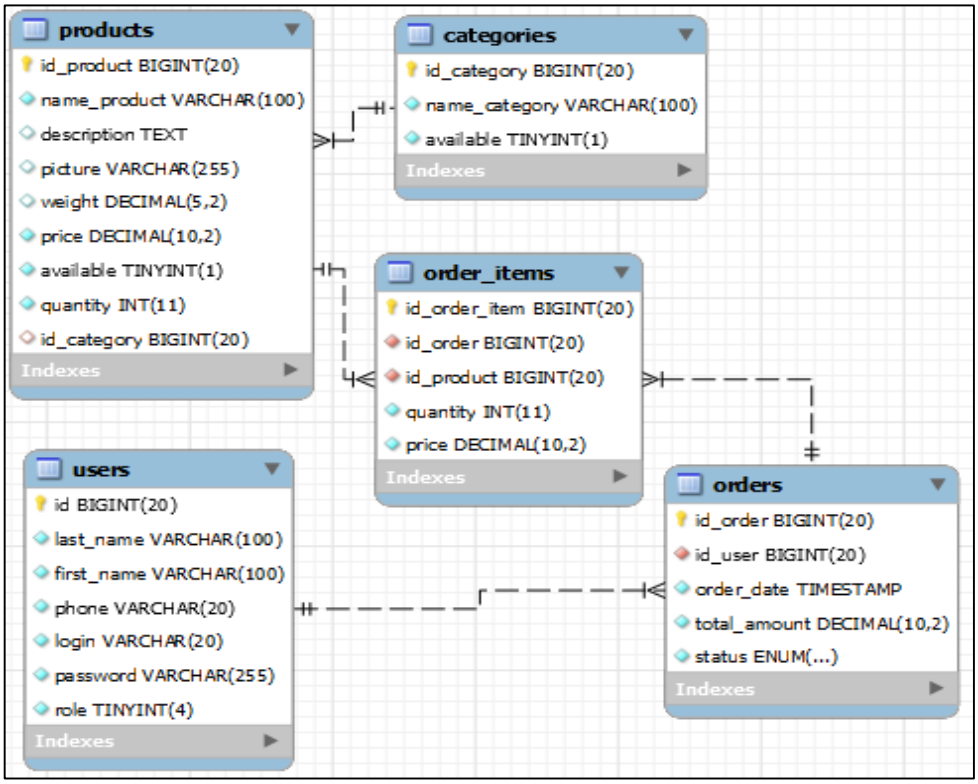


Рисунок 11 - ER-модель базы данных

Уровень нормализации – 3.

В таблицах с 5 по 9 представлены поля, тип данных поля и описание.

Таблица 4 – Таблицы ER-модели

Таблица	Описание
users	Таблица Пользователи
orders	Таблица Заказы
order_items	Таблица Элемент заказа
products	Таблица Продукты
category	Таблица Категория

Таблица 5 – Таблица «users»

Поле	Тип данных	Описание
id_user	BIGINT	Код пользователя
last_name	VARCHAR(255)	Имя
first_name	VARCHAR(255)	Фамилия
phone	VARCHAR(255)	Номер телефона
login	VARCHAR(255)	Пароль
password	VARCHAR(255)	Логин
Role	TINYINT(1)	Роль 0 – пользователь 1 – администратор

Таблица 6 – Таблица «orders»

Поле	Тип данных	Описание
id_order	BIGINT	Код заказа
order_date	TIMESTAMP	Дата заказа
total_amount	DECIMAL(10,2)	Итоговая сумма
status	ENUM('Создан', 'Принят', 'В процессе', 'Готов к выдаче', 'Отменён')	Статус
id_user	BIGINT	Код пользователя

Таблица 7 – Таблица «order_items»

Поле	Тип данных	Описание
id_order_item	BIGINT	Код элемента заказа
id_order	BIGINT	Код зака
id_product	BIGINT	Код продукции
quantity	INT	Количество
price	DECIMAL(10,2)	Цена

Таблица 8 – Таблица «products»

Поле	Тип данных	Описание
id_product	BIGINT	Код придукции
name_product	VARCHAR(100)	Название прилокции
description	TEXT	Описание
picture	VARCHAR(255)	Изображение
weight	DECIMAL(5,2)	Вес
price	DECIMAL(10,2)	Цена
available	TINYINT(1)	Доступность
quantity	INT	Количество
id_category	BIGINT	Код категории

Таблица 9 – Таблица «categories»

Поле	Тип данных	Описание
id_category	BIGINT	Код категории
name_category	VARCHAR(100)	Название категории
available	TINYINT	Доступность

Для разработки ER-модели был использован инструмент MySQL Workbench.

После завершения проектирования базы данных веб-приложения «Кондитерская» получиться готовая схема, по которой будет разрабатываться база данных.

4.4 Проектирование интерфейса

На этапе проектирования интерфейса веб-приложения были разработаны пртотипы.

Для построения пртотипов было использованно CASE-средство: Draw.io.

Важным этапом в процессе проектирования веб-приложения для кондитерской является создание прототипов. Они позволяют наглядно представить, как будет выглядеть интерфейс и как будет работать приложение.

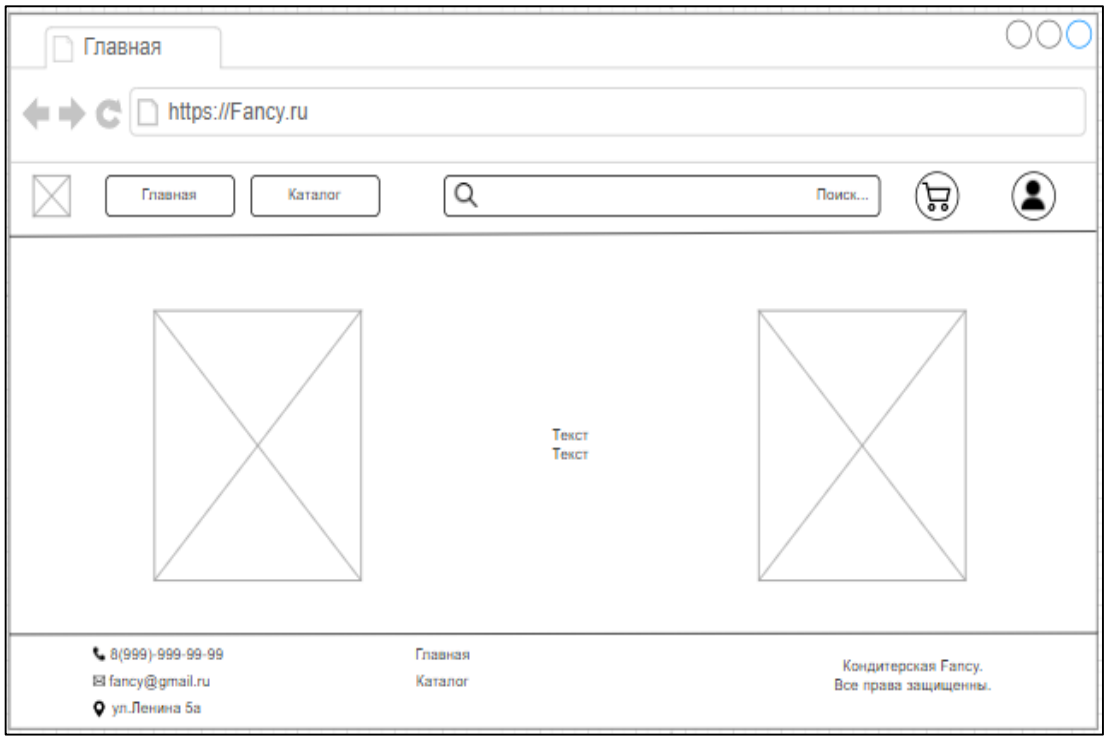


Рисунок 12 – Прототип «Главная страница»

На главной странице расположена краткая информация о кондитерской.

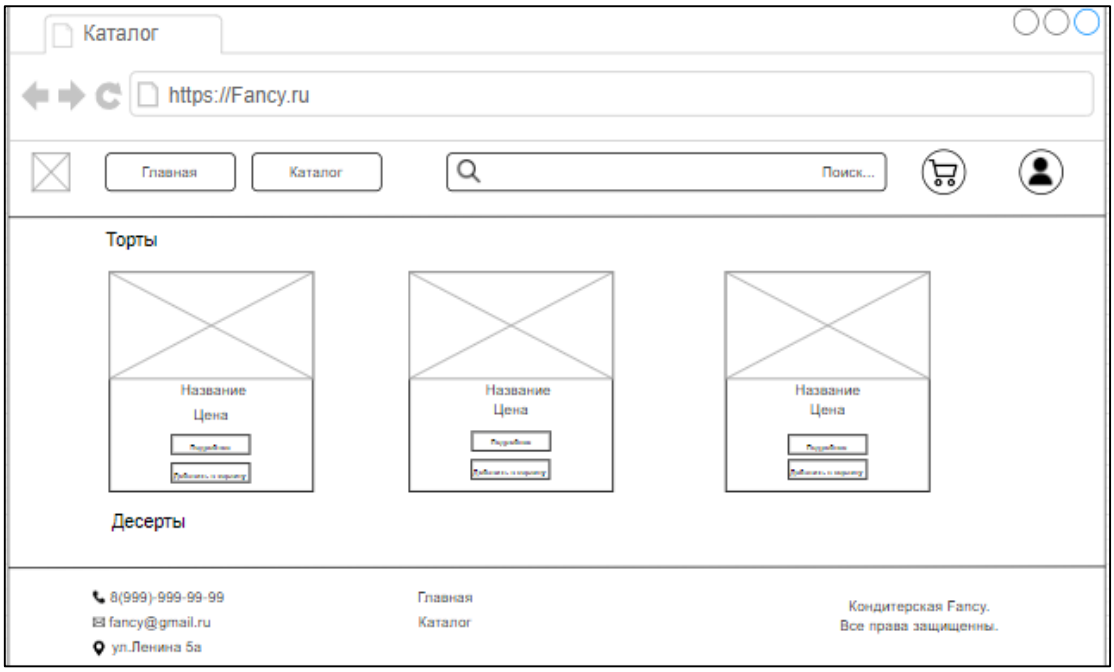


Рисунок 13 – Прототип «Каталог»

На вкладке каталога пользователь может ознакомится с доступным ассортиментом и перейти на вкладку подробнее или добавить в корзину продукцию.

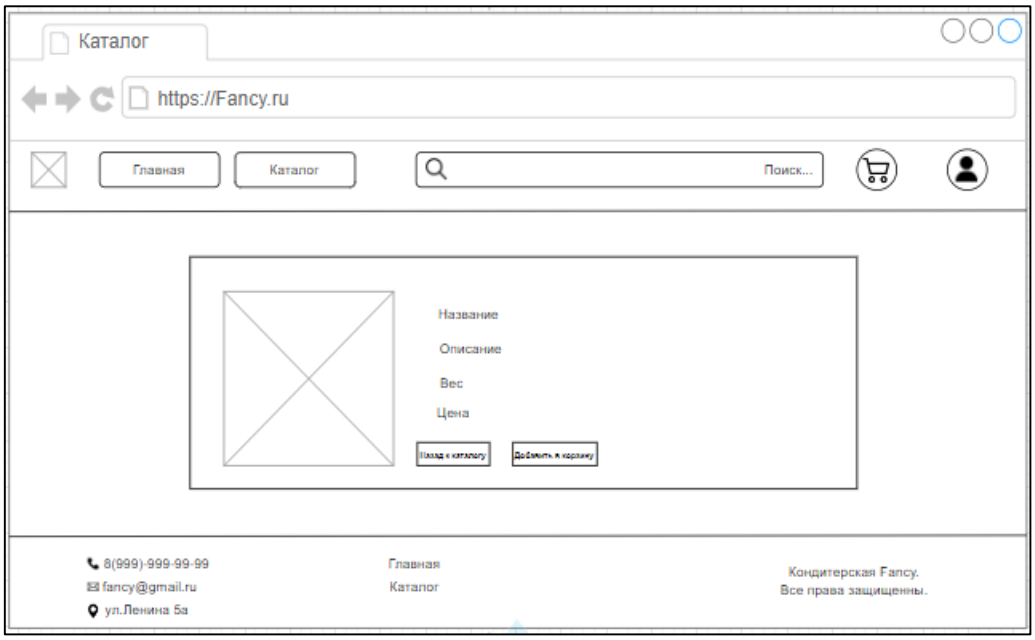


Рисунок 14 – Прототип «Карточка продукции»

При переходе на страницу с подробным описание товара пользователь так же может добавить продукцию в корзину и более подробно ознакомится с характеристиками.

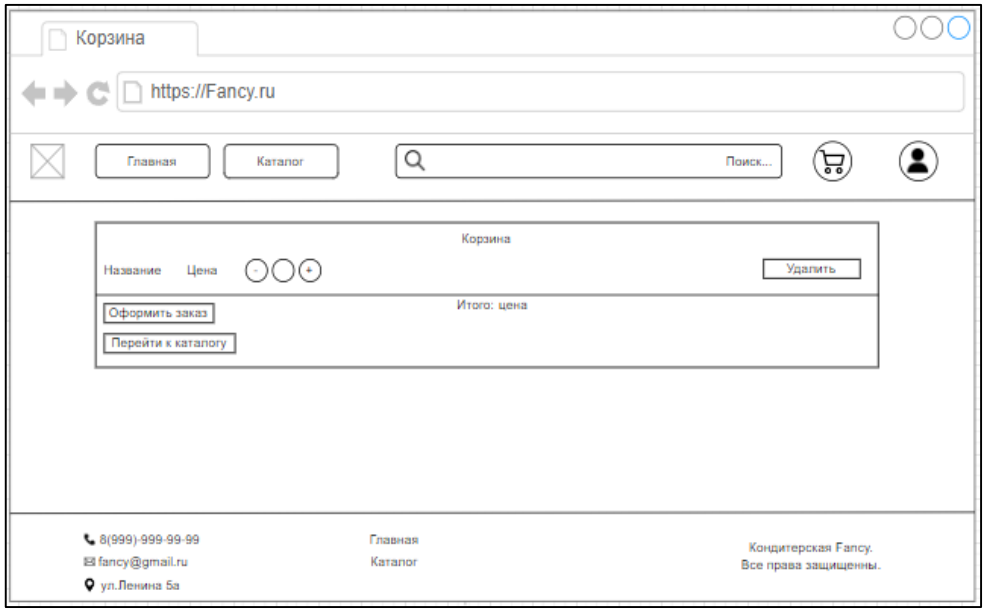


Рисунок 15 – Прототип «Корзина»

При переходе на вкладку корзина будет доступна продукция, которую пользователь добавил. Он может изменять количество, удалить продукцию, оформить заказ и просмотреть итоговую сумму.

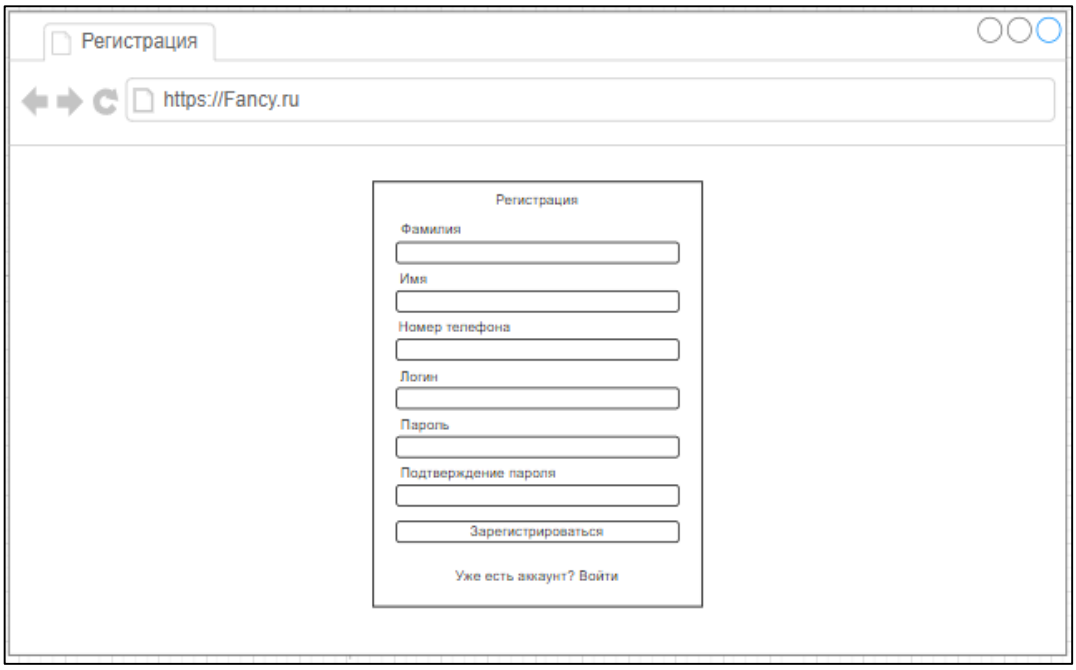


Рисунок 16 – Прототип «Регистрация»

Пользователь проходит регистрацию для оформления заказа.

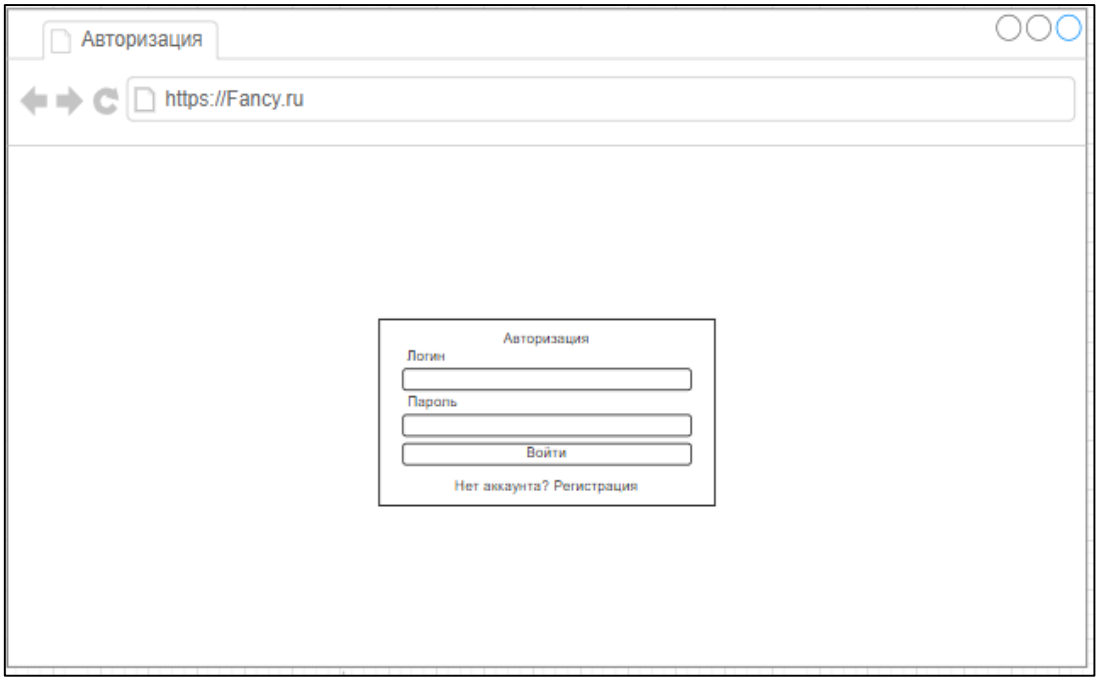


Рисунок 17 – Прототип «Авторизация»

После регистрации пользователь авторизуется для просмотра личного кабинета.

Рисунок 18 – Прототип «Оформление заказа»

После успешного оформления заказа пользователю показывается страница об успешном оформлении заказа.

Рисунок 19 – Прототип «Личный кабинет»

При переходе в личный кабинет пользователь может изменить свои данные и посмотреть историю заказа.

Создание прототипов веб-приложения для кондитерской дало возможность представить все необходимые функции, структурировать информацию для пользователей и наглядно показать как будет выглядеть интерфейс.

5 Разработка веб-приложения

5.1 Разработка интерфейса веб-приложения

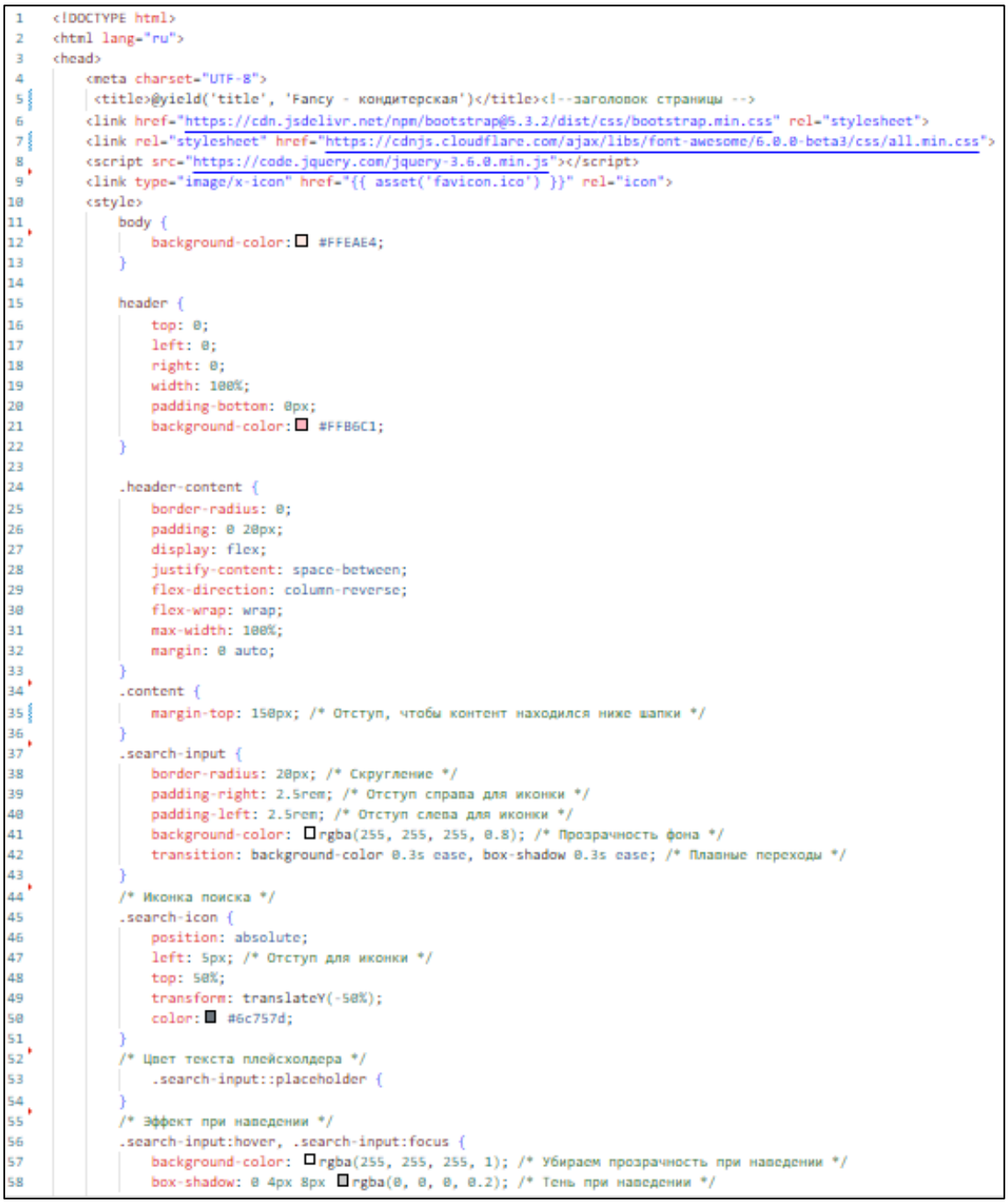
Для создания веб-приложения для кондитерской используется язык программирования PHP и фреймворк Laravel.

PHP – один из самых востребованных языков для разработки веб-приложений. А Laravel предоставляет широкий набор инструментов и функций, которые облегчают процесс разработки, повышают эффективность и соответствуют современным стандартам веб-разработки.

Интерфейс веб-приложения разработан с использованием HTML и CSS, а также интеграцией с Bootstrap и Font Awesome для упрощения создания адаптивного и визуально привлекательного дизайна. Основные элементы интерфейса включают шапку, подвал и контентную часть главной страницы.

Шаблоны (layouts) в Laravel используются для создания общей структуры веб-страниц, которая включает в себя основные компоненты, такие как шапка (header), подвал (footer), навигационное меню и контентная область (Content). Это позволяет избежать дублирования кода и упрощает поддержку приложения.

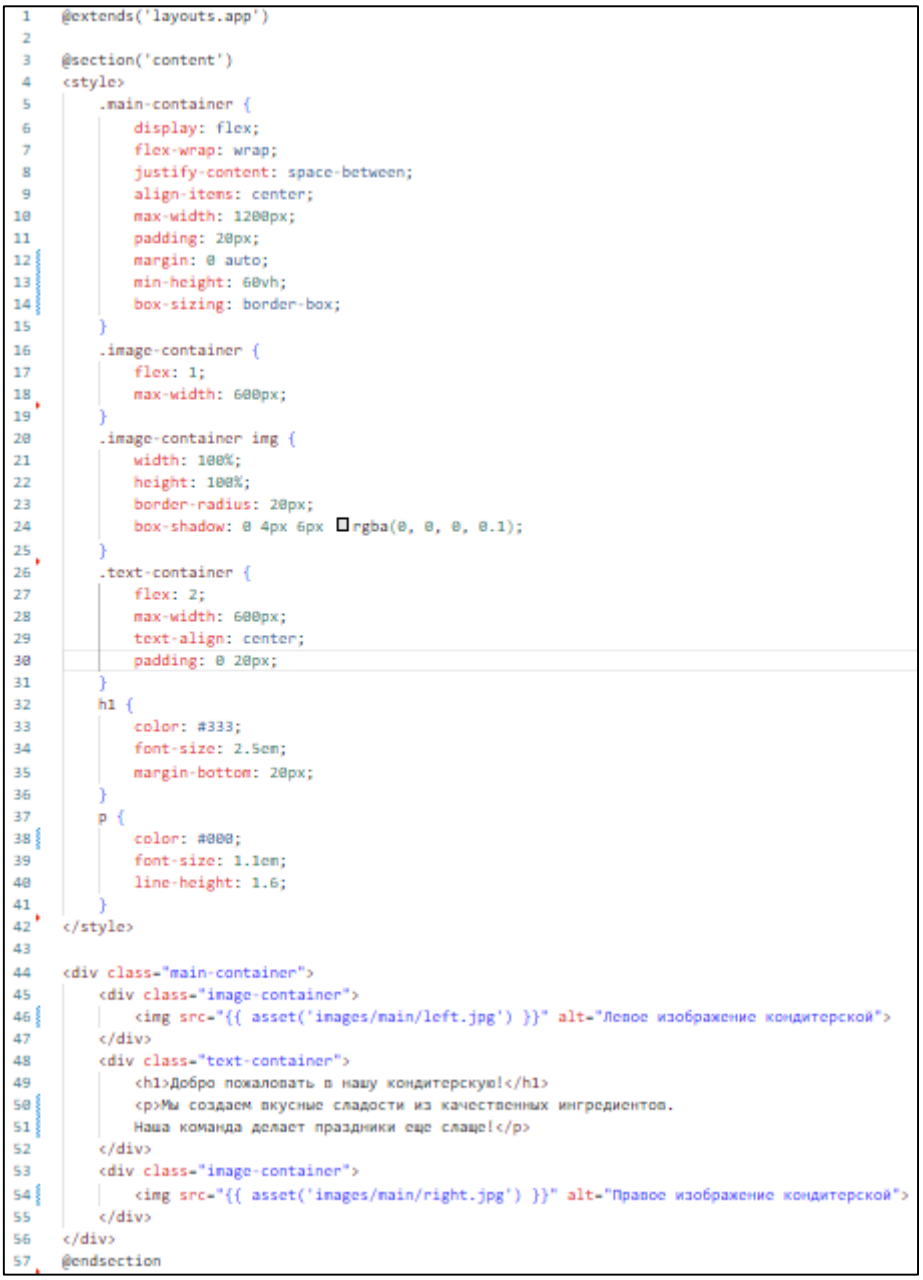
На рисунке 20 представлена часть кода для шаблона (layouts), которая демонстрирует, как используются Blade-шаблоны для создания общей структуры страниц. Эти шаблоны позволяют разделить интерфейс на компоненты, такие как шапка и подвал, что значительно упрощает разработку.



```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <title>@yield('title', 'Fancy - кондитерская')</title><!--заголовок страницы -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <link type="image/x-icon" href="{{ asset('favicon.ico') }}" rel="icon">
    <style>
        body {
            background-color: #FFEAE4;
        }

        header {
            top: 0;
            left: 0;
            right: 0;
            width: 100%;
            padding-bottom: 0px;
            background-color: #FFB6C1;
        }

        .header-content {
            border-radius: 0;
            padding: 0 20px;
            display: flex;
            justify-content: space-between;
            flex-direction: column-reverse;
            flex-wrap: wrap;
            max-width: 100%;
            margin: 0 auto;
        }
        .content {
            margin-top: 150px; /* Отступ, чтобы контент находился ниже шапки */
        }
        .search-input {
            border-radius: 20px; /* Скругление */
            padding-right: 2.5rem; /* Отступ справа для иконки */
            padding-left: 2.5rem; /* Отступ слева для иконки */
            background-color: rgba(255, 255, 255, 0.8); /* Прозрачность фона */
            transition: background-color 0.3s ease, box-shadow 0.3s ease; /* Плавные переходы */
        }
        /* Иконка поиска */
        .search-icon {
            position: absolute;
            left: 5px; /* Отступ для иконки */
            top: 50%;
            transform: translateY(-50%);
            color: #6c757d;
        }
        /* Цвет текста плейсхолдера */
            .search-input::placeholder {
        }
        /* Эффект при наведении */
        .search-input:hover, .search-input:focus {
            background-color: rgba(255, 255, 255, 1); /* Убираем прозрачность при наведении */
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /* Тень при наведении */
```

Рисунок 20 – Часть HTML кода шаблона (layouts)

На рисунке 21 представлен blade шаблон для главной страницы с использованием шаблона (layouts).



```blade
@extends('layouts.app')

@section('content')
<style>
    .main-container {
        display: flex;
        flex-wrap: wrap;
        justify-content: space-between;
        align-items: center;
        max-width: 1200px;
        padding: 20px;
        margin: 0 auto;
        min-height: 60vh;
        box-sizing: border-box;
    }
    .image-container {
        flex: 1;
        max-width: 600px;
    }
    .image-container img {
        width: 100%;
        height: 100%;
        border-radius: 20px;
        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    }
    .text-container {
        flex: 2;
        max-width: 600px;
        text-align: center;
        padding: 0 20px;
    }
    h1 {
        color: #333;
        font-size: 2.5em;
        margin-bottom: 20px;
    }
    p {
        color: #000;
        font-size: 1.1em;
        line-height: 1.6;
    }
</style>

<div class="main-container">
    <div class="image-container">
        <img src="{{ asset('images/main/left.jpg') }}" alt="Левое изображение кондитерской">
    </div>
    <div class="text-container">
        <h1>Добро пожаловать в нашу кондитерскую!</h1>
        <p>Мы создаем вкусные сладости из качественных ингредиентов.
        Наша команда делает праздники еще слаще!</p>
    </div>
    <div class="image-container">
        <img src="{{ asset('images/main/right.jpg') }}" alt="Правое изображение кондитерской">
    </div>
</div>
@endsection
```

Рисунок 21 – Blade шаблон главной страницы

На рисунке 22 представлен результат главной страницы то, как она выглядит с использованием шаблона (layouts) и применением стилей.

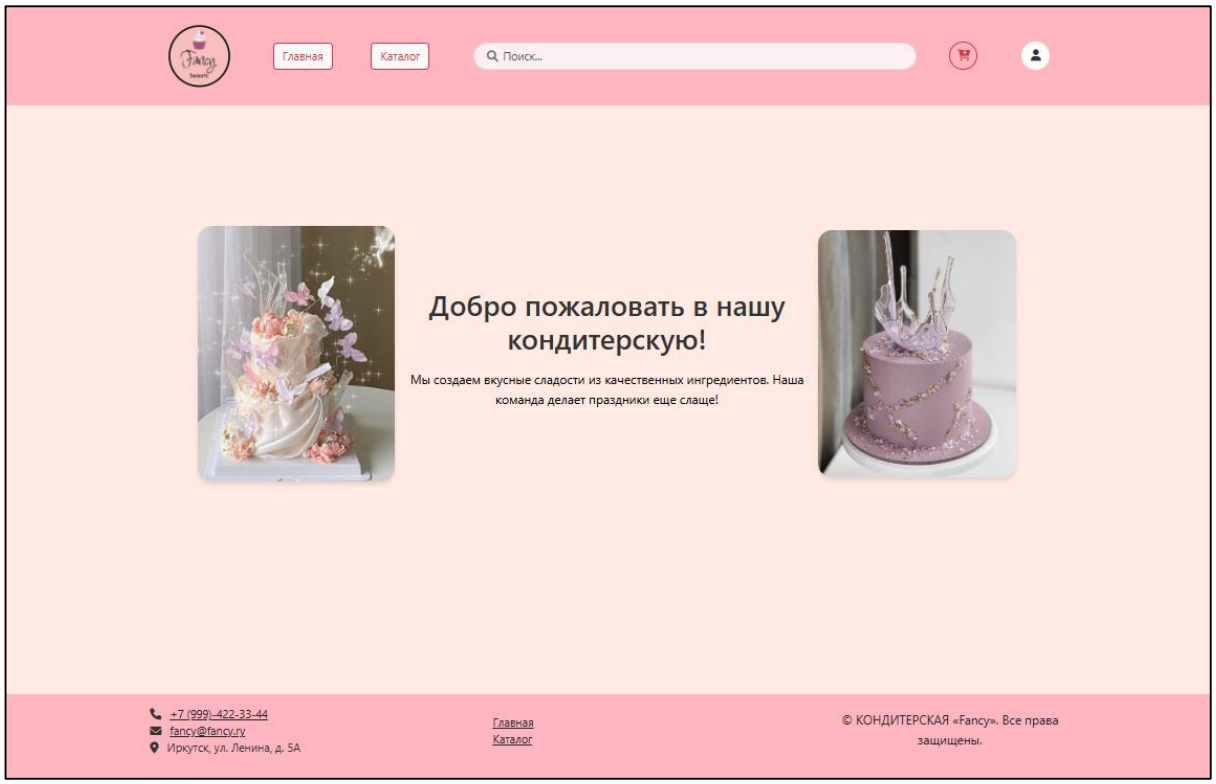


Рисунок 22 – Главная страница

5.2 Разработка базы данных веб-приложения

Для разработки веб-приложения управления данными и обеспечения их сохранности и целостности выбрана реляционная система СУБД MySQL, которая была выбранная по ряду причин описанные в разделе «Анализ инструментальных средств разработки».

При создании базы данных была использована ER-модель (рисунок 11) приведенная к 3 нормальной форме. База данных создана и настроена с использованием миграции в фреймворке Laravel. Для создании миграции используется команда: php artisan make:migration.

На рисунке 23 представлены миграции, которые хранятся в папке database/ migrations.

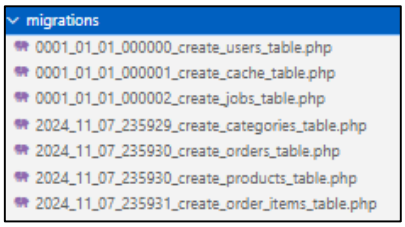


Рисунок 23 – Файлы миграции

На рисунке 24 представлена миграция таблицы «users».

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('last_name', 100); // Фамилия
            $table->string('first_name',100); // Имя
            $table->string('phone',20)->unique(); // Телефон
            $table->string('login',20)->unique(); // Логин
            $table->string('password');
            $table->tinyinteger('role')->default(0); // Роль по умолчанию 0
            $table->timestamps();
        });
```

Рисунок 24 – Миграция таблицы «users»

На рисунке 25 представлена миграция таблицы «categories».

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateCategoriesTable extends Migration
{
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id('id_category');
            $table->string('name_category', 100);
            $table->boolean('available')->default(true);
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('categories');
    }
}

```

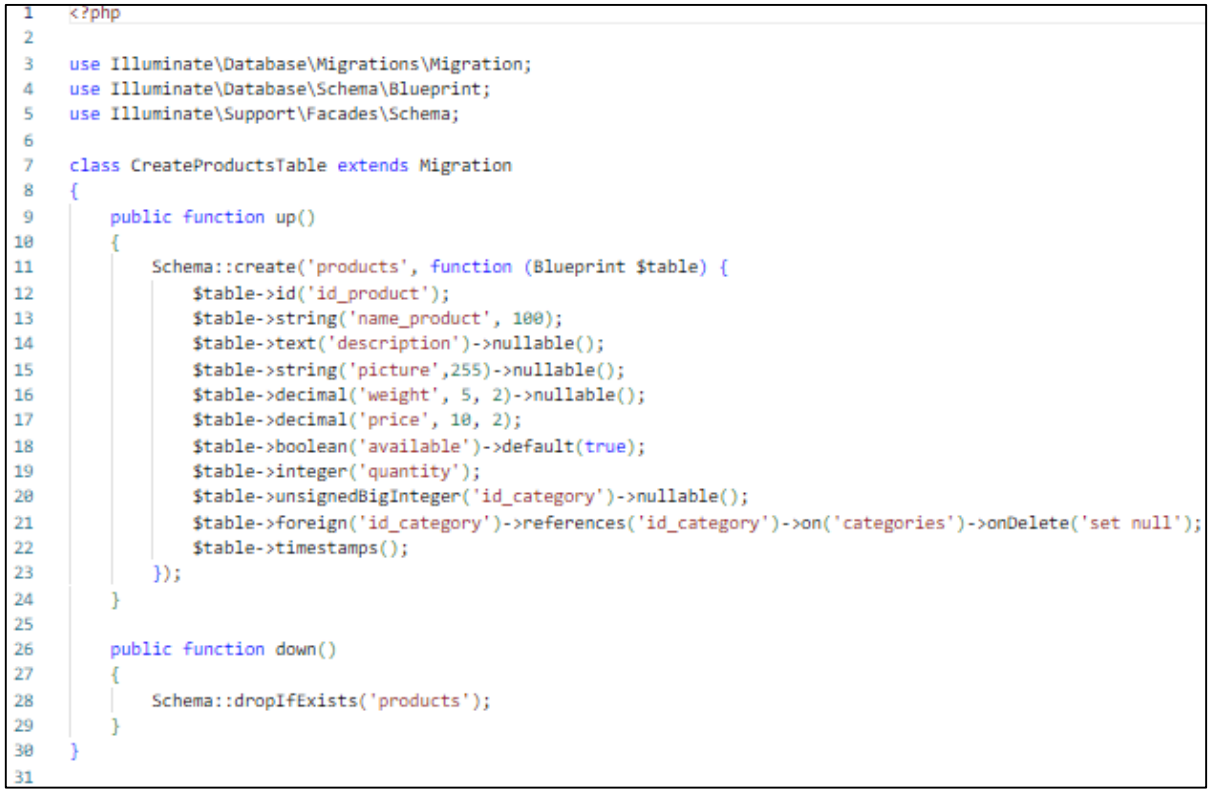
Рисунок 25 – миграция таблицы «categories»

На рисунке 26 представлена миграция таблицы «orders».

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateOrdersTable extends Migration
{
    public function up()
    {
        Schema::create('orders', function (Blueprint $table) {
            $table->id('id_order');
            $table->unsignedBigInteger('id_user');
            $table->timestamp('order_date')->default(DB::raw('CURRENT_TIMESTAMP'));
            $table->decimal('total_amount', 10, 2);
            $table->enum('status', ['Создан', 'Принят', 'В процессе', 'Готов к выдаче', 'Отменён'])->default('Создан');
            $table->foreign('id_user')
            ->references('id')
                ->on('users')
                ->onDelete('cascade');
            $table->timestamps();
        });
    }


    public function down()
    {
        Schema::dropIfExists('orders');
    }
}
```

Рисунок 26 – Миграция таблицы «orders»

На рисунке 27 представлена миграция таблицы «products».



```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateProductsTable extends Migration
{
    public function up()
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id('id_product');
            $table->string('name_product', 100);
            $table->text('description')->nullable();
            $table->string('picture',255)->nullable();
            $table->decimal('weight', 5, 2)->nullable();
            $table->decimal('price', 10, 2);
            $table->boolean('available')->default(true);
            $table->integer('quantity');
            $table->unsignedBigInteger('id_category')->nullable();
            $table->foreign('id_category')->references('id_category')->on('categories')->onDelete('set null');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('products');
    }
}

```

Рисунок 27 – Миграция таблицы «products»

На рисунке 28 представлена миграция таблицы «order_items».

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateOrderItemsTable extends Migration
{
    public function up()
    {
        Schema::create('order_items', function (Blueprint $table) {
            $table->id('id_order_item');
            $table->unsignedBigInteger('id_order');
            $table->unsignedBigInteger('id_product');
            $table->integer('quantity');
            $table->decimal('price', 10, 2);
            $table->foreign('id_order')->references('id_order')->on('orders')->onDelete('cascade');
            $table->foreign('id_product')->references('id_product')->on('products')->onDelete('cascade');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('order_items');
    }
}

```

Рисунок 28 – Миграция таблицы «order_items»

5.3 Разработка веб-приложения

Для подключения к базе данных и выполнения миграции таблиц осуществляется с помощью файла «.env».

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=22105_Fancy_kondit
DB_USERNAME=root
DB_PASSWORD=
```

Рисунок 29 – Часть env файла с подключением к базе данных

Следующим шагом для разработки веб-приложения «Кондитерская» это создание моделей. Модель описывает структуру и поведение соответствующей таблицы в базе данных. В фреймворке Laravel модель создается с помощью команды: php artisan make:model.

На рисунке 30 представлена модель «User» представляет пользователей системы. Она наследуется от класса Authenticatable, что позволяет использовать её для аутентификации и управления пользовательскими данными.

```php
<?php

namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use Notifiable;

    protected $fillable = [
        'last_name',
        'first_name',
        'phone',
        'login',
        'password',
        'role',
    ];

    protected $hidden = [
        'password',
    ];

    public function isAdmin()
    {
        return $this->role === 1;
    }
}
```

Рисунок 30 – Модель «User»

Контроллеры в Laravel позволяют организовать логику обработки запросов в отдельные классы. Это делает код более структурированным и поддерживаемым. Для создания контроллеров в Laravel используется команда: php artisan make:controller name.

На рисунке 31 представлен «RegisterController» контроллер обрабатывает регистрацию пользователей, валидирует данные и создает учетную запись. Весь код контроллера представлен в приложении Б.

```php
    public function register(Request $request)
    {
        $request->validate([
            'last_name' => 'required|string|max:100',
            'first_name' => 'required|string|max:100',
            'middle_name' => 'nullable|string|max:100',
            'phone' => 'required|string|max:20|unique:users,phone',
            'login' => 'required|string|max:20|unique:users,login',
            'password' => 'required|string|min:8|confirmed',
        ], [
            'phone.unique' => 'Номер телефона уже зарегистрирован',
            'login.unique' => 'Логин занят',
            'password.confirmed' => 'Введённые пароли не совпадают',
            'password.min' => 'Длина пароля не может быть меньше 8 символов',
        ]);

        User::create([
            'last_name' => $request->last_name,
            'first_name' => $request->first_name,
            'middle_name' => $request->middle_name,
            'phone' => $request->phone,
            'login' => $request->login,
            'password' => Hash::make($request->password),
        ]);

        return redirect()->route('login')->with('status', 'Регистрация успешна!');
    }
```

Рисунок 31 – «RegisterController» контроллер

Для правильной работы контроллеров в Laravel необходимо настроить маршруты, которые определяют, какие запросы будут обрабатываться теми или

иными методами контроллеров. Маршруты объявляются в файле routes/web.php. Они настроены для обработки различных запросов, таких как регистрация, авторизация, управление профилем, работа с каталогом, администрирование и корзина.

На рисунке 32 представлены часть маршрутов для пользователя.

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Auth\RegisterController;
use App\Http\Controllers\Auth\LoginController;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\CatalogController;
use App\Http\Controllers\CartController;
use App\Http\Controllers\OrderController;

use App\Http\Controllers\AdminController;

Route::prefix('lk')->group(function () {
    // Регистрация и авторизация для гостей
    Route::get('/register', [RegisterController::class, 'showRegistrationForm'])->name('register')->middleware('guest');
    Route::post('/register', [RegisterController::class, 'register'])->middleware('guest');
    Route::get('/login', [LoginController::class, 'showLoginForm'])->name('login')->middleware('guest');
    Route::post('/login', [LoginController::class, 'login'])->middleware('guest');
    // Доступ к профилю только для авторизованных пользователей
    Route::get('/profile', [ProfileController::class, 'show'])->name('profile/profile.show')->middleware('auth');
    Route::post('/profile/update', [ProfileController::class, 'update'])->name('profile.update')->middleware('auth');
});
// Выход только для авторизованных пользователей
Route::post('/logout', [LoginController::class, 'logout'])->name('logout')->middleware('auth');

//каталог продукции
Route::get('/catalog', [CatalogController::class, 'index'])->name('shop.catalog');
```

Рисунок 32 – Файл маршрута

6 Документирование программного продукта

6.1 Руководство пользователя веб-приложения

Для того, чтобы открыть программный продукт, необходимо в терминале выполнить команду php artisan serve. Запустится локальный сервер, и программный продукт будет доступен по адресу: http://127.0.0.1:8001. После перехода по ссылке осуществляется переход на главную странницу веб-приложения.

На рисунке 33 представлена главная страница веб-приложения, где можно перейти в каталог продукции и пройти регистрацию/авторизацию.

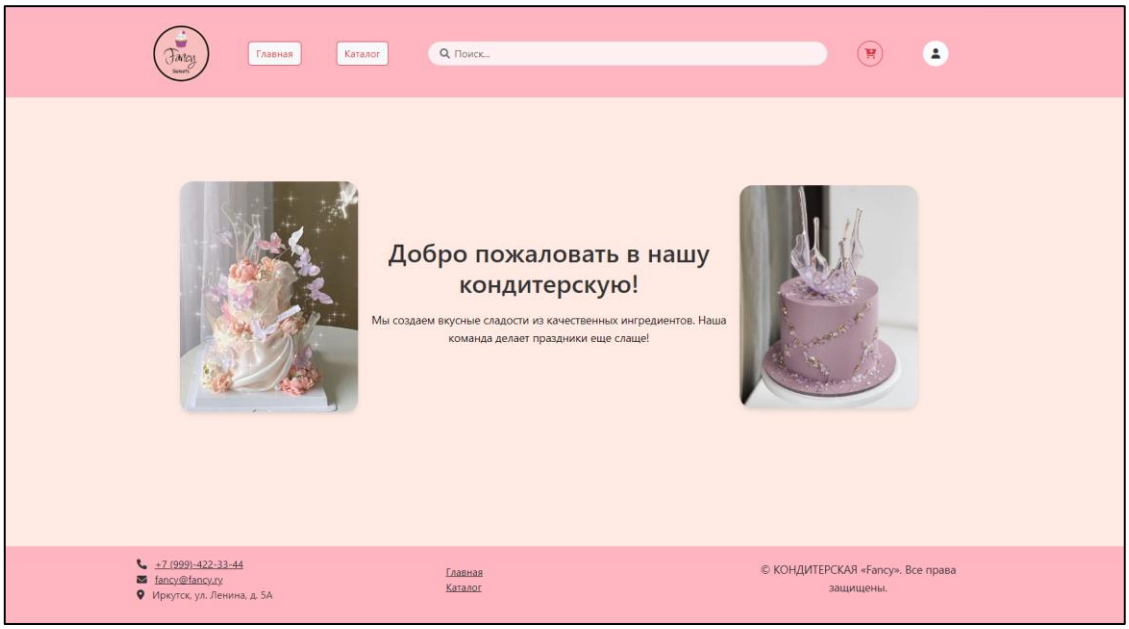


Рисунок 33 – Главная страница

На рисунке 34 представлена страница каталога с карточками продукции, которые можно добавить в корзину для оформления заказа и перейти по кнопке подробнее на страницу с подробными характеристиками продукции.

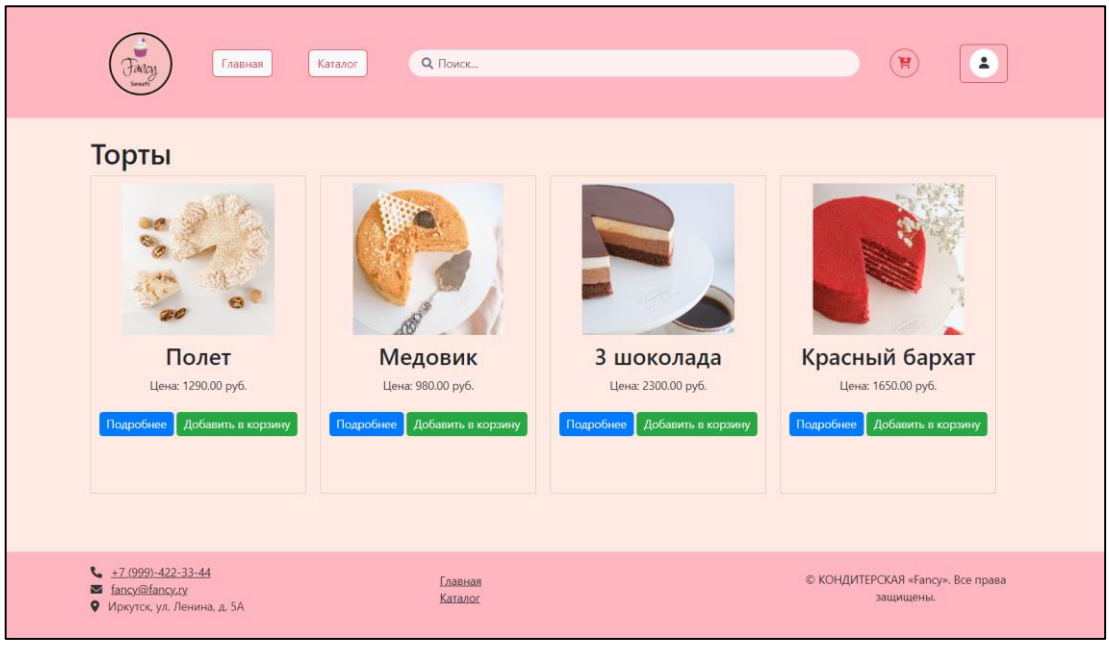


Рисунок 34 – Страница каталога продукции

Для возможности оформления заказа пользователь должен быть авторизован в веб-приложении. Если пользователь не авторизован, то он переходит на страницу авторизации, если он не авторизовался в веб-приложении ранее, то может перейти на страницу регистрации и пройти ее. На рисунках с 35 по 36 представлены формы регистрации и авторизации.

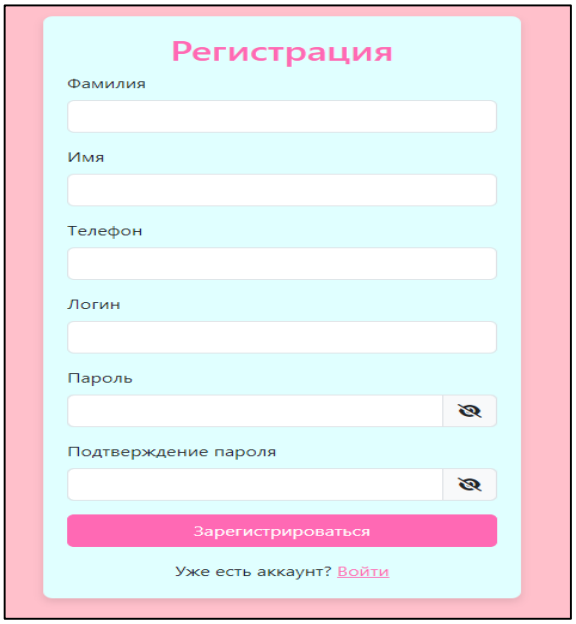


Рисунок 35 – Форма регистрации

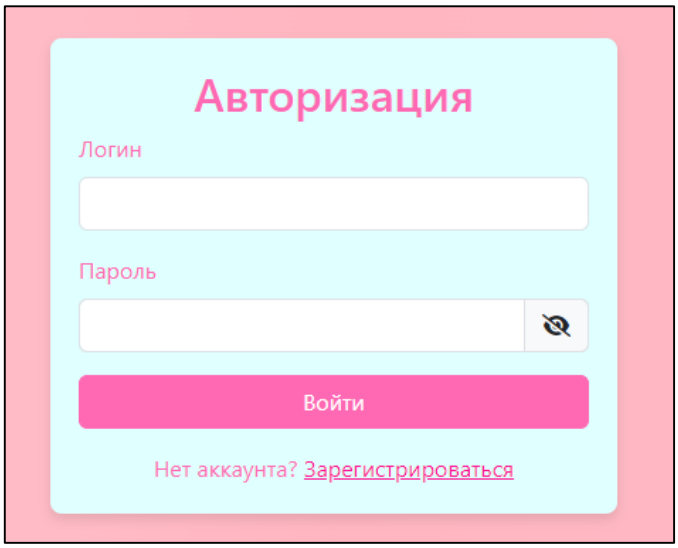


Рисунок 36 – Форма авторизации

После того как пользователь зашел в систему, то осуществляется переход на вкладку корзины, и он повторно нажимает на кнопку оформления заказа и при успешном оформлении появляется страница об успешном оформлении заказа (рисунок с 37 по 38).

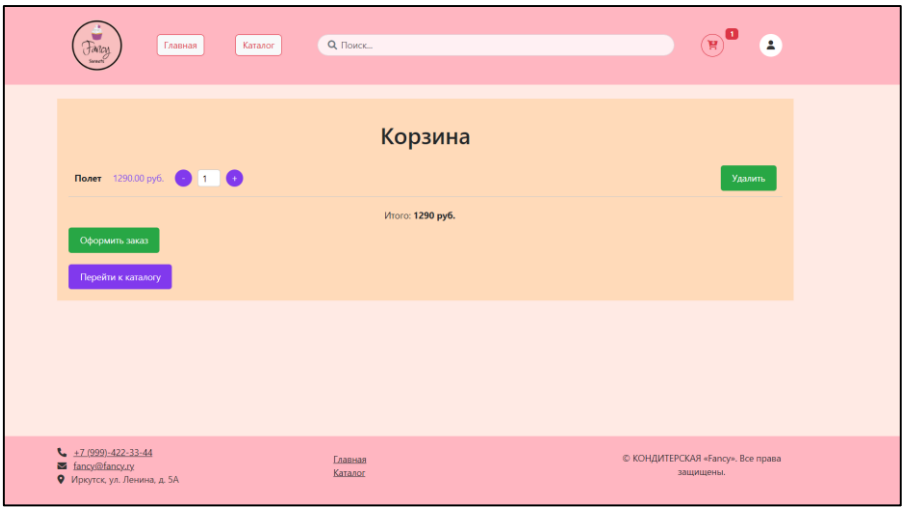


Рисунок 37 – Страница корзины с кнопкой оформления заказа

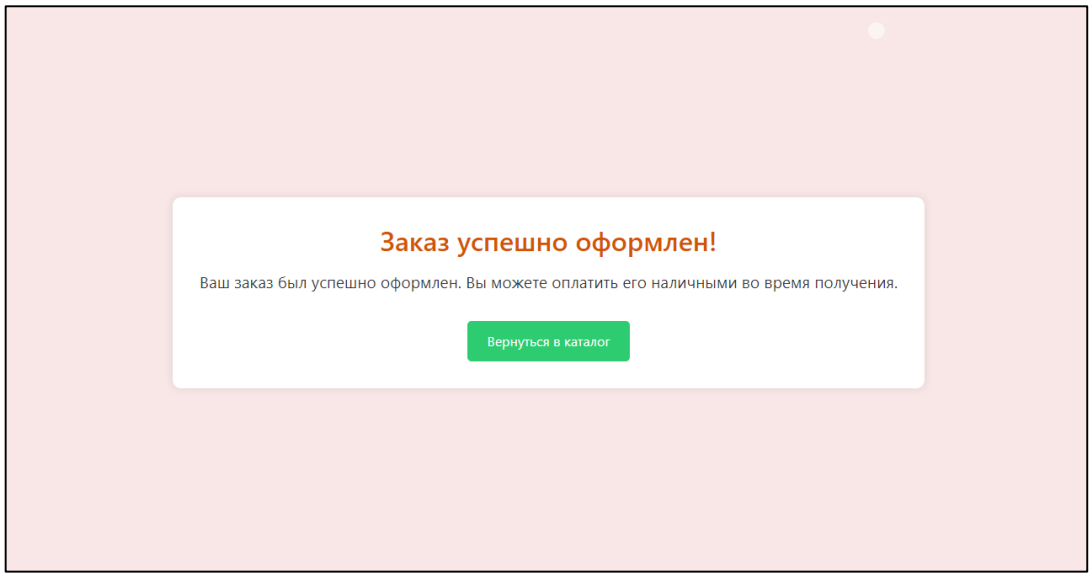


Рисунок 38 – Страница об успешном оформлении заказа

После успешного оформления заказа пользователь может посмотреть историю заказов в личном кабинете и следить за статусом заказа (рисунок 39).

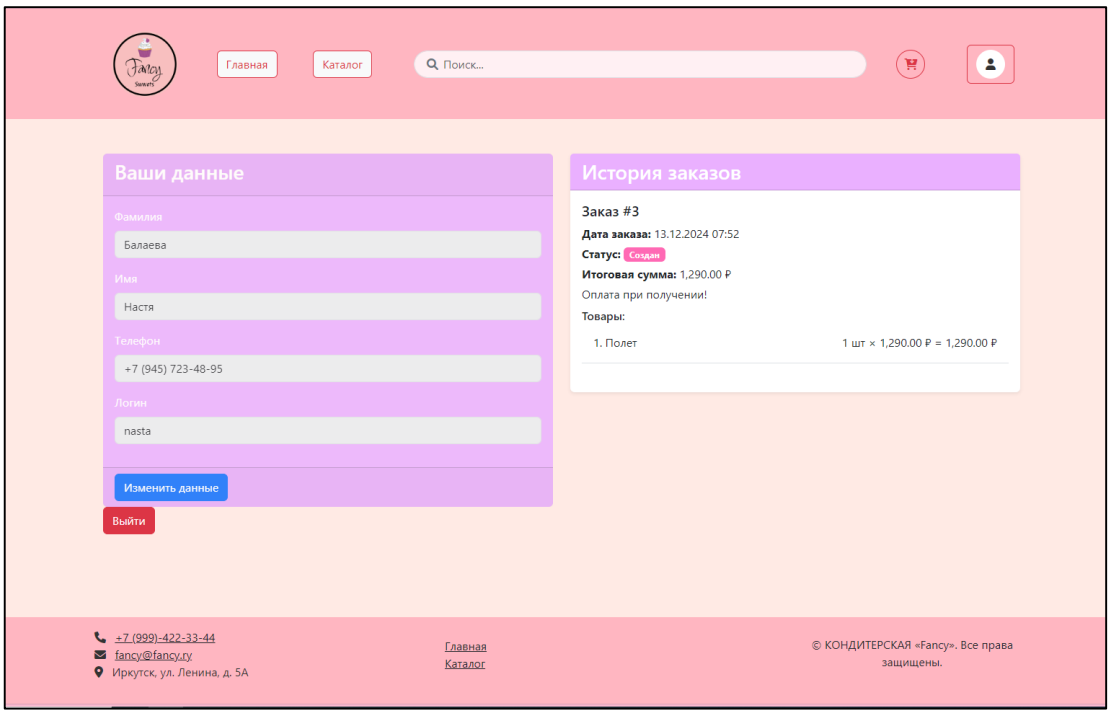


Рисунок 39 – Страница личного кабинета

Заключение

В ходе выполнения курсовой работы было разработано веб-приложение – «Кондитерская». В рамках работы успешно были решены все поставленные задачи:

1. Разработка дизайна.
2. Разработка базы данных.
3. Разработка алгоритма веб-приложения.
4. Разработка прототипов веб-приложения.
5. Разработка программного продукта веб-приложения.
6. Разработка каталога продукции.
7. Написание руководства пользователя.

Был определен и реализован следующий функционал программного продукта:

1. Просмотр каталога продукции и переход на страницу с подробными характеристиками.
2. Регистрация и авторизация пользователей в веб-приложении.
3. Поиск по каталогу продукции.
4. Личный кабинет пользователя.
5. Панель администратора с возможностями: просмотр данных пользователей, добавление, удаление и редактирование каталога продукции и категорий, просмотр и изменение статуса заказа пользователей.

Был разработан программный продукт веб-приложение с интуитивно понятным пользовательским интерфейсом.

Также в процессе работы был изучен фреймворк Laravel, который стал основой для разработки приложения.

В дальнейшем веб-приложение «Кондитерская» может быть расширено функционалом, таким как система отзывов и рейтингов.

Список используемых источников

1 Laravel: – Документация – URL: https://laravel.com/docs/11.x– (дата обращения 20.11.24) – Текст: электронный.

2 skillbox: Что такое MySql. – URL: https://www.geeksforgeeks.org/. (дата обращения 30.09.24) – Текст: электронный.

3 select-dev: Язык программирования PHP. URL: https://select-dev.ru/php-chto-eto/. (дата обращения 30.09.24). Текст – электронный.

4 Bootstrap: Документация. – URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/. (дата обращения 28.11.24). Текст: электронный.

5 Microsoft Learn Challenge: Что такое VS CODE. – URL: https://learn.microsoft.com/ru-ru/shows/visual-studio-code/. (дата обращения 30.09.24). Текст: электронный.

6 gitverse: Что такое UML. – URL: https://gitverse.ru/blog/articles/development/272-chto-takoe-yazyk-modelirovaniya-uml-i-zachem-on-nuzhen. (дата обращения 24.10.24). Текст: электронный.

7 mchost: Что такое phpMyAdmin. – URL: https://mchost.ru/articles/chto-takoe-phpmyadmin/ (дата обращения 01.10.24). Текст: электронный.

8 hsbi: CASE-средства проектирования баз данных – URL: https://hsbi.hse.ru/articles/case-sredstva-proektirovaniya-baz-dannykh/?ysclid=m4qmonj7cg944358862 (дата обращения 05.10.24). Текст: электронный.

9 web-creator: Язык программирования JavaScript. – URL: https://web-creator.ru/technologies/javascript (дата обращения 05.10.24). Текст: электронный.

10 skyeng: Что такое PostgreSQL. – URL: https://skyeng.ru/magazine/chto-takoe-postgresql/ (дата обращения 30.09.24). Текст: электронный.

Приложение А – Техническое задание

Министерство образования Иркутской области

Государственное бюджетное профессиональное

Образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

Техническое задание

ВЕБ-ПРИЛОЖЕНИЕ «КОНДИТЕРСКАЯ»

Руководитель: _____ (М.А. Кудрявцева)

(подпись, дата)

Студент: _____ (Е.М Кривогорницына)

(подпись, дата)

Иркутск 2024

1 Введение

1.1 Общие сведения

Документ представляет собой техническое задание на создание веб-приложения «Кондитерская».

1.2 Цели и задачи

Целью создания веб-приложения «Кондитерская» является автоматизация процесса работы кондитерской.

Задачи веб-приложения включают:

- Реализация каталога кондитерской продукции.
- Регистрация и авторизация пользователей.
- Добавление и удаление продукции в корзине.
- Реализация личного кабинета пользователя с возможность изменения личных данных и просмотра истории заказов.
- Реализация панели администратора, в которой администратор может: просматривать данные пользователей, просматривать и обновлять статус заказов и управлять каталогом продукции и категории (добавление, редактирование, удаление).
- Реализация поиска по каталогу продукции.
- Реализация оформления заказов.

2 Основания для разработки

2.1 Нормативные документы

Документ основывается на следующих нормативных документах:

- ГОСТ 34.602-2020 "Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы".

2.2 Проектные документы

Проектные документы включают:

- Пояснительную записку.

3 Назначение системы

3.1 Общее описание

Веб-приложение «Кондитерская» предназначена для автоматизации процесса взаимодействия клиента с каталогом продукции и оформление заказа.

Веб-приложение предоставляет возможность:

Пользователю:

- Пройти регистрацию и авторизацию.
- Ознакомиться с характеристиками кондитерской продукции.
- Просмотр корзины.
- Осуществлять поиск по каталогу.
- Оформить заказ.
- Просмотреть и изменить персональные данные в личном кабинете.
- Просмотреть историю заказов.

Администратор:

- Управление каталогом и категориями кондитерской продукции: Изменение характеристик кондитерского изделия, добавление и удаление.
- Просматривать информацию о пользователях: личные данные и историю заказов.
- Изменение и просмотр статуса заказа пользователей.

3.2 Преимущества и новизна

Автоматизация веб-приложение «Кондитерская» будет предоставлять:

- Интуитивно понятный интерфейс для пользователей.
- Возможность оформления заказов в любое время суток.

4 Требования к системе

4.1 Функциональные требования

– Каталог кондитерской продукции:

Отображение списка доступных кондитерских изделий с характеристиками.

– Корзина:

Добавления и удаления продукции.

Изменение количества продукции.

Переход к каталогу.

Оформление заказа.

– Оформление заказа:

Проверка на авторизованного пользователя.

– Регистрация и авторизация пользователя:

Регистрация нового пользователя в веб-приложении

Авторизация пользователя в веб-приложении.

– Личный кабинет пользователя:

Просмотр и изменение личных данных (кроме логина).

Просмотр истории заказа и отслеживание статуса заказа.

Выход из профиля.

– Панель администратора:

Просмотр данных пользователя.

Добавление, изменение и удаление продукции и категории.

Просмотр данных заказа и изменение статуса заказа.

4.2 Технические требования

– Производительность:

Обработка до 3 задач одновременно.

Время отклика системы не более 5 секунд при загрузке данных.

– Надежность:

Доступность системы не менее 99,5% в год.

Резервное копирование данных не реже чем 1 раза в месяц.

– Безопасность:

Аунтефикация пользователей и администратора через логин и пароль.

4.3 Эксплуатационные требования

– Удобство использования:

Интуитивно понятный пользовательский интерфейс.

Простота навигации в веб-приложении.

5 Требования к техническому обеспечению

5.1 Оборудование

- Сервер: Серверная платформа с процессором не менее 4 ядер, 16 ГБ ОЗУ, SSD объемом 256 ГБ.
- Клиентские рабочие станции: ПК с ОС Windows 10, 4 ГБ ОЗУ, 2 ГБ свободного места на диске.

5.2 Сетевые требования

- Сеть: Доступ в Интернет со скоростью не ниже 10 Мбит/с.
- Сетевые протоколы: Поддержка TCP/IP, HTTP/HTTPS.

6 Требования к программному обеспечению

6.1 Программные компоненты

– Операционная система: Серверная версия Windows Server.

– Базы данных: MySQL версии не ниже 10.0.

– Программное обеспечение: веб-сервер Apache версия не ниже 2.4, Visual Studio Code.

6.2 Интерфейсы

– Интерфейс пользователя: Веб-интерфейс с поддержкой браузера Chrome версия 131.

7 Организационно-технические требования

7.1. Этапы разработки

В таблице 9 представлены сроки и этапы разработки системы.

Таблица 9 – Сроки и этапы разработки системы

№	Этап	Срок выполнения
1	Предпроектное исследование предметной области (выбор темы, постановка цели, задач, описание области применения, исследование предметной области)	До 18.09.24
2	Разработка технического задания (выбор архитектуры программного обеспечения, выбор типа пользовательского интерфейса, выбор языка и среды программирования)	До 23.09.24
3	Проектирование программного обеспечения. (разработка структурной и функциональной схемы ПО, проектирование базы данных (инфологическое, ER-модель, физическая модель)	До 25.10.24
4	Разработка (программирование) и отладка программного продукта	До 14.12.24
5	Составление программной документации (оформление ПЗ, написание руководства пользователя, составление презентации и речи)	До 16.12.24

Приложение Б – Листинг RegisterController

```
<?php
namespace App\Http\Controllers\Auth;
use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;
use Illuminate\Validation\Rule;
class RegisterController extends Controller
{
public function showRegistrationForm()
{
return view('auth.register');
}

public function register(Request $request)
{
$request->validate([
'last_name' => 'required|string|max:100',
'first_name' => 'required|string|max:100',
'phone' => 'required|string|max:20|unique:users,phone',
'login' => 'required|string|max:20|unique:users,login',
'password' => 'required|string|min:8|confirmed',
], [
'phone.unique' => 'Номер телефона уже зарегистрирован',
'login.unique' => 'Логин занят',
'password.confirmed' => 'Введённые пароли не совпадают',
'password.min' => 'Длина пароля не может быть меньше 8 символов',
]);
User::create([
'last_name' => $request->last_name,
'first_name' => $request->first_name,
'middle_name' => $request->middle_name,
'phone' => $request->phone,
'login' => $request->login,
'password' => Hash::make($request->password),
]);
return redirect()->route('login')->with('status', 'Регистрация успешна!');
}}
```