Федеральное государственное образовательное бюджетное учреждение высшего образования

**«Финансовый университет при Правительстве**

**Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

(наименование структурного подразделения)

Дипломный проект

Тема «Разработка веб-сайта кондитерских изделий»

Студент Емельянова Дарья Игоревна

Учебная группа 4ИСИП-621 1

Специальность 09.02.07 Информационные системы и программирование

Студент Прицепа Илана Александровна

Учебная группа 4ИСИП-621 1

Специальность 09.02.07 Информационные системы и программирование

Руководитель

дипломного проекта 1 1 А. Ю. Чернышев 1

(подпись) (инициалы, фамилия)

Консультант

дипломного проекта

(при наличии) 1 1 1 1

(подпись) (инициалы, фамилия)

Председатель предметной

(цикловой) комиссии 1 1 1 1

(подпись) (инициалы, фамилия)

Москва – 2025 г.

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc200537520)

[**Глава 1. Предпроектное исследование предметной области** 6](#_Toc200537521)

[**1.1. Обоснование выбора и анализ предметной области** 6](#_Toc200537522)

[**1.2. Сравнительный анализ существующих приложений** 7](#_Toc200537523)

[**1.3. Постановка требований к приложению** 9](#_Toc200537524)

[**1.5. Обоснование выбора инструментальных средств разработки** 14](#_Toc200537525)

[**1.6. Характеристика инструментальных средств разработки** 16](#_Toc200537526)

[**1.7. Информационно-логическая модель веб-приложения** 18](#_Toc200537527)

[**Глава 2. Разработка приложения** 22](#_Toc200537528)

[**2.1. Проектирование и реализация базы данных приложения** 22](#_Toc200537529)

[**2.2. Разработка дизайна графического пользовательского интерфейса.** 24](#_Toc200537530)

[**2.3. Проектирование пользовательского интерфейса.** 26](#_Toc200537531)

[**2.3 Разработка серверной части приложения** 33](#_Toc200537532)

[**2.4. Отладка и тестирование веб-приложения** 35](#_Toc200537533)

[**2.6. Публикация приложения на хостинге** 37](#_Toc200537534)

[**2.7. Разработка руководства по использованию приложения** 38](#_Toc200537535)

[**ЗАКЛЮЧЕНИЕ** 40](#_Toc200537536)

[**СПИСОК ЛИТЕРАТУРЫ (ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ) И ИНТЕРНЕТ-РЕСУРСОВ** 42](#_Toc200537537)

[**ПРИЛОЖЕНИЕ** 43](#_Toc200537538)

# **ВВЕДЕНИЕ**

Современные информационные технологии оказывают значительное влияние на все сферы бизнеса, в том числе на рынок кондитерских изделий. В условиях высокой конкуренции и постоянно изменяющихся предпочтений потребителей, компании вынуждены искать новые пути для продвижения своей продукции и улучшения взаимодействия с клиентами. Одним из наиболее эффективных способов достижения этих целей является создание веб-сайта, который позволяет не только презентовать ассортимент продукции, но и оптимизировать процессы продаж, а также повысить уровень доверия и лояльности со стороны покупателей.

Веб-сайт для индивидуального предпринимателя, занимающейся производством кондитерских изделий, предоставляет уникальные возможности для расширения рынка сбыта, продвижения бренда и обеспечения более тесного контакта с целевой аудиторией. Удобный и функциональный сайт способен не только предоставить всю необходимую информацию о товарах, но и предоставить пользователям возможность удобно и быстро оформлять заказы, участвовать в акциях и получать информацию о новинках.

Актуальность темы дипломного проекта обусловлена растущим интересом потребителей к онлайн-покупкам и потребностью компаний в эффективных цифровых платформах для продвижения своих товаров. Сайты, предоставляющие информацию о продукции, возможности онлайн-заказов и оплаты, становятся неотъемлемой частью конкурентоспособности на рынке. Важно создать платформу, которая будет не только эстетически привлекательной, но и удобной для пользователей, обеспечивая комфортный процесс выбора и покупки кондитерских изделий.

Объектом исследования является веб-сайт для частного бизнеса кондитерских изделий.

Предмет исследования – разработка веб-сайта кондитерских изделий “Sweetlana”.

Цель – разработка функционального веб-сайта для частной кондитерской, который будет удовлетворять требованиям пользователей и бизнеса.

Задачи дипломного проекта:

- Анализ потребностей пользователей и определение функциональных требований к сайту.

- Разработка дизайна и структуры сайта с учетом специфики рынка кондитерских изделий.

- Реализация функционала для оформления заказов, отображения продукции и взаимодействия с клиентами.

- Проведение тестирования сайта и его оптимизация для обеспечения высокой производительности и безопасности.

- Разработка рекомендаций для дальнейшего улучшения сайта и его продвижения.

Разграничение ответственности:

Емельянова Дарья Игоревна — ответственна за разработку серверной части.

В мои задачи входит:

* 1. Проектировка и реализация REST API;
  2. Разработка модели данных и взаимодействие с базой данных;
  3. Настройка бизнес-логики приложения;
  4. Обработка ошибок и валидация данных;
  5. Реализация аутентификации и авторизации;
  6. Тестирование и отладка backend-части;
  7. Настройка инфраструктуры запуска;
  8. Размещение приложения и базы данных на хостинге.

Прицепа Илана Александровна — ответственен за разработку клиентской части.

В мои задачи входит:

1. Проектирование архитектуры frontend-приложения;
2. Разработка дизайна приложения с учётом основных принципов UX/UI;
3. Разработка пользовательского интерфейса веб-сайта на основе утверждённого дизайна;
4. Интеграция с REST API, настройка взаимодействия с backend частью;
5. Реализация клиентской логики веб-сайта;
6. Тестирование и оптимизация производительности клиентской части;
7. Сборка и деплой веб-приложения на хостинг;

Результатом проекта станет готовый веб-сайт, который будет соответствовать современным стандартам и поможет повысить продажи и укрепить свою позицию на рынке кондитерских изделий.

# **Глава 1. Предпроектное исследование предметной области**

## **1.1. Обоснование выбора и анализ предметной области**

Разработка веб-сайта для индивидуального предпринимателя, занимающейся производством и продажей кондитерских изделий, является актуальной задачей в условиях современной цифровой экономики. С увеличением популярности онлайн-торговли и стремлением бизнеса выходить на новые каналы сбыта, создание профессионального веб-сайта становится важным элементом успешного функционирования предприятия. Веб-сайт кондитерской компании предоставляет уникальные возможности для привлечения клиентов, повышения узнаваемости бренда и увеличения продаж.

Кондитерская отрасль является одной из наиболее востребованных в сфере потребительских товаров. В условиях роста конкуренции на рынке кондитерских изделий, веб-сайт играет важную роль не только как средство продвижения, но и как полноценный канал для прямых продаж, позволяя значительно расширить клиентскую базу и улучшить коммуникацию с покупателями. Веб-сайт становится важным инструментом для демонстрации продукции, ее характеристик, акций и новинок, а также для получения обратной связи от клиентов, что способствует улучшению качества обслуживания и повышению лояльности потребителей.

Анализ предметной области показывает, что большинство кондитерских компаний, несмотря на конкурентные преимущества, не полностью используют потенциал интернета для расширения своего рынка. Многие из них имеют устаревшие сайты или вовсе их отсутствие, что негативно влияет на их конкурентоспособность. В связи с этим, создание качественного веб-сайта становится важной задачей, которая позволяет кондитерским компаниям улучшить маркетинг, повысить узнаваемость бренда и обеспечить удобство покупки для клиентов.

Для разработки веб-сайта в данном проекте будет учитываться потребность в удобной навигации, понятном интерфейсе, обеспечении функционала онлайн-магазина, а также доступности информации о продукции и возможностях доставки. Важно также учитывать возможности для дальнейшего расширения функционала сайта, такие как интеграция с социальными сетями, поддержка пользовательских отзывов и рейтингов, а также реализация маркетинговых инструментов, таких как акции, скидки и персонализированные предложения.

Таким образом, создание веб-сайта для кондитерской компании отвечает актуальным требованиям современного бизнеса и позволит значительно улучшить взаимодействие с клиентами, повысить эффективность продаж и укрепить позиции на рынке.

## **1.2. Сравнительный анализ существующих приложений**

Разработка веб-сайта кондитерских изделий требует учёта актуальных тенденций и пользовательских ожиданий, сформированных уже существующими цифровыми решениями в данной нише. В этой связи особую ценность представляет сравнительный анализ конкурентных сайтов, осуществляющих онлайн-продажу тортов и десертов. Целью анализа является выявление сильных и слабых сторон аналогичных проектов с последующим применением лучших практик при создании сайта Sweetlana.

В качестве объектов анализа были выбраны сайты «Доктор Бейкер» и Cake Buro, представляющие различные подходы к визуальной подаче, пользовательскому опыту и функциональности. Полученные результаты сопоставлены с проектируемым решением Sweetlana, которое нацелено на создание современного, визуально привлекательного и технологически продвинутого продукта.

Анализ проводился по следующим ключевым критериям: ассортимент сайта, интерфейс (UX) и пользовательский опыт (UI), функционал и наличие личного кабинета, удобность оформления заказа, уровень адаптации интерфейса под мобильные и планшетные устройства, SEO и производительность, а также ключевые особенности, формирующие конкурентное преимущество.

Результаты сравнительного анализа представлены в таблице 1.

Таблица 1 – Сравнительный анализ конкурентов

|  |  |  |  |
| --- | --- | --- | --- |
| Критерий | Доктор Бейкер | Cake Buro | Sweetlana |
| Ассортимент | Торты, капкейки, тематические наборы (детские, свадебные и др.) | Дизайнерские, элитные торты | Трендовые и традиционные дессерты, нестандартные дизайны тортов |
| UX и UI | Семейный, яркий, устарелый и неудобная навигация | Современный и минималистичный дизайн с интуитивным и понятным интерфейсом | Чистый и аппетитный цветовой стиль, продуманная архитектура |
| Личный кабинет | Присутствует, но нельзя изменять данные и настройки аккаунта | Полностью отсутствует | Присутствует с возможностью просмотра статуса и истории заказов, а также изменение настроек профиля |
| Оформление заказа | Полностью реализовано, включая выбор даты, веса, оформления и надписей | Есть форма заказа, но требуется звонок от менеджера для уточнения заказа | Полностью реализовано |
| Мобильная адаптация | Средняя, есть проблемы с масштабированием некоторых элементов | Отлично адаптирована, быстро загружается, хорошо просматривается | Удобная, современная и визуально понятная |
| Скорость загрузки сайта | Медленная, задержки | Высокая, оптимизированные изображения | Высокая, гибкое приложение |
| SEO/поисковая оптимизация | Сайт плохо индексируется, медленные страницы | Хорошая структура URL, описание товаров |  |
| Особенности | Упор на праздник, детские заказы, креативность | Упор на эстетику, премиальность и мастерство своего дела | Конструктор тортов |

Сравнительный анализ показал, что существующие решения на рынке кондитерских веб-сайтов обладают как сильными сторонами, так и существенными недостатками. CakeBuro выделяется визуальной эстетикой, высокой скоростью загрузки и качественной мобильной адаптацией, но ограничивает пользователя в функциональности и не предоставляет полноценного личного кабинета. В свою очередь, «Доктор Бейкер» предлагает широкий ассортимент и наличие личного кабинета, однако страдает от устаревшего интерфейса, слабой адаптивности и медленной работы сайта.

На этом фоне проект Sweetlana формируется как сбалансированное решение, объединяющее лучшие практики конкурентов: современный дизайн, продуманную структуру, мобильную оптимизацию, а также расширенный функционал личного кабинета и конструктора тортов. Такой подход позволяет создать конкурентоспособный продукт, ориентированный как на визуальное восприятие, так и на удобство пользователя.

## **1.3. Постановка требований к приложению**

Для разработки веб-сайта кондитерских изделий были сформулированы функциональные и нефункциональные требования, разделённые по архитектурным уровням приложения: клиентская часть (frontend) и серверная часть (backend). Такой подход позволяет системно подойти к проектированию и обеспечить устойчивую работу как пользовательского интерфейса, так и внутренней логики обработки данных.

**Клиентская часть**

Клиентская часть представляет собой пользовательский интерфейс веб-приложения и отвечает за визуализацию данных, отображение контента, обработку действий пользователя и предоставление удобных средств навигации. Её основная задача — обеспечить интуитивно понятное, стабильное и адаптивное взаимодействие с системой на всех типах устройств (десктоп, планшет, смартфон).

**Функциональные требования**

1) Информационный раздел

* отображение информации о частном бизнесе (о нас, контакты, политика конфиденциальности);
* витрина продукции с возможностью фильтрации по типам, категориям, цене и поиском по названию;
* карточки товаров с изображениями, составом, ценой и возможностью настройки (начинка, надпись, упаковка).

2) Работа с пользователями

* регистрация и авторизация;
* личный кабинет: история и статусы заказов, редактирование профиля.

3) Оформление заказа

* корзина (добавление, удаление, изменение количества товаров);
* пошаговая форма заказа с выбором даты, времени и способа оплаты, доставки;
* отображение статуса заказа после оформления.

4) Обратная связь

* связь через мессенджеры (Telegram, WhatsApp).

5) Интерактивные элементы

* конструктор тортов;
* поисковая строка и адаптивное навигационное меню.

**Нефункциональные требования**

1) Адаптивность

* корректное отображение интерфейса на всех типах устройств (ПК, планшеты, смартфоны).

2) Удобство использования интерфейса:

* чёткая логика взаимодействия и понятная структура страниц;
* минимальное количество действий до оформления заказа.

3) Скорость отклика

* загрузка страниц и ключевых компонентов не более 3 секунд при стандартном соединении.

4) Доступность и развёртывание

* интерфейс должен быть интуитивно понятен пользователям;
* публикация клиентской части на платформе Vercel.

5) Совместимость

* поддержка актуальных браузеров (Chrome, Firefox, Safari, Edge).

**Серверная часть**

Серверная часть отвечает за внутреннюю логику работы сайта, обработку заказов, хранение данных и безопасность. В условиях индивидуального управления важно обеспечить стабильную и безопасную работу без необходимости постоянного технического обслуживания.

**Функциональные требования**

1) Работа с пользователями

* хранение персональных данных;
* обработка авторизации и регистрации;
* проверка прав доступа и безопасности операций.

2) Управление товарами

* CRUD-операции для каталога продукции (создание, редактирование, удаление, отображение);
* хранение информации о начинках, ценах, доступности товаров.

3) Обработка заказов

* приём и сохранение заказов в базе данных;
* расчёт общей стоимости с учётом параметров;
* назначение и отслеживание статуса.

4) Личный кабинет и история заказов

* истории заказов пользователя;
* доступ к изменениям профиля и настройкам.

5) Платёжная система

* интеграция с внешними API платёжных;
* проверка статуса оплаты и отображение его клиенту.

**Нефункциональные требования**

1) Безопасность

* хеширование паролей;
* защита от SQL-инъекций.

2) Надёжность

* стабильная работа при высоких нагрузках;
* защита от сбоев.

3) Производительность

* быстрая обработка пользовательских запросов;
* минимальная задержка ответа API.

4) Масштабируемость

* возможность добавления новых модулей без полной переработки архитектуры.

5) Обслуживаемость

* чёткая структура кода;
* наличие логирования, документация по внутренним API.

6) Резервное копирование

* регулярное сохранение базы данных и настроек приложения.

**1.4. Описание архитектуры приложения**

Архитектура веб-сайта «Sweetlana» по продаже кондитерских изделий разработана с использованием современных технологий, что обеспечивает высокую производительность, безопасность и удобство поддержки кодовой базы.

Проект реализован как Full-stack приложение, состоящее из следующих уровней:

* клиентский (Frontend) — отвечает за интерфейс пользователя и взаимодействие с ним.
* серверный (Backend) — обрабатывает бизнес-логику, управляет данными и предоставляет API.
* базы данных (Database) — хранение и организация данных о товарах, пользователях, заказах и т. д.

Для реализации пользовательского интерфейса используется фреймворк Next.js 13+ с App Router , поддерживающий такие возможности, как Server Components, SSR (Server Side Rendering) и SSG (Static Site Generation). Это положительно влияет на SEO, повышает производительность и позволяет предварительно загружать данные ещё на стороне сервера. Маршрутизация организована через файловую структуру app/, что обеспечивает гибкий и интуитивно понятный подход к созданию страниц.

Пример структуры клиентской части:

* layout.tsx — общий макет, применяемый ко всем страницам;
* page.tsx — главная страница сайта;
* about/page.tsx — страница "О нас" с маршрутом /about;
* about/styles/style.scss — таблицы стилей для раздела "О нас";
* components/ — каталог переиспользуемых UI-компонентов;
* api/ — утилиты и хелперы для работы с API;
* public/ — статические файлы, такие как изображения и шрифты.

Серверная логика реализована с использованием фреймворка NestJS — мощного и модульного инструмента для разработки Node.js-приложений. Архитектура основана на принципах ООП и функционального программирования, использует такие паттерны проектирования, как Dependency Injection, Middleware, Guards и Interceptors. Это делает приложение легко тестируемым, расширяемым и поддерживаемым.

Backend предоставляет REST API, через который клиент получает и отправляет данные. Основные файлы и директории серверной части:

* main.ts — точка входа приложения;
* app.controller.ts — главный контроллер;
* app.service.ts — логика обработки данных;
* app.module.ts — основной модуль приложения;
* auth/ — модуль авторизации;
* products/ — модуль товаров;
* users/ — модуль пользователей;
* orders/ — модуль заказов;
* fillings/ — модуль начинок;
* database/ — настройки TypeORM и модели;
* config/ — конфигурационные файлы (CORS, сессии).

В качестве системы хранения данных выбрана реляционная СУБД MySQL, которая обеспечивает надёжное управление такими сущностями, как товары, пользователи, заказы и категории. Для взаимодействия с базой данных в NestJS используется ORM TypeORM, позволяющая работать с данными через TypeScript-модели.

В рамках реализации веб-сайта «Sweetlana» были созданы следующие таблицы:

* products — хранение информации о кондитерских изделиях.
* users — данные зарегистрированных пользователей и их роли
* orders — информация о заказах, оформленных пользователями.
* order\_items — детализация состава заказов.

## **1.5. Обоснование выбора инструментальных средств разработки**

Выбор технологий и инструментов разработки является важным этапом при создании веб-приложений, поскольку от него зависят такие ключевые характеристики проекта, как производительность, масштабируемость, удобство сопровождения, безопасность и скорость разработки. В рамках дипломного проекта «Sweetlana» были тщательно подобраны современные и устойчивые инструменты для реализации клиентской и серверной частей, а также базы данных.

Для реализации клиентской части был выбран фреймворк Next.js с использованием App Router, который предоставляет мощные возможности для создания динамических и статических страниц. Использование Server Side Rendering (SSR) и Static Site Generation (SSG) позволило улучшить SEO и повысить скорость загрузки сайта, что особенно важно для коммерческих интернет-магазинов. App Router обеспечивает гибкую структуру маршрутов, ленивую загрузку страниц и простоту управления состоянием, что делает его актуальным выбором для современных веб-приложений. В качестве основного JavaScript-библиотеки используется React, которая позволяет строить интерфейс в виде компонентов, что способствует переиспользованию кода и его модульности.

Для стилизации веб-сайта применяется препроцессор Sass, предоставляющий расширенные возможности CSS, такие как переменные, миксины и вложенные правила.

Для управления состоянием формы и валидации вводимых данных используется библиотека react-hook-form, которая оптимизирована для высокой производительности и позволяет легко управлять формами без лишнего рендера компонентов.

Для взаимодействия с сервером применяется библиотека Axios, которая обеспечивает удобную работу с HTTP-запросами, включая обработку ошибок, перехватчики и управление таймаутами.

Серверная часть была реализована с использованием NestJS — мощного прогрессивного фреймворка для построения надежных и легко тестируемых серверных приложений на базе Node.js. NestJS предлагает модульную архитектуру, поддержку TypeScript из коробки, внедрение зависимостей и готовые решения для работы с REST API, аутентификацией и валидацией. Эти особенности обеспечивают высокую читаемость кода, простоту расширения и долгосрочное сопровождение приложения.

Для хранения данных использовалась реляционная система управления базами данных MySQL, которая отличается высокой надёжностью, хорошей документированностью и широким сообществом. А для взаимодействия с базой данных применялась ORM TypeORM, позволяющая работать с данными в объектно-ориентированном стиле, сохраняя при этом преимущества SQL. TypeORM обеспечила простоту миграций, валидацию данных и типизацию запросов, что положительно сказалось на скорости разработки и безопасности приложения.

Для шифрования паролей применяется библиотека bcrypt, гарантирующая сохранность учетных данных пользователей.

Для управления сессиями на сервере используется middleware express-session, которое позволяет хранить данные о текущем пользователе между HTTP-запросами. Библиотека nestjs-session обеспечивает интеграцию с NestJS и удобное управление сессиями в рамках модульной архитектуры.

Для валидации входящих данных используются библиотеки class-validator и class-transformer, которые позволяют декларативно задавать правила валидации DTO (Data Transfer Objects) и преобразовывать входящие данные в соответствующие модели.

Для публикации веб-сайта «Sweetlana» в сети Интернет был выбран облачный хостинг-провайдер Railway. Это современная платформа, предназначенная для быстрого развёртывания и управления приложениями, которая предоставляет широкий набор инструментов для автоматизации процессов разработки, тестирования и публикации проектов.

Данный выбор инструментальных средств разработки был обоснован стремлением к созданию надёжного, масштабируемого и современного веб-приложения, соответствующего требованиям электронной коммерции. Применённые технологии обеспечивают не только функциональность и производительность, но и упрощают дальнейшее развитие и сопровождение проекта.

## **1.6. Характеристика инструментальных средств разработки**

Для реализации и сопровождения веб-приложения использовался комплекс современных инструментальных средств:

Клиентская часть:

* Next.js — фреймворк для React, обеспечивает SSR/SSG, быстрый рендеринг и отличное SEO;
* React — библиотека для создания компонентного интерфейса;
* Sass — препроцессор для удобной и структурированной стилизации;
* react-hook-form — эффективное управление состоянием и валидация форм;
* Axios — работа с HTTP-запросами.

Серверная часть:

* NestJS — прогрессивный фреймворк на TypeScript, модульная архитектура, удобен для масштабирования;
* TypeScript — строгая типизация и повышение надёжности кода;
* bcrypt — безопасное хеширование паролей;
* express-session и nestjs-session — управление сессиями пользователей;
* class-validator и class-transformer — удобная валидация и трансформация данных.

База данных:

* MySQL — надёжная реляционная СУБД;
* TypeORM — работа с БД в объектно-ориентированном стиле, удобные миграции.

Публикация и хостинг:

* Railway — облачный хостинг-провайдер и интеграция с GitHub.

Инструменты для документирования и визуализации:

* Figma — прототипирование и дизайн интерфейсов, работа в команде онлайн;
* Microsoft Word — оформление пояснительной записки, структурирование текста, таблиц и иллюстраций;
* Microsoft PowerPoint — создание презентаций с изображениями, схемами и диаграммами для защиты проекта.

Данный выбор инструментальных средств разработки был обоснован стремлением к созданию надёжного, масштабируемого и современного веб-приложения, соответствующего требованиям электронной коммерции. Применённые технологии обеспечивают не только функциональность и производительность, но и упрощают дальнейшее развитие и сопровождение проекта.

## **1.7. Информационно-логическая модель веб-приложения**

Для правильного построения и визуализации сайта кондитерских изделий нам нужны: функциональная и структурная диаграммы, и диаграмма вариантов использования.

Функциональная диаграмма – это диаграмма, которая иллюстрирует потоки данных в системе или процессе. Она помогает визуализировать, как данные перемещаются от одной функции к другой в рамках системы и как они обрабатываются. Функциональные диаграммы используются в системном анализе и проектировании для описания взаимодействия компонентов системы и определения их связей (см. Рисунок 1).

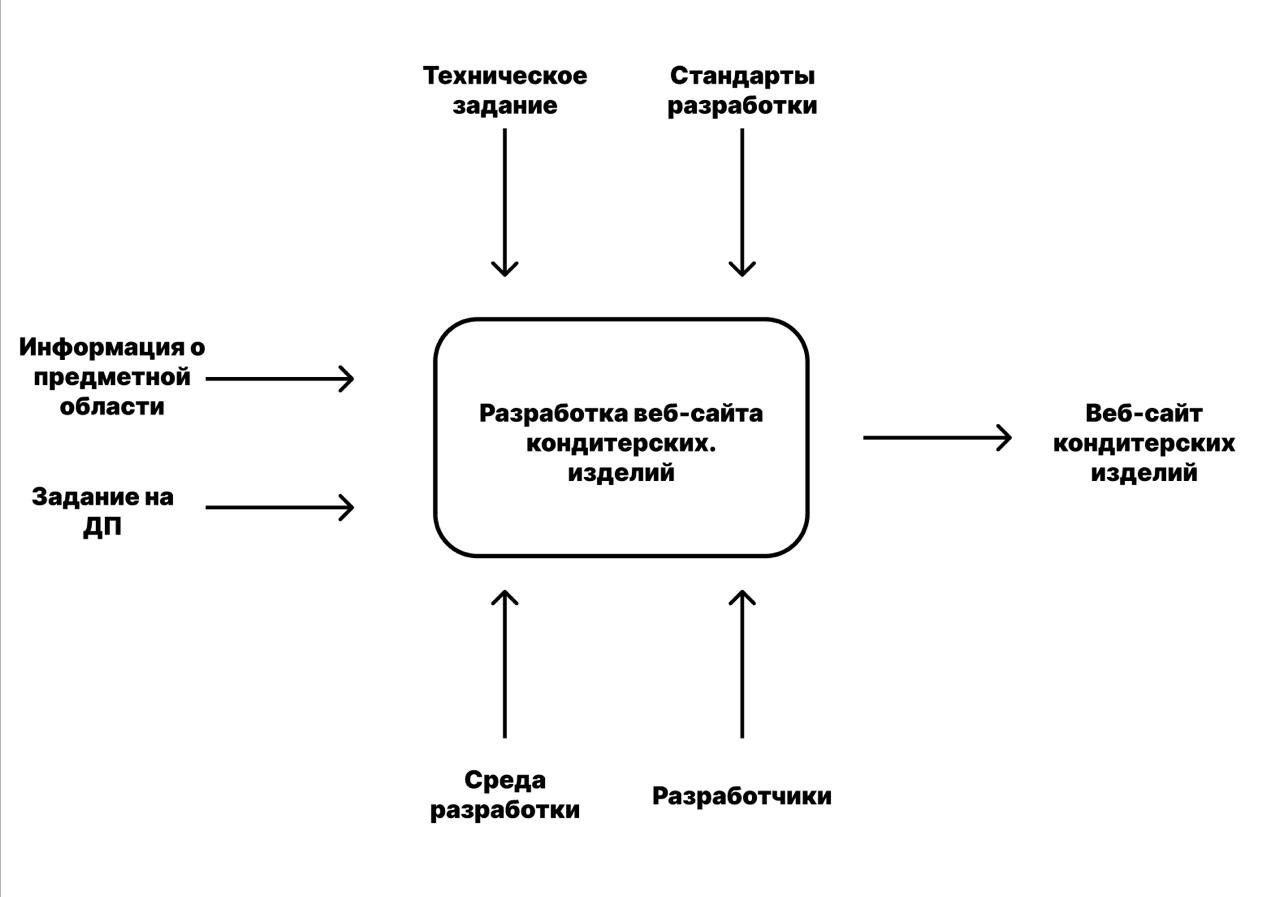


Рисунок 1 - Функциональная диаграмма

Структурная диаграмма является важным инструментом для понимания и визуализации структуры самого приложения, его компонентов и взаимодействия между ними. Она позволяет нам проектировать и организовывать код и функциональность. Такая диаграмма обычно состоит из блоков, представляющих компоненты, и стрелок, обозначающих связи между ними (см. Рисунок 2).



Рисунок 2 - Карта сайта

Диаграмма вариантов использования помогает иллюстрировать, как разные пользователь взаимодействует с веб-сайтом через различные варианты использования. На ней можно отобразить различные сценарии или функциональные возможности (см. Рисунок 3).



Рисунок 3 – Диаграмма вариантов использования

# **Глава 2. Разработка приложения**

## **2.1. Проектирование и реализация базы данных приложения**

В качестве системы управления базами данных (СУБД) веб-сайта кондитерских изделий выбрана MySQL. В этой же программе создана база данных под названием sweetlana, которая будет использоваться для хранения всех данных проекта. Основные задачи и содержание, которой были следующие:

- Информации о пользователях.

- Управление каталогом товаров.

- Данные о начинках для тортов.

- Обработка и изменение статусов заказов пользователей.

- Связь между различными сущностями.

На основе этих требований были созданы и реализованы основные таблицы и их связи: users, products, filling, addons, order\_items и orders (см. Рисунок).

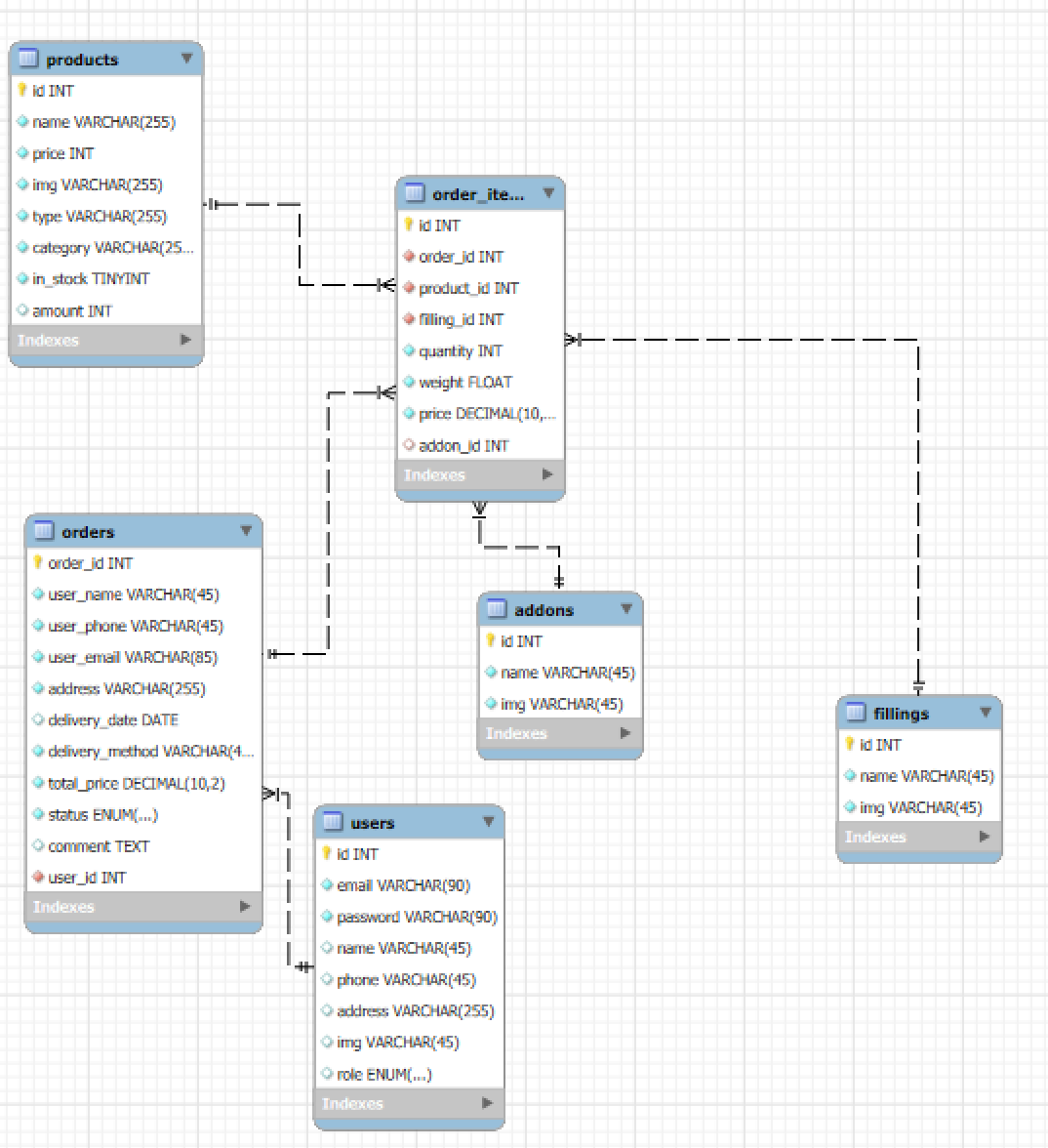


Рисунок 4. Диаграмма сущность связь

Диаграмма «сущность-связь» помогает эффективно спроектировать и организовать данные, что важно для поддержания их целостности и быстродействия системы. Также помогает визуализировать, какие сущности (например, пользователи, продукты, комментарии) существуют в системе и как они связаны между собой

Каждая таблица была спроектирована с учетом типов данных, ключей и ограничений:

- Primary Key: уникальный идентификатор каждой записи.

- Foreign Key: обеспечивает связи между таблицами (user\_id в таблице orders ссылается на id в таблице users).

- Data Types: использовались подходящие типы данных для каждого поля.

Таблица users предназначена для хранения информации о пользователях системы. В ней содержатся такие поля, как электронная почта, пароль, имя, номер телефона, адрес, изображение профиля и роль пользователя. Это позволяет реализовать регистрацию, авторизацию и разграничение прав доступа внутри приложения.

Таблица orders отражает заказы пользователей. Здесь фиксируются данные о заказчике (имя, телефон, электронная почта), адрес доставки, способ доставки, дата заказа, итоговая стоимость, статус заказа, а также ссылка на пользователя через внешний ключ (user\_id). Это обеспечивает возможность хранения истории заказов и аналитики по пользователям.

Таблица order\_items служит связующим звеном между заказами и товарами, реализуя структуру "многие ко многим". Каждый элемент заказа содержит ссылки на заказ (order\_id), товар (product\_id), выбранную начинку (filling\_id) и дополнительную опцию (addon\_id), а также хранит количество, вес и цену позиции. Такое построение позволяет формировать сложные составные заказы с индивидуальными настройками для каждой позиции.

Таблица products содержит перечень товаров, представленных в магазине, с такими атрибутами, как наименование, цена, изображение, тип, категория, количество на складе и признак наличия. Данная структура позволяет гибко управлять ассортиментом и отслеживать остатки.

Таблицы fillings и addons хранят возможные варианты начинок и дополнительных опций для товаров. Каждая начинка и каждая добавка имеют уникальный идентификатор, наименование и изображение. Связи между этими таблицами и таблицей order\_items реализованы посредством внешних ключей, что обеспечивает корректную обработку индивидуальных пожеланий клиента при формировании заказа.

## **2.2. Разработка дизайна графического пользовательского интерфейса.**

Важной частью разработки современного веб-сайта является создание привлекательного и интуитивно понятного графического пользовательского интерфейса (GUI). Для проекта особое внимание уделялось эстетике и эмоциональному восприятию, поскольку оформление сайта напрямую влияет на уровень доверия и лояльности потенциальных клиентов. Основная работа велась в Figma.

На рисунке 5 – дизайн решение приложения, разработанное в соответствии с принципами визуальной гармонии, удобства и соответствия тематике кондитерской продукции. Для этого использовались тёплые и аппетитные цветовые решения, вызывающие ассоциации с десертами, свежей выпечкой и уютом. Основная цветовая палитра включает нежные оттенки бежевого, розового, карамельного и белого, которые формируют позитивное впечатление и подчёркивают тематику сайта.

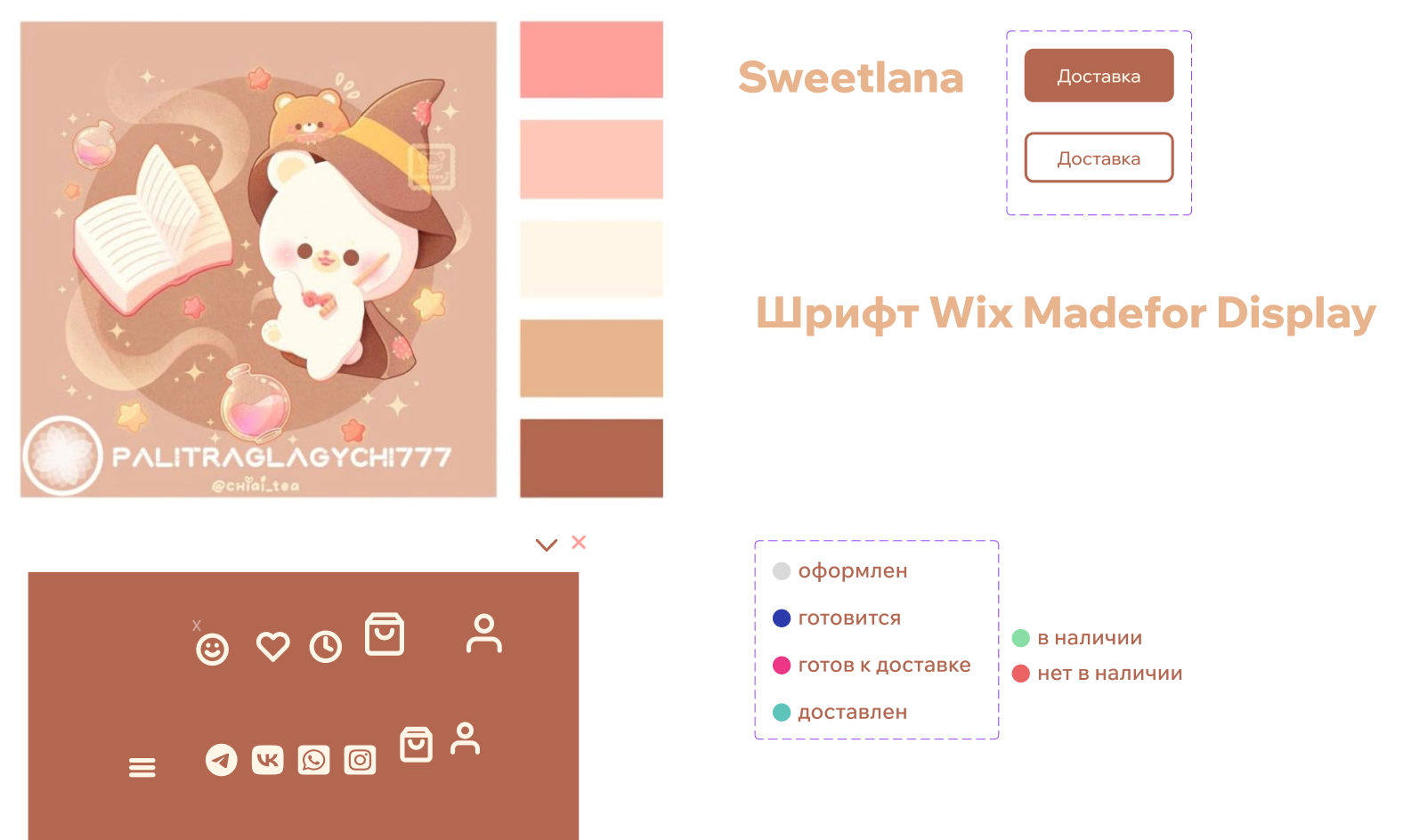


Рисунок 5 – Основной дизайн

В ходе проектирования графического интерфейса для сайта кондитерских изделий особое внимание было уделено разнообразию пользовательских сценариев и адаптации дизайна под различные устройства. Все макеты были выполнены в единой стилистике, поддерживающей фирменную цветовую гамму и тематические акценты.

На рисунках 6-7 изображены макеты главных страниц. Дизайн отражает концепцию “витрины”, где посетитель сразу видит привлекательные изображения продукции, акционные предложения и краткую информацию о кондитерской. Применены крупные визуальные элементы, плавные переходы и понятные навигационные блоки, что способствует формированию положительного первого впечатления и побуждает пользователя к дальнейшему изучению ассортимента.

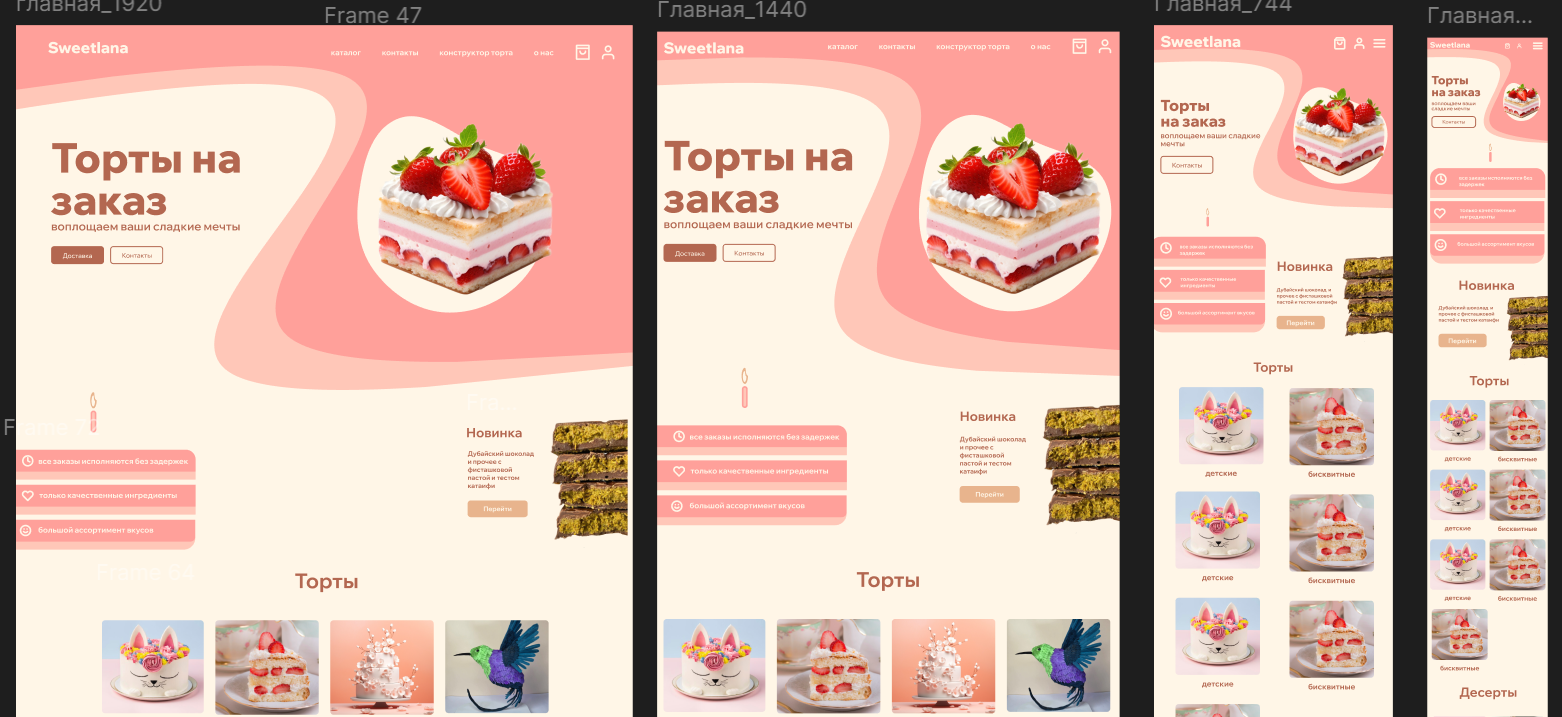


Рисунок 6 – Главная страница часть 1

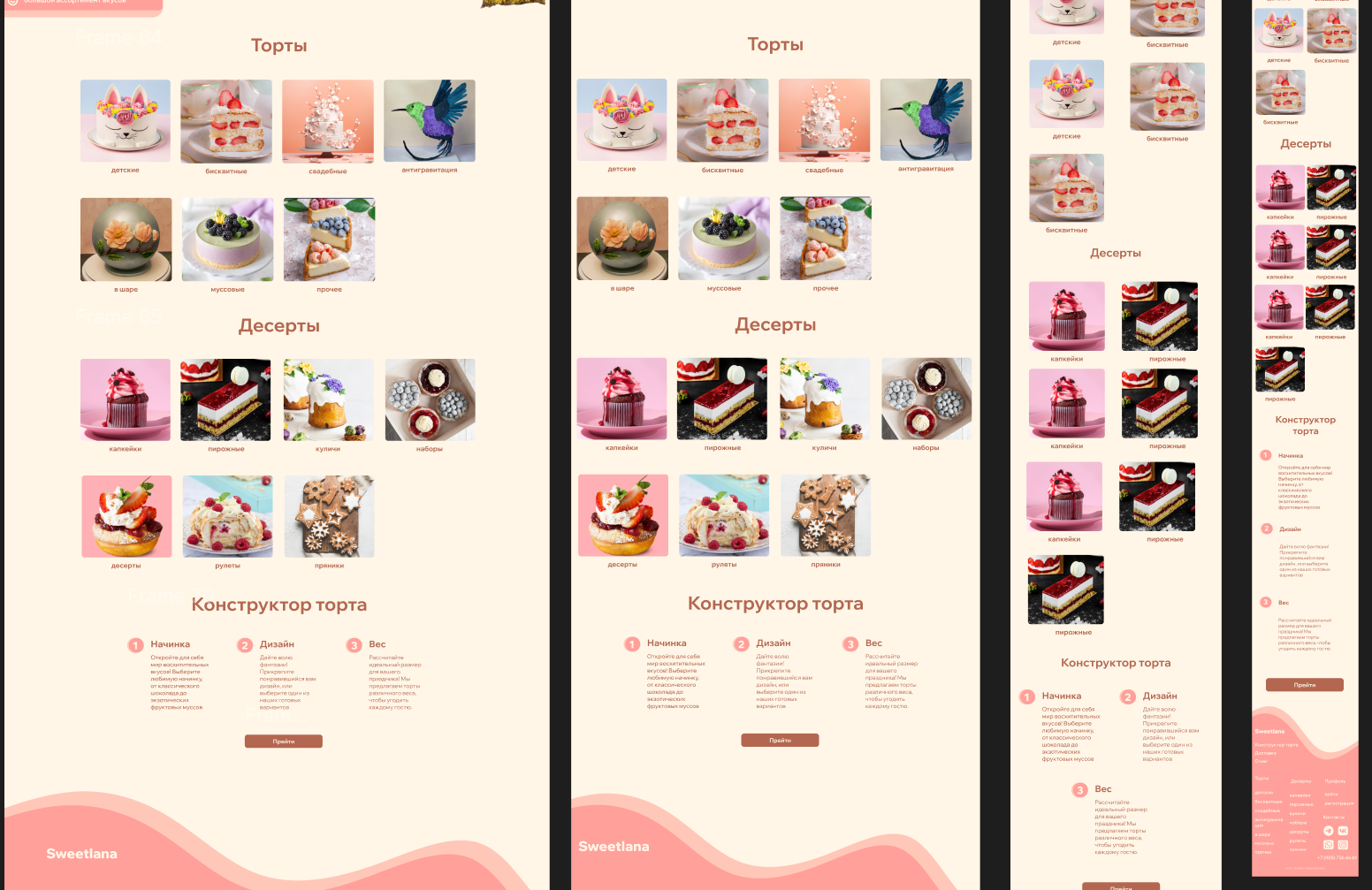


Рисунок 7 – Главная страница часть 2

Рисунок 8 демонстрирует реализацию бургер-меню для планшетов и мобильных устройств. Благодаря этому меню остаётся удобным и компактным на небольших экранах, а навигация по сайту — доступной вне зависимости от устройства. Также добавлены анимации, что улучшает стилистику.

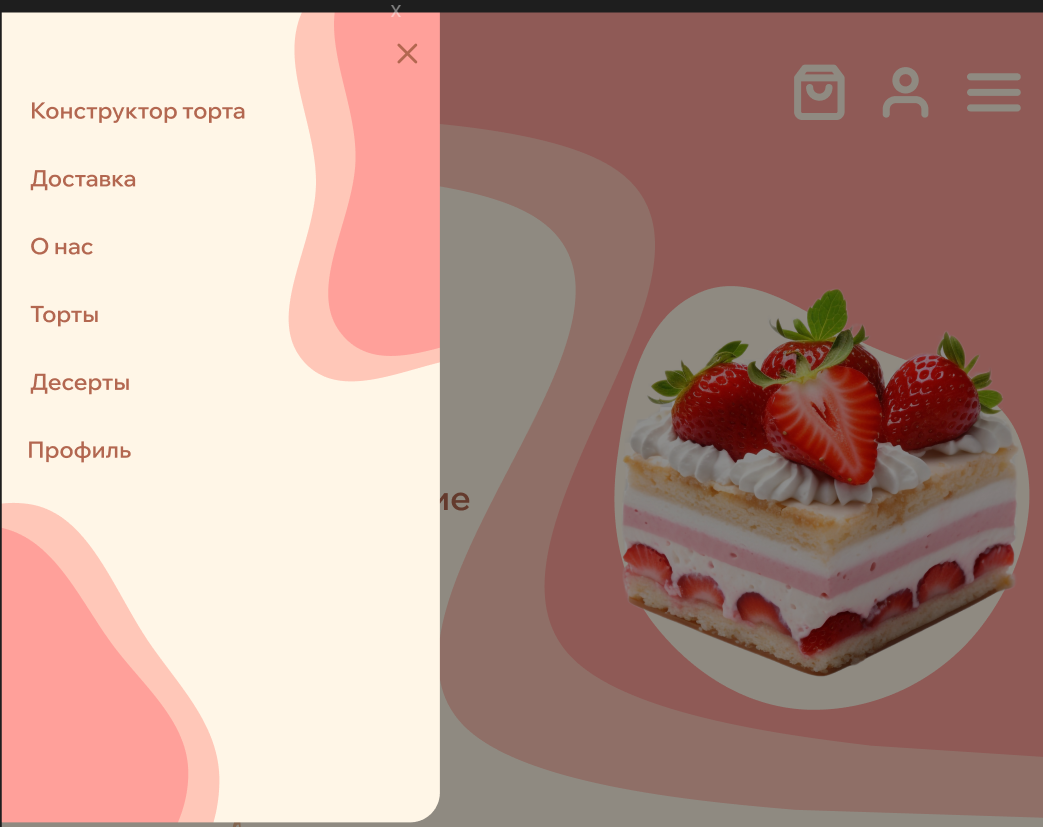


Рисунок 8 – Бургер-меню для планшетов и мобильных устройств

Рисунок 9 иллюстрирует страницу каталога для категории “Торты”. Здесь предусмотрена удобная фильтрация товаров, информативные карточки с изображениями и описаниями продукции. Каталог оформлен так, чтобы пользователь мог быстро найти интересующий товар и ознакомиться с его характеристиками.



Рисунок 9 – Часть страницы каталога категории Торты

Рисунок 10 показывает форму авторизации и регистрации. Эта часть интерфейса выполнена в лаконичном стиле, с акцентом на простоту и удобство ввода данных. Используются подсказки и визуальные индикаторы, помогающие пользователю корректно заполнить поля формы.



Рисунок 10 – форма авторизации и регистрации

Рисунок 11 отображает страницу профиля пользователя. На ней реализована возможность просмотра и редактирования личной информации, а также отслеживания истории заказов. Дизайн этой страницы также выдержан в общей цветовой гамме, что повышает узнаваемость и удобство интерфейса.



Рисунок 11 – Страница профиля

Рисунок 12 представляет макет административной панели сайта. Этот раздел предназначен для управления ассортиментом, обработкой заказов и взаимодействием с пользователями. Интерфейс админ-панели построен с учётом требований к удобству работы с большим объёмом информации: применяются структурированные таблицы, кнопки быстрого доступа к ключевым функциям и визуальные акценты для выделения важных элементов.

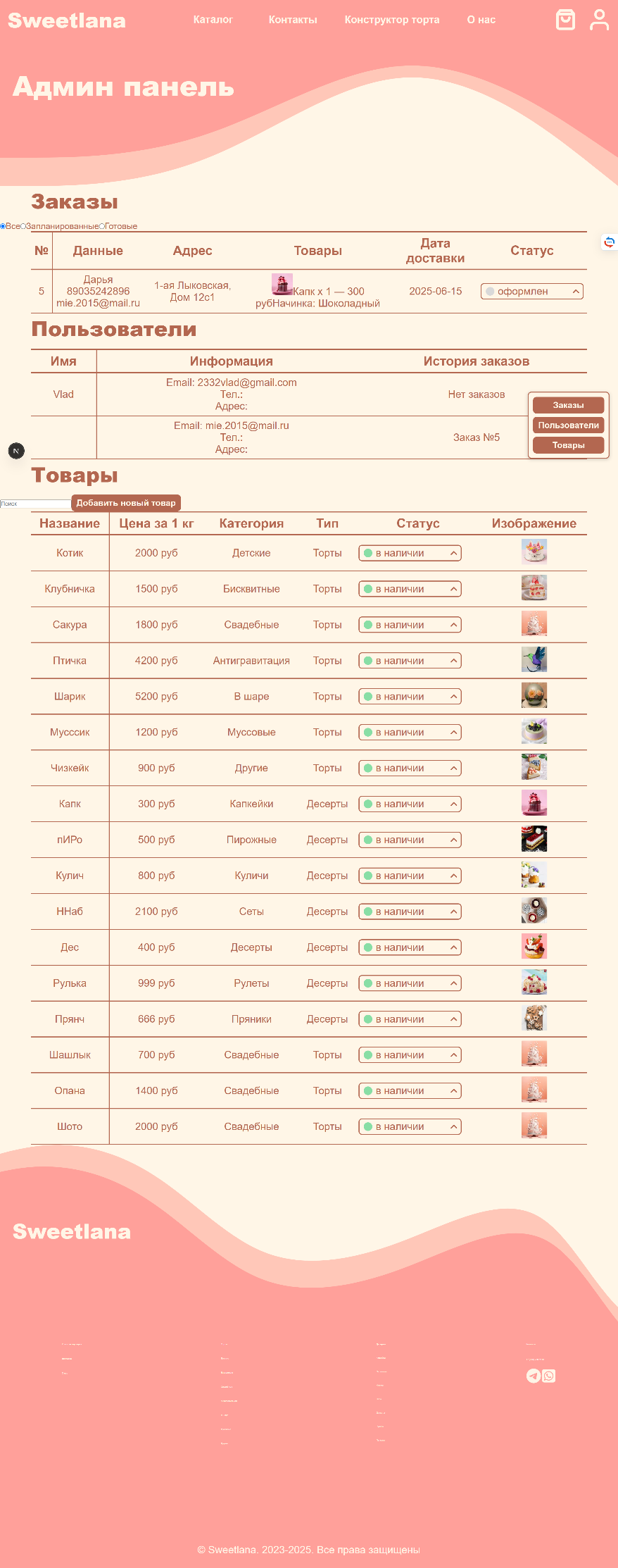


Рисунок 12 – Админ-панель

## **2.3. Проектирование пользовательского интерфейса.**

Разработка клиенсткой части веб-сайта кондитерских изделий начинается с базового шаблона приложения. Листинг 1 демонстрирует базовый компонент макета приложения, реализованный с помощью технологии Next.js App Router. App Router в Next.js — это система, которая автоматически управляет страницами и маршрутами сайта. С его помощью легко создавать и структурировать страницы, макеты и навигацию, а также реализовывать вложенные и динамические маршруты без сложной ручной настройки. Данный компонент RootLayout служит шаблоном для всех страниц сайта и определяет общую структуру пользовательского интерфейса.

Листинг 1 – Базовый компонент макета приложения

|  |
| --- |
| import "@/styles/globals.scss";  import { ReactNode } from "react";  import Header from "./components/Header";  import Footer from "./components/Footer";  export default function RootLayout({children,} :     {children: ReactNode;})    {    return (      <html lang="en">        <body>          <Header/>          <main>            {children} {/\* Контент страницы \*/}          </main>          <Footer/>        </body>      </html>    );  } |

В современных реалиях проект строится из отдельных компонентов, таких как Header, Footer, AuthForm. Каждый из них выполняет свою функцию и отвечает за определённую часть страницы. На листинге 2 - компонент Header, в котором реализованы логотип, меню навигации, иконки корзины и профиля, а также бургер-меню для мобильных устройств.  
Header автоматически меняет стили в зависимости от страницы, проверяет авторизацию пользователя, открывает форму входа или профиль, а также поддерживает быстрый переход к нужным разделам сайта. Такой подход позволяет сделать структуру сайта понятной и повторно использовать одни и те же элементы на разных страницах, предоставляя доступ ко всем основным функциям.

Листинг 2 – компонент Header

|  |
| --- |
| 'use client';  import React, { useState, useEffect } from 'react';  import { usePathname } from "next/navigation";  import Link from 'next/link';  import BurgerMenu from "./burger-menu/BurgerMenu";  import NavLinks from "./navLinks";  import Image from 'next/image';  import AuthForm from './auth-form/AuthForm';  import { useRouter } from 'next/navigation';  import { mainLinks, mobileLinks } from '../lib/nav-links';  import axios from 'axios';  import localforage from 'localforage';  interface User {    id: number;    name: string;    email: string;    role: string;  }  export default function Header() {    const pathname = usePathname();    const isHomePage = pathname === "/";    const isAboutPage = pathname === "/about";    const beige1Class = isHomePage ? "home-beige" : isAboutPage ? "about-beige" : "catalog-beige";    const pink1Class = isHomePage ? "home-pink" : isAboutPage ? "about-pink" : "catalog-pink";    const [showForm, setShowForm] = useState(false);    const [isReg, setIsReg] = useState(true);    const [user, setUser] = useState<User | null>(null);    const [scrollToDelivery, setScrollToDelivery] = useState(false);    const router = useRouter();    useEffect(() => {      if (scrollToDelivery && pathname === "/about") {        const timeout = setTimeout(() => {          document.getElementById("delivery")?.scrollIntoView({ behavior: "smooth" });          setScrollToDelivery(false);        }, 100);        return () => clearTimeout(timeout);      }      const fetchUserProfile = async () => {        try {          const response = await axios.get('/api/user/profile', { withCredentials: true });          if (response.data?.id) {            setUser(response.data);          } else {            setUser(null);          }        } catch (error) {          console.log('Error fetching user profile', error);          setUser(null);        }      };      fetchUserProfile();    }, [pathname, scrollToDelivery]);    const handleProfileClick = () => {      if (user) {        router.push('/user/profile');      } else {        setShowForm(true);      }    };    const handleCartClick = () => {      router.push('/cart')    };    const handleScroll = () => {      if (pathname !== "/about") {        setScrollToDelivery(true);        router.push("/about");      } else {        document.getElementById("delivery")?.scrollIntoView({ behavior: "smooth" });      }    };    const handleSubmit = () => {      setShowForm(false);    };    return (      <header className={beige1Class}>        <div className={pink1Class}>          <div className="navigation">            <Link key={'home'} href={'/'}>              <h3>Sweetlana</h3>            </Link>            <nav className="menu">              <NavLinks links={mainLinks} handleScroll={handleScroll} />            </nav>            <div className="header-icons">              <Image                src="/icons/shopping-cart.svg"                alt="cart"                height={51}                width={51}                onClick={handleCartClick} />              <Image                src="/icons/profile.svg"                alt="profile"                height={51}                width={51}                onClick={handleProfileClick} />              <BurgerMenu links={mobileLinks} />            </div>          </div>        </div>        {showForm && (          <AuthForm            onSubmit={handleSubmit}            isReg={isReg}            setIsReg={setIsReg}            setShowForm={setShowForm}          />        )}      </header>    )  } |

Страницы сайта совершают различные запросы к серверу для получения нужных данных. На листинге 3 – типовой пример загрузки данных с сервера на странице товара. После получения productId отправляется GET-запрос к API, данные сохраняются в состоянии и отображаются на странице. Такой подход обеспечивает актуальность информации о продуктах, позволяет отображать подробные карточки товаров, а также аналогично реализуется для получения данных о пользователях, заказах или списках товаров на других страницах сайта.

Листинг 3 – Запрос данных о товаре

|  |
| --- |
| const [product, setProduct] = useState<Product | null>(null);  useEffect(() => {      if (!productId) return;      axios        .get(`/api/products/product/${productId}`)        .then((response) => {          setProduct(response.data);        })        .catch((error) => {          console.log('Ошибка', error);        });  } |

Полноценная главная страница с адаптациями под различные устройства (см. Приложени А)

На листинге 4 представлен компонент формы входа и регистрации пользователя. Компонент поддерживает два режима работы: авторизация и регистрация нового пользователя. Для валидации данных используется библиотека react-hook-form. При успешной регистрации или входе пользователь перенаправляется на свою страницу профиля или административную панель (в зависимости от роли). Такой компонент обеспечивает безопасность, удобство и прозрачность работы с учётными записями на сайте, а также поддерживает проверку уникальности email при регистрации

Листинг 4 – Форма авторизации и регистрации (AuthForm)

|  |
| --- |
| 'use client';  import { useForm } from 'react-hook-form';  import axios from 'axios';  import Image from 'next/image';  import { useRouter } from 'next/navigation';  import styles from './auth-form.module.scss';  import { useState } from 'react';  interface AuthFormProps {    onSubmit: (data: FormData) => void;    isReg: boolean;    setIsReg: (isReg: boolean) => void;    setShowForm: (showForm: boolean) => void;  }  interface FormData {    name?: string;    email: string;    password: string;    confirmPassword?: string;  }  const AuthForm: React.FC<AuthFormProps> = ({ onSubmit, isReg, setIsReg, setShowForm }) => {    const { register, handleSubmit, watch, setError, formState: { errors }, trigger } = useForm<FormData>();    const [checkingEmail, setCheckingEmail] = useState(false);    const router = useRouter();    const onSubmitForm = async (data: FormData) => {      try {        const endpoint = isReg ? '/api/user/register' : '/api/user/login';        const payload = isReg          ? { name: data.name, email: data.email, password: data.password }          : { email: data.email, password: data.password };        const response = await axios.post(endpoint, payload, { withCredentials: true });        onSubmit(response.data);        if (response.data.role === 'admin') {          router.push('/user/admin');        } else {          router.push('/user/profile');        }        setShowForm(false);      } catch {        setError('email', { message: 'Неверные данные или пользователь уже существует' });      }  };  const validateEmailUniqueness = async (email: string) => {    if (!isReg) return true;    setCheckingEmail(true);    try {      const res = await axios.post('/api/user/check-email', { email });      return res.data?.unique ?? true;    } catch {      return false;    } finally {      setCheckingEmail(false);    }  };  return (    <div className={styles.modal}>      <div className={styles['modal-content']}>        <span className={styles["close-form"]} onClick={() => setShowForm(false)}>X</span>        <p className={styles.modalH}>{isReg ? 'Регистрация' : 'Авторизация'}</p>        <form onSubmit={handleSubmit(onSubmitForm)}>          {isReg && (            <input              type="text"              placeholder="Имя"              {...register('name', { required: 'Введите имя' })}              className={styles.pinkInput}            />          )}          {errors.name && <p style={{ color: 'red' }}>{errors.name.message}</p>}          <input            type="email"            placeholder="Почта"            {...register('email', {              required: 'Введите email',              pattern: {                value: /^\S+@\S+$/i,                message: 'Неверный формат email',              },              validate: validateEmailUniqueness            })}            onBlur={() => trigger('email')}            className={styles.pinkInput}          />          {checkingEmail && <p>Проверка email...</p>}          {errors.email && <p style={{ color: 'red' }}>{errors.email.message}</p>}          <input            type="password"            placeholder="Пароль"            {...register('password', {              required: 'Введите пароль',              minLength: {                value: 6,                message: 'Минимум 6 символов',              },            })}            className={styles.pinkInput}          />          {errors.password && <p style={{ color: 'red' }}>{errors.password.message}</p>}          {isReg && (            <input              type="password"              placeholder="Подтверждение пароля"              {...register('confirmPassword', {                required: 'Подтвердите пароль',                validate: value =>                  value === watch('password') || 'Пароли не совпадают',              })}              className={styles.pinkInput}            />          )}          {errors.confirmPassword && <p style={{ color: 'red' }}>{errors.confirmPassword.message}</p>}          <button type="submit">{isReg ? 'Зарегистрироваться' : 'Войти'}</button>          <p className={styles.law}>            Нажимая отправить, вы соглашаетесь с нашими условиями использования          </p>        </form>        <p className={styles.law} onClick={() => setIsReg(!isReg)}>          {isReg ? 'Уже есть аккаунт?' : 'Нет аккаунта?'}        </p>      </div>      <div className={styles['image-container']}>        <Image          src="/images/vectors/form.svg"          alt="form"          layout="responsive"          height={165}          width={782}        />      </div>    </div>  );  };  export default AuthForm; |

На листинге 5 представлен объект с переводами типов и категорий продукции для удобства отображения информации на сайте. Благодаря такой библиотеке легко поддерживать локализацию и выводить корректные русские названия категорий товаров в каталоге и фильтрах. Структура переводов позволяет централизованно управлять отображением информации на русском языке, а также гибко масштабировать список категорий при добавлении новых видов продукции.

Листинг 5 – Библиотека переводов

|  |
| --- |
| export const typeTranslations: { [key: string]: string } = {    cake: "Торты",    dessert: "Десерты"  };  export const categoryTranslations: { [key: string]: { [key: string]: string } } = {    cake: {      child: "Детские",      biscuit: "Бисквитные",      wedding: "Свадебные",      gravit: "Антигравитация",      ball: "В шаре",      muss: "Муссовые",      others: "Другие"    },    dessert: {      cupcakes: "Капкейки",      pies: "Пирожные",      kuliches: "Куличи",      sets: "Сеты",      desserts: "Десерты",      rulets: "Рулеты",      breads: "Пряники"    }  };  export const categorTranslations: { [key: string]: string } = {      child: "Детские",      biscuit: "Бисквитные",      wedding: "Свадебные",      gravit: "Антигравитация",      ball: "В шаре",      muss: "Муссовые",      others: "Другие",      cupcakes:"Капкейки",      pies:"Пирожные",      kuliches:"Куличи",      sets:"Сеты",      desserts:"Десерты",      rulets:"Рулеты",      breads:"Пряники"    }; |

На листинг 6 реализуется страница корзины. Он отвечает за отображение всех выбранных товаров пользователя, изменение количества, удаление позиций, подсчёт итоговой стоимости, а также оформление заказа через заполнение формы доставки и выбора способа оплаты. Компонент взаимодействует с локальным хранилищем браузера (localforage) для сохранения данных корзины и с сервером для оформления заказа. После успешного подтверждения заказа корзина очищается, а пользователь видит уведомление об успешной отправке. Такой подход обеспечивает удобство оформления заказов и интеграцию клиентской и серверной логики.

Листинг 6 – Корзина товаров

|  |
| --- |
| 'use client'  import { useEffect, useState } from 'react';  import localforage from 'localforage';  import { useRouter } from 'next/navigation';  import axios from 'axios';  import Image from 'next/image';  import styles from './styles/cart.module.scss'  interface CartItem {      uid: string;      productId: number;      name: string;      price: number;      quantity: number;      weight: number;      type: string;      img: string;      fillingId: number;      filling: string;      addons: string[];  }  interface UserData {    id: number;    name: string;    phone: string;    email: string;    [key: string]: any;  }  export default function CartPage() {      const [cart, setCart] = useState<CartItem[]>([]);      const [showModal, setShowModal] = useState(false);      const [totalPrice, setTotalPrice] = useState(0);      const [userData, setUserData] = useState<UserData | null>(null);      const [formData, setFormData] = useState({          name: '',          phone: '',          email: '',          address: '',          deliveryDate: '',          deliveryMethod: '',          paymentMethod: 'cash',          comment: ''      });      const router = useRouter();      useEffect(() => {          const fetchCart = async () => {              const storedCart = (await localforage.getItem<CartItem[]>('cart')) || [];              setCart(storedCart);              const total = storedCart.reduce((sum: number, item: CartItem) => sum + item.price \* item.quantity, 0);              setTotalPrice(total);          };          const fetchUserProfile = async () => {              try {                  const response = await axios.get('/api/user/profile', { withCredentials: true })                  if (response.data) {                      setUserData(response.data);                      console.log('User profile data:', response.data);                      setFormData((prevData) => ({                          ...prevData,                          name: response.data.name || '',                          phone: response.data.phone || '',                          email: response.data.email || '',                      }));                  } else {                      console.log('User not authenticated');                  }              } catch (error) {                  console.log('Error fetching user profile', error);              }          };          if (showModal) {              const timer = setTimeout(() => {                  router.push('/user/profile');              }, 2000);              return () => clearTimeout(timer);          }          fetchCart();          fetchUserProfile();      }, [showModal, router]);      const handleChangeQuantity = async (uid: string, delta: number) => {          const updated = cart.map(item =>              item.uid === uid ? { ...item, quantity: Math.max(1, item.quantity + delta) } : item          );          await localforage.setItem('cart', updated);          setCart(updated);          setTotalPrice(updated.reduce((sum, i) => sum + i.price \* i.quantity, 0));      };      const handleRemoveItem = async (itemId: number) => {          const updatedCart = cart.filter((item) => item.productId !== itemId);          await localforage.setItem('cart', updatedCart);          setCart(updatedCart);          const total = updatedCart.reduce((sum: number, item: CartItem) => sum + item.price \* item.quantity, 0);          setTotalPrice(total);      };      const handleSubmit = async (event: React.FormEvent) => {          event.preventDefault();          const isAuthenticated = userData && userData.id;          if (isAuthenticated) {              const orderData = {                  user\_id: userData.id,                  user\_name: userData.name || formData.name,                  user\_phone: userData.phone || formData.phone,                  user\_email: formData.email,                  address: formData.address,                  delivery\_date: formData.deliveryDate,                  delivery\_method: formData.deliveryMethod,                  total\_price: totalPrice,                  status: 'оформлен',                  comment: formData.comment,                  items: cart.map(item => ({                      productId: item.productId,                      fillingId: item.fillingId,                      addons: item.addons,                      quantity: item.quantity,                      weight: item.weight,                      price: item.price,                  }))              };              console.log(orderData)              try {                  await axios.post('/api/orders', orderData);                  await localforage.removeItem('cart');                  setCart([]);                  setFormData({                      name: '',                      phone: '',                      email: '',                      address: '',                      deliveryDate: '',                      deliveryMethod: '',                      paymentMethod: 'cash',                      comment: ''                  });                  setShowModal(true);              } catch (error) {                  console.log('Ошибка при оформлении заказа:', error);              }          } else {              alert('Пожалуйста, авторизуйтесь для оформления заказа.');              // router.push('/login'); // Перенаправление на страницу логина          }      };      const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {          const { name, value } = e.target;          setFormData((prevData) => ({              ...prevData,              [name]: value,          }));      };      return (          <div className={styles['make-order']}>              <h3>Корзина</h3>              <section>                  {cart.length === 0 ? <p>Корзина пуста</p> : (                      <div className={styles.products}>                          {cart.map((item) => (                              <div className={styles.product} key={item.productId}>                                  <Image                                      className={styles['item-img']}                                      src={`/images/catalog/${item.type}/${item.img}`}                                      width={150}                                      height={150}                                      alt={item.name}                                  />                                  <div>                                      <p>Торт &quot;{item.name}&quot;</p>                                      <p>Начинка: {item.filling}</p>                                      <p>Кол-во в коробке:</p>                                  </div>                                  <div>                                      <p>{item.price} руб</p>                                      <p className={styles.quantityProduct}>                                          <span className={styles.changeQuantity} onClick={() => handleChangeQuantity(item.uid, -1)}>–</span>                                          <span>{item.quantity}</span>                                          <span className={styles.changeQuantity} onClick={() => handleChangeQuantity(item.uid, +1)}>+</span>                                      </p>                                      <p>{item.price \* item.quantity} руб</p>                                      <Image                                          className={styles.trash}                                          src={'/icons/trash1.svg'}                                          alt='trash'                                          width={30}                                          height={30}                                          onClick={() => handleRemoveItem(item.productId)}                                      />                                  </div>                                  {/\* <hr /> \*/}                              </div>                          ))}                          <h2>Итого: {totalPrice} руб</h2>                      </div>                  )}                  <div className={styles.order}>                      <h4>Оформление заказа</h4>                      <form onSubmit={handleSubmit} className={styles['order-block']}>                          <div>                              <p>Способы доставки</p>                              <label>                                  <input                                      type="radio"                                      name="deliveryMethod"                                      value="pickup"                                      checked={formData.deliveryMethod === 'pickup'}                                      onChange={handleInputChange}                                  />                                  Самовывоз                              </label>                              <label>                                  <input                                      type="radio"                                      name="deliveryMethod"                                      value="delivery"                                      checked={formData.deliveryMethod === 'delivery'}                                      onChange={handleInputChange}                                  />                                  Доставка в пределах МКАД                              </label>                          </div>                          <div className={styles['order-inputs']}>                              <input                                  type="text"                                  name="name"                                  placeholder="Имя"                                  value={formData.name || ''}                                  onChange={handleInputChange}                              />                              <input                                  type="number"                                  name="phone"                                  placeholder="Телефон"                                  value={formData.phone || ''}                                  onChange={handleInputChange}                              />                              <input                                  type="email"                                  name="email"                                  placeholder="E-mail"                                  value={formData.email || ''}                                  onChange={handleInputChange}                              />                              <input                                  type="text"                                  name="address"                                  placeholder="Адрес доставки"                                  value={formData.address || ''}                                  onChange={handleInputChange}                              />                              <input                                  type="date"                                  name="deliveryDate"                                  value={formData.deliveryDate || ''}                                  onChange={handleInputChange}                              />                              <input                                  type="text"                                  name="comment"                                  placeholder="Комментарий"                                  value={formData.comment}                                  onChange={handleInputChange}                              />                          </div>                          <hr />                          <div>                              <p>Способы оплаты</p>                              <label>                                  <input                                      type="radio"                                      name="paymentMethod"                                      value="cash"                                      checked={formData.paymentMethod === 'cash'}                                      onChange={handleInputChange}                                  />                                  Наличные                              </label>                              <label>                                  <input                                      type="radio"                                      name="paymentMethod"                                      value="card"                                      checked={formData.paymentMethod === 'card'}                                      onChange={handleInputChange}                                  />                                  Карта                              </label>                          </div>                          <hr />                          <div>                              <p>Заказ позиций {totalPrice} рублей</p>                              <p>Итого: {totalPrice}</p>                          </div>                          <button type="submit" className={styles.submitCart}>Подтвердить заказ</button>                      </form>                  </div>              </section>              {showModal && (                  <div className={styles.modalOverlay}>                      <div className={styles.modalContent}>                          <p>Заказ успешно оформлен!</p>                      </div>                  </div>              )}          </div>      );  } |

## **2.3 Разработка серверной части приложения**

Для обеспечения полной функциональности веб-приложения была спроектирована и реализована серверная архитектура, поддерживающая такие ключевые функции, как регистрация и авторизация пользователей, управление товарами, оформление заказов и работа с корзиной. В качестве основы для бэкенд-разработки был выбран фреймворк NestJS версии 11.x — современное решение на базе Node.js, предоставляющее модульную структуру, поддержку TypeScript и удобные механизмы Dependency Injection. Это позволило реализовать с применением современных практик разработки RESTful API и объектно-реляционного взаимодействия с базой данных, что обеспечило надёжность, гибкость и масштабируемость решения.

Модульная архитектура и структура приложения  
Серверная часть разделена на тематические модули: users, products, orders, auth, каждый из которых содержит собственные контроллеры, сервисы и сущности базы данных. Такой подход способствует удобному масштабированию и сопровождению проекта.

На листинге 1 работа с реляционной базой данных MySQL, в которой используется TypeORM, позволяющий описывать структуру таблиц с помощью entity-классов.

Листинг 1 – Пример entity-класса товара

|  |
| --- |
| import { Entity, PrimaryGeneratedColumn, Column } from 'typeorm';  @Entity('products')  export class Product {    @PrimaryGeneratedColumn()    id: number;    @Column()    name: string;    @Column()    price: number;    @Column()    img: string;    @Column()    type: string;    @Column()    category: string;      @Column({ default: true })    in\_stock: boolean;    @Column()    amount: number;  } |

На листинге 2 реализация REST API.   
Основная бизнес-логика реализована в сервисах и контроллерах.

Листинг 2 – Пример контроллера получения всех товаров

|  |
| --- |
| import { Controller, Get } from '@nestjs/common';  import { ProductsService } from './products.service';  @Controller('products')  export class ProductsController {  constructor(private readonly productsService: ProductsService) {}  @Get()  findAll() {  return this.productsService.findAll();  }  }} |

На листинге 3 показа валидация данных и DTO. Для проверки корректности входных данных для бд применяются классы DTO с использованием декораторов библиотеки class-validator.

Листинг 3 – Пример DTO для создания пользователя

|  |
| --- |
| import { Entity, PrimaryGeneratedColumn, Column, OneToMany } from 'typeorm';  import { UserRole } from './dto/user-role.enum';  import { Order } from '../orders/order.entity';  @Entity('users')  export class User {    @PrimaryGeneratedColumn()    id: number;    @Column({ unique: true })    email: string;    @Column()    password: string;    @Column()    name: string;    @Column()    phone: string;    @Column()    address: string;     @Column({      type: 'enum',      enum: UserRole,      default: UserRole.USER,    })    role: UserRole;     @OneToMany(() => Order, (order) => order.user)    orders: Order[];  } |

На листинге 5 – оформление заказа. Реализация бизнес-логики оформления заказа осуществляется в сервисе заказов.

Листинг 5 – Пример метода создания заказа

|  |
| --- |
| async createOrder(createOrderDto: CreateOrderDto): Promise<Order> {  const order = this.ordersRepository.create(createOrderDto);  return await this.ordersRepository.save(order);  }} |

На листинге 6 – Сессии и авторизация через express-session. Для управления авторизацией и хранением состояния пользователя используется middleware express-session в связке с nestjs-session. Сессия создаётся при входе пользователя, далее сервер автоматически определяет пользователя по session ID, который хранится в cookie.

Листинг 6 – Настройки сессий в main.ts:

|  |
| --- |
| import \* as session from 'express-session';  app.use(  session({  secret: process.env.SESSION\_SECRET,  resave: false,  saveUninitialized: false,  cookie: { maxAge: 86400000 },  }),  ); |

## **2.4. Отладка и тестирование веб-приложения**

Отладка и тестирование веб-приложений являются ключевыми этапами разработки, поскольку они напрямую влияют на стабильность и качество конечного продукта. Тестирование позволяет выявить критические ошибки до запуска и обеспечивает корректную работу всех функций. Особое внимание уделяется кроссбраузерной совместимости, адаптивности интерфейса и нагрузочному тестированию, чтобы гарантировать бесперебойную работу платформы при любом уровне посещаемости. В таблице 1 представлены результаты отладки и тестирования программы.

Таблица 1 – Тестирование формы регистрации

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № теста | Входные данные | Вводимое значение | Ожидаемая  реакция программы | Фактическая реакция программы | Ошибка выявлена |
| 1 | Имя  Почта  Пароль | “”  “”  “” | Сообщение «Заполните обязательное поле» | Рисунок Б.1 | Нет |
| 2 | Имя  Почта  Пароль | “”  “занятая почта”  “” | Сообщение «Данная почта уже занята» | Рисунок Б.3 | Нет |
| 3 | Имя  Фамилия  Почта  Пароль | “”  “”  “”  llll | Сообщения:  «Заполните обязательное поле»,  «Пароль должен содержать минимум 8 символов» | Рисунок Б.2 | Нет |
| 3 | Имя  Почта  Пароль | Алина  miemaeko@gmail.com  5367421 | Успешная регистрация | Рисунок Б.3 | Нет |

## **2.5. Публикация приложения на хостинге**

Завершающим этапом реализации дипломного проекта стало размещение веб-приложения «Sweetlana» на облачной платформе Railway. Этот шаг позволил обеспечить круглосуточную доступность сайта для пользователей, протестировать работоспособность всех компонентов в реальных условиях и продемонстрировать проект заказчику или комиссии. Railway был выбран в качестве хостинга благодаря простоте развертывания, поддержке современных технологий, возможности быстрого масштабирования и удобному управлению переменными окружения (см. Рисунок 13-14).

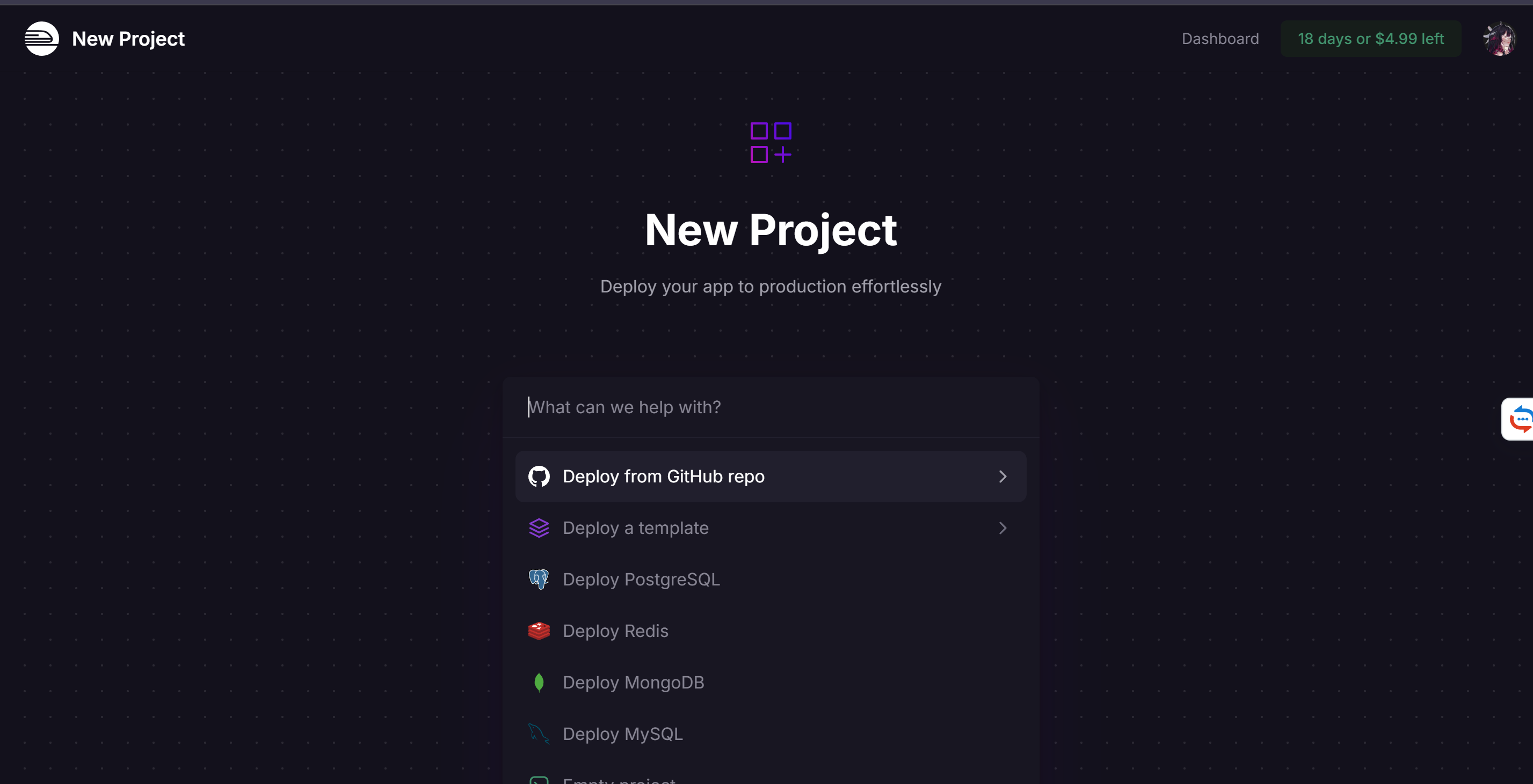


Рисунок 13 – Создание проекта в Railway

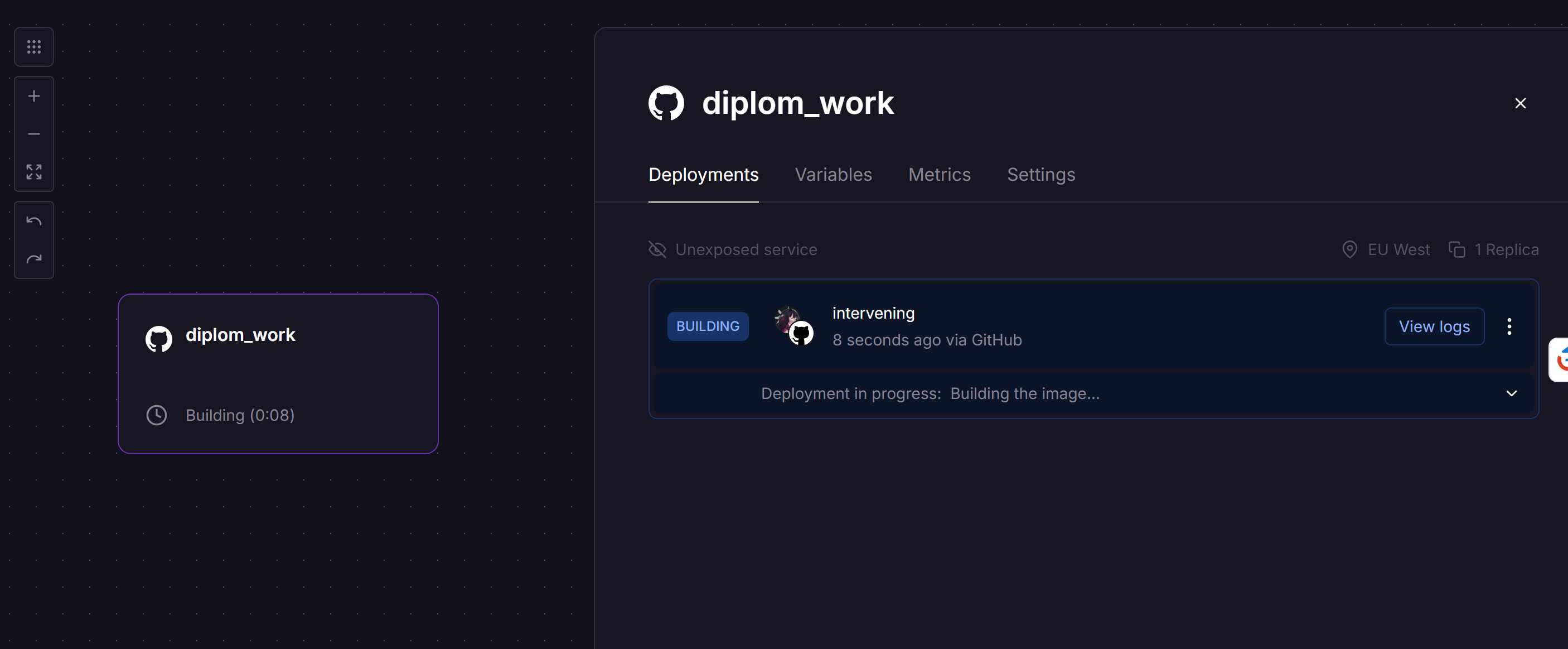


Рисунок 14 – Подключение репозитория

В процессе публикации отдельно были размещены клиентская (Next.js) и серверная (NestJS) части приложения, а также база данных MySQL. Для каждого компонента был создан отдельный проект на Railway, что позволило гибко настраивать взаимодействие между сервисами и упростило обновление кода. Платформа Railway предоставляет инструменты для автоматического деплоя из репозитория GitHub, что ускоряет внедрение изменений и повышает надёжность обслуживания приложения.

После публикации веб-сайт стал доступен по уникальному доменному имени, предоставляемому Railway, и прошёл финальное тестирование на корректность работы всех функций, производительность и стабильность. Использование облачного хостинга позволило убедиться в готовности проекта к реальной эксплуатации и обеспечить демонстрацию его возможностей в любой момент времени.

## **2.6. Разработка руководства по использованию приложения**

Для предоставление понятной и структурированной информации о функциональных возможностях сайта, способах навигации, регистрации, оформления заказов и администрирования было разработано руководство по использованию.

Пользователя встречает главная страница с несколькими категориями и видов кондитерских изделий

Для пользователей сайта руководство доступные следующие действия:

* просмотру каталога товаров с фильтрацией по категориям и ценам;
* добавлению выбранных товаров в корзину;
* оформлению заказа с
* регистрации и авторизации в системе;

После входа пользователь сможет оформить с указанием контактных данных и адреса доставки, а также просматривать истории и статусов заказов в личном кабинете.

Для администратора системы подготовлены отдельные функции:

* процесс входа в панель управления;
* добавление, редактирование и удаление товаров;
* управление списком пользователей и заказов;
* настройку параметров сайта (категории товаров, начинки).

Интерфейс приложения сопровождается подсказками и информационными сообщениями, которые помогают пользователю правильно взаимодействовать с функциями сайта. В случае ошибок система выводит понятные уведомления, объясняющие проблему и предлагающие пути её решения.

# **ЗАКЛЮЧЕНИЕ**

В рамках выполнения дипломного проекта была успешно реализована основная цель — разработка современного и функционального веб-сайта «Sweetlana» для индивидуального предпринимателя, занимающегося производством и продажей кондитерских изделий. Созданная система решает ключевые задачи по продвижению продукции, расширению рынка сбыта и формированию долгосрочных отношений с покупателями за счёт современных цифровых технологий.

В процессе работы над проектом был пройден полный цикл разработки веб-приложения: от анализа требований и проектирования архитектуры до внедрения, публикации и тестирования. Решение охватывает как серверную, так и клиентскую части, что позволило создать цельную и гибкую платформу для ведения бизнеса в цифровой среде.

Серверная часть (ответственная — Емельянова Дарья Игоревна):

* Спроектирован и реализован REST API для всех ключевых функций сайта: регистрация и аутентификация пользователей, обработка заказов, управление ассортиментом, работа с корзиной и обратной связью.
* Разработана и реализована структура базы данных с учётом особенностей предметной области: хранение информации о продуктах, заказах, пользователях и дополнительных опциях.
* Обеспечена валидация и обработка ошибок на стороне сервера, реализованы механизмы авторизации и разграничения прав доступа для пользователей и администратора.
* Настроены процессы тестирования, оптимизации и деплоймента, серверная часть и база данных успешно размещены на облачной платформе Railway.

Клиентская часть (ответственная — Прицепа Илана Александровна):

* Спроектирована архитектура клиентского приложения с использованием современных инструментов и подходов (Next.js, React, TypeScript).
* Разработан адаптивный и эстетически привлекательный пользовательский интерфейс с учётом лучших практик UX/UI и специфики кондитерской продукции.
* Реализован основной функционал: каталог продукции с фильтрацией, просмотр детальной информации о товарах, оформление заказов, регистрация и авторизация, а также личный кабинет пользователя.
* Осуществлена интеграция с REST API, настроено управление состоянием и оптимизирована производительность приложения.
* Клиентская часть размещена на хостинге Railway с обеспечением стабильной работы и быстрой доставки изменений.

Реализация этого проекта позволила не только получить готовый инструмент для продвижения и увеличения продаж кондитерских изделий, но и существенно расширить практические навыки в области веб-разработки, архитектурного проектирования и пользовательского опыта. В ходе работы применялись современные технологии и лучшие практики: компонентный подход, автоматизация процессов деплоя, адаптивный дизайн, безопасная обработка пользовательских данных и интеграция с внешними сервисами.

Перспективы развития:

* Добавление системы аналитики продаж и отчётности для владельца бизнеса.
* Использование телеграмм бота для удобного управления заказами.
* Интеграция с внешними платёжными сервисами и службами доставки.
* Внедрение новых маркетинговых инструментов: акции, рассылки, программа лояльности.

Таким образом, созданный веб-сайт «Sweetlana» является эффективной платформой для ведения бизнеса на современном рынке кондитерских изделий, способствуя укреплению позиций предприятия, привлечению новых клиентов и формированию положительного имиджа компании.

# **СПИСОК ЛИТЕРАТУРЫ (ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ) И ИНТЕРНЕТ-РЕСУРСОВ**

Учебная и научная литература:

1. ГОСТ Р 7.0.12-2011 Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила. – М.: Стандартинформ, 2012. – 61 с.
2. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2010. – 92 с.
3. ГОСТ 7.32-2017 Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2017. – 47 с.
4. ГОСТ 7.82-2001 Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. – 39 с.
5. ГОСТ Р 7.0.100-2018 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2018. – 122 с.
6. ГОСТ Р 7.0.5-2008 Библиографическая ссылка. Общие требования и правила составления. – М.: Стандартинформ, 2008. – 32 с.
7. Единая система программной документации. – М.: Стандартинформ, 2005. – 128 с.

Интернет документы:

1. Разработка веб-приложений: основные этапы. – [Электронный ресурс]. – URL: <https://www.purrweb.com/ru/blog/kak-sozdat-veb-prilozhenie/> (дата обращения: 02.03.2024).
2. Документация Next.js – [Электронный ресурс]. – URL: https://nextjs.org/docs (дата обращения: 02.03.2024).
3. Документация NestJS – [Электронный ресурс]. – URL: https://docs.nestjs.com (дата обращения: 02.03.2024)
4. Документация MySQL. – [Электронный ресурс]. – URL: <https://dev.mysql.com/doc/> (дата обращения: 02.03.2024).
5. Разработка интерфейсов: гайд по UX/UI [Электронный ресурс]. – URL: <https://uxdesign.cc/> (дата обращения: 02.03.2024).
6. Разработка и дизайн интернет-магазинов: современные подходы [Электронный ресурс]. – URL: https://vc.ru/marketing/296308-dizayn-internet-magazina (дата обращения: 02.03.2024)
7. Официальная документация TypeScript [Электронный ресурс]. — URL: https://www.typescriptlang.org/docs/ (дата обращения: 02.03.2024).
8. Прототипирование и тестирование интерфейсов в Figma [Электронный ресурс]. – URL: https://habr.com/ru/articles/543404/ (дата обращения: 02.03.2024)
9. Стандарты безопасности хранения паролей [Электронный ресурс]. – URL: https://cheatsheetseries.owasp.org/cheatsheets/Password\_Storage\_Cheat\_Sheet.html дата обращения: 02.03.2024).
10. Официальная документация Axios [Электронный ресурс]. — URL: https://axios-http.com/docs/intro (дата обращения: 02.03.2024).
11. Документация SASS [Электронный ресурс]. — URL: https://sass-lang.com/documentation/ (дата обращения: 02.03.2024).
12. Разработка и дизайн интернет-магазинов: современные подходы [Электронный ресурс]. – URL: https://vc.ru/marketing/296308-dizayn-internet-magazina (дата обращения: 02.03.2024).
13. Документация Railway [Электронный ресурс]. – URL: https://docs.railway.app/ (дата обращения: 02.03.2024).
14. Habr.com — сообщество и публикации по веб-разработке [Электронный ресурс]. — URL: https://habr.com/ru/hub/webdev/ (дата обращения: 02.03.2024).

# **ПРИЛОЖЕНИЕ**

**ПРИЛОЖЕНИЕ А**

Листинг 2 полная главная страница

|  |
| --- |
| export default function HomePage() {    return (      <div>        <section className={styles['first-block']}>          <div className={styles['first-block-half1']}>            <h1>              <span>Торты</span>              <span>на заказ</span>            </h1>            <p>воплощаем ваши сладкие мечты</p>            <div className={styles['buttons-cake']}>              <button className={styles['button-delivery']}>Доставка</button>              <button className={styles['button-contacts']}>Контакты</button>            </div>          </div>          <div className={styles['first-block-half2']}>            <Image              className={styles["vector-cake"]}              src="/images/vectors/cakeMain.svg"              alt="vector-cake"              height={612}              width={600} />            <Image              className={styles["strawberry-cake"]}              src="/images/page/pageCake.png"              alt="strawberry-cake"              height={558}              width={558} />          </div>        </section>        <section className={styles.news}>          <div className={styles.benefits}>            <div className={styles.candle}>              <Image                src="/images/vectors/candle.svg"                alt="candle"                height={125}                width={20} />            </div>            <div className={styles.firstLayerofCake}>              <div className={styles['layer-content']}>                <Image                  src="/icons/clock.svg"                  alt="clock"                  height={37}                  width={37} />                <span>все заказы исполняются без задержек</span>              </div>              <div className={styles.groundLayerofCake}></div>            </div>            <div className={styles.secondLayerofCake}>              <div className={styles['layer-content']}>                <Image                  src="/icons/heart.svg"                  alt="heart"                  height={37}                  width={37} />                <span>только качественные ингредиенты</span>              </div>              <div className={styles.groundLayerofCake}></div>            </div>            <div className={styles.thirdLayerofCake}>              <div className={styles['layer-content']}>                <Image                  src="/icons/smile.svg"                  alt="smile"                  height={37}                  width={37} />                <span>большой ассортимент вкусов </span>              </div>              <div className={styles.groundLayerofCake}></div>            </div>          </div>          <div className={styles.newChocolate}>            <div>              <p className={styles.newChocolateH}>Новинка</p>              <p>Дубайский шоколад  и прочее с фисташковой пастой и тестом катаифи</p>              <button>Перейти</button>            </div>            <div>              <Image                className={styles.chocolateBar}                src="/images/page/chocolateBar.png"                alt="chocolateBar"                height={424}                width={318} />            </div>          </div>        </section>        <section className={styles.cakes}>          <h4>Торты</h4>          <Categories type='cake' />        </section>        <section className={styles.desserts}>          <h4>Десерты</h4>          <Categories type='dessert' />        </section>        <section className={styles.builder}>          <h4>Конструктор Торта</h4>          <div className={styles['builder-blocks']}>            <div className={styles['blocks-adv']}>              <div className={styles.circle}>1</div>              <div>                <p>Начинка</p>                <p>Откройте для себя мир восхитительных вкусов! Выберите любимую начинку, от классического шоколада до экзотических фруктовых муссов</p>              </div>            </div>            <div className={styles['blocks-adv']}>              <div className={styles.circle}>2</div>              <div>                <p>Дизайн</p>                <p>Дайте волю фантазии! Прикрепите понравившийся вам дизайн, или выберите один из наших готовых вариантов</p>              </div>            </div>            <div className={styles['blocks-adv']}>              <div className={styles.circle}>3</div>              <div>                <p>Вес</p>                <p>Рассчитайте идеальный размер для вашего праздника! Мы предлагаем торты различного веса, чтобы угодить каждому гостю.</p>              </div>            </div>          </div>          <button className={styles['button-builder']}><Link href='/builder'>Перейти</Link></button>        </section>      </div>    );  } |