Министерство образования Свердловской области

ГАПОУ СО «Уральский государственный колледж имени И.И. Ползунова»

ДП. 09.02.07.06.ПЗ

УТВЕРЖДАЮ

Заведующий кафедрой ИСиАУ

\_\_\_\_\_\_\_\_\_\_/М.А. Гузачев/

ОСОБЕННОСТИ ПРОЕКТИРОВАНИЯ САЙТА ИНТЕРНЕТ-МАГАЗИНА

"ГЕОMUSIC"

Пояснительная записка

Руководитель

\_\_\_\_\_\_\_\_\_/И.А. Орабинский /

Разработал

\_\_\_\_\_\_\_\_\_\_/Н.И. Георгиев/

Екатеринбург 2025

Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Уральский государственный колледж имени И.И. Ползунова»

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ:  Зам. директора по учебно-производственной работе  \_\_\_\_\_\_\_\_\_\_\_\_\_\_/И.И. Репина/  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

**ЗАДАНИЕ**

**НА ДИПЛОМНОЕ ПРОЕКТИРОВАНИЕ**

Студенту: Георгиеву Николаю Игоревичу

Специальности:09.02.07 «Информационные системы и программирование»

Группы: ИП-401

**1** **Тема дипломного проекта:** Особенности проектирования сайта интернет магазина "ГеоMusic"

**2 Исходные данные дипломного проекта:** Программное обеспечение: WebStorm, PostgreSQL. Языки программирования: JavaScript (фреймворк React и Express). Среда выполнения: Node.js.

**3 Содержание пояснительной записки:**

3.1 Описание предмета исследования:

3.1.1. Анализ задания на дипломное проектирование

3.1.2. Описание выбранных технологий и средств разработки

3.1.3 Описание преимуществ и недостатков выбранной технологии

3.1.4. Описание возможностей и ограничений

3.2 Описание разработки сайта

3.2.1. Описание разработки базы данных в PosgreSQL

3.2.2 Описание разработки серверной части сайта

3.2.3 Описание разработки клиентской части сайта

**4 Практическая часть**

4.1 Выполнение анализа задания на дипломное проектирование

4.2 Пример использования и развертывания сайта

4.2.1 Демонстрация функций сайта

4.2.2 Демонстрация работоспособности сайта

4.3 Работа с источниками и обработка информации

4.4 Подготовка презентации для защиты дипломного проекта

4.5 Оформление пояснительной записки

**5 Состав презентации:**

5.1. Цели и задачи дипломного проекта

5.2. Описание предмета исследования

5.3. Описание разработки сайта

5.4. Описание самого сайта

**6 График выполнения дипломного проекта:**

|  |  |
| --- | --- |
| **Наименование раздела дипломного проекта** | **Срок выполнения** |
| Анализ задания на дипломное проектирование | 01.05.25 |
| Выбор программных и инструментальных средств | 15.05.25 |
| Анализ выбранной технологии разработки сайта | 19.05.25 |
| Разработка серверной части сайта | 25.05.25 |
| Разработка клиентской части сайта | 29.05.25 |
| Тестирование программного продукта | 01.06.25 |
| Написание и оформление пояснительной записки, прохождение нормоконтроля | 05.06.25 |
| Подготовка презентации для защиты дипломного проекта | 09.06.25 |

Срок выдачи: «24» марта 2025 г.

Срок сдачи студентом дипломного проекта: 23 июня 2025

Задание рассмотрено на заседании кафедры ИСиАУ (протокол № 7 от «21» марта 2025)

Заведующий кафедрой ИСиАУ / М.А. Гузачёв/

Руководитель дипломного проекта /И.А. Орабинский/

Задание по дипломному проектированию получил /Н.И. Георгиев

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc201225549)

[1 Анализ задач и выбор средств реализации 5](#_Toc201225550)

[1.1 Постановка задачи и определение требований к сайту 5](#_Toc201225551)

[1.2 Обзор и анализ существующих решений-аналогов 6](#_Toc201225552)

[1.3 Обоснование выбора технологического стека 7](#_Toc201225553)

[1.3.1 Технологии для разработки серверной части 7](#_Toc201225554)

[2 Проектирование и разработка сайта 11](#_Toc201225555)

[2.1 Проектирование архитектуры и базы данных 11](#_Toc201225556)

[2.2 Разработка серверной части 11](#_Toc201225557)

[2.3 Разработка клиентской части 11](#_Toc201225558)

[2.4 Взаимодействие серверной и клиентской части 11](#_Toc201225559)

[3 Тестирование и демонстрация работы 11](#_Toc201225560)

[3.1 Стратегии и методы тестирования 11](#_Toc201225561)

[3.2 Результат тестирования основного функционала 11](#_Toc201225562)

[3.3 Демонстрация работы сайта 11](#_Toc201225563)

[Заключение 12](#_Toc201225564)

[Список использованных источников 13](#_Toc201225565)

[Приложение А 14](#_Toc201225566)

[Приложение Б 15](#_Toc201225567)

ВВЕДЕНИЕ

Сегодня интернет стал неотъемлемой частью нашей жизни, изменив способы общения, работы и, конечно, торговли. Всё больше людей предпочитают делать покупки онлайн, и бизнес активно переходит в цифровую среду. Интернет-магазины удобны как для покупателей, которым не нужно выходить из дома, так и для продавцов, которые могут автоматизировать многие процессы и сократить расходы. Поэтому создание качественных и удобных сайтов для онлайн-торговли — это актуальная и востребованная задача.

Данный проект посвящен разработке интернет-магазина «ГеоMUSIC», специализирующегося на продаже музыкальных инструментов и оборудования.

Цель этого дипломного проекта — изучить современные методы и технологии создания сайтов.

Чтобы достичь этой цели, нужно выполнить следующие задачи:

1. Спланировать, как будет работать сайт, и продумать его структуру.
2. Спроектировать базу данных, где будет храниться вся информация о товарах, категориях и пользователях.
3. Написать серверную часть (backend), которая будет обрабатывать все запросы: от показа товаров до авторизации пользователей.
4. Создать клиентскую часть (frontend) — то, что видит пользователь: каталог товаров, корзину, страницу входа.
5. Сделать удобную панель для сотрудников магазина, чтобы они могли легко добавлять информацию.
6. Протестировать сайт, чтобы убедиться, что всё работает правильно и без ошибок.

Практическая значимость работы в том, что её результат — это готовый сайт интернет-магазина. Его можно использовать как основу для реального интернет-магазина, помогая бизнесу автоматизировать продажи и предложить клиентам удобный сервис для покупок.

1. Анализ задач и выбор средств реализации
   1. Постановка задачи и определение требований к сайту

Целью данного дипломного проекта является разработка и реализация сайта для интернет-магазина музыкальных инструментов "ГеоMusic". Проект направлен на создание современного, функционального и удобного для пользователя решения.

В рамках проекта необходимо решить следующие ключевые задачи:

1. Спроектировать и реализовать пользовательский интерфейс на основе предварительно разработанного макета.
2. Разработать структуру базы данных для хранения информации о товарах, пользователях и заказах.
3. Реализовать серверную часть (backend) для обработки бизнес-логики, включая управление каталогом товаров и аутентификацию пользователей.
4. Создать клиентскую часть (frontend), обеспечивающую интерактивное взаимодействие пользователя с сайтом.
5. Обеспечить безопасное хранение данных, таких как пароли пользователей и информация о товарах.

Функциональные требования для пользователей:

* + Просмотр каталога товаров;
  + Просмотр детальной информации о товаре;
  + Просмотр и редактирование корзины;
  + Система регистрации и авторизации пользователей.

Функциональные требования для администратора (сотрудника магазина):

* + Наличие защищенного интерфейса для управления информацией.

Нефункциональные требования:

* Архитектура: Приложение должно быть реализовано по принципу одностраничного приложения (Single Page Application, SPA), что обеспечивает плавную и быструю работу сайта без перезагрузки страниц.
* Безопасность: Конфиденциальные данные должны храниться в базе данных, доступ к функциям администрирования должен быть ограничен.
* Среда развертывания: В рамках дипломного проекта серверная и клиентская части будут развернуты на локальном сервере для демонстрации работоспособности.
  1. Обзор и анализ существующих решений-аналогов

Для определения лучших практик и выявления конкурентных преимуществ был проведен анализ существующих на рынке интернет-магазинов музыкальных инструментов. В качестве аналогов были рассмотрены такие платформы, как «Музторг».

Анализ проводился по следующим критериям:

* Структура каталога и удобство навигации.
* Функциональность карточки товара.
* Процесс оформления заказа и работа с корзиной.

По результатам анализа было выявлено, что ключевыми факторами успеха являются интуитивно понятный интерфейс, быстрая работа сайта и детальное описание товаров. Разрабатываемый проект "ГеоMUSIC" будет учитывать эти аспекты, делая акцент на минималистичном дизайне и высокой производительности за счет использования архитектуры SPA.

Для разработки сайта были выбраны следующие технологии: JavaScript, Node.js, Express.js, Sequelize, REST API, PostgreSQL, React.js, React-Bootstrap, Axios, React Router DOM, MobX.

* 1. Обоснование выбора технологического стека

Для разработки интернет-магазина «ГеоMUSIC» был выбран современный и гибкий технологический стек (набор технологий), полностью основанный на языке JavaScript. Данный подход позволяет использовать единый язык для серверной (backend) и клиентской (frontend) частей, что унифицирует процесс разработки и упрощает поддержку проекта. Выбранные технологии и инструменты можно разделить на две основные группы: средства для разработки серверной части и средства для разработки клиентской части.

* + 1. **Инструменты и технологии для разработки серверной части**

Серверная часть отвечает за бизнес-логику, обработку запросов от клиента, взаимодействие с базой данных и обеспечение безопасности.

1. JavaScript (JS) — это интерпретируемый язык программирования, который используют для написания frontend- и backend-частей сайтов, а также мобильных приложений. Он позволяет создавать интерактивные и динамические веб-страницы, добавлять на сайт функциональные элементы, такие как анимации, кнопки, всплывающие окна и другие. Если HTML — это "скелет" страницы, а CSS — её "одежда", то JS — это "мозги", которые оживляют всё: анимации, формы, чаты, всплывающие окна и даже сложные веб-приложения.

Плюсы:

* Работает во всех браузерах;
* Можно писать и frontend, и backend, и мобильное ПО;
* Прост в изучении;
* Огромное комьюнити и множество библиотек;
* Асинхронность (обработка нескольких запросто одновременно).

Минусы:

* Зависимость от браузера (разные браузеры выполняют код по-разному);
* Не безопасен (код на JS виден пользователю, а значит в нём нельзя хранить важную информацию, которую пользователю видеть не нужно).

1. Node.js — это среда выполнения JavaScript на стороне сервера, основанная на движке V8 (из Chrome). Она позволяет использовать JavaScript для создания быстрых и масштабируемых серверных приложений, что раньше было возможно только в браузере. Он идеален для стартапов, веб-API и реального времени, но не для тяжелых вычислений.

Плюсы:

* Один язык для frontend и backend разработки;
* Высокая скорость;
* Огромная экосистема (более 2-х миллионов готовых пакетов);
* Легко разрабатывать ПО для маленьких сервисов;
* Безопасность (все важные данные (пароли, ключи) хранятся на сервере).

Минусы:

* Плохо подходит для CPU-задач (плохо справляется с тяжелыми вычислениями);
* Ошибки могут появляться только при запуске (решается расширением TypeScript);
* Частые обновления, приходится часто за ними следить.

1. Express.js — это минималистичный и гибкий веб-фреймворк для Node.js, который упрощает создание серверов, API и веб-приложений.

Плюсы:

* Простота и минимализм (сервер создаётся в 5 строчек кода);
* Гибкость (можно добавить только нужные функции);
* Множество учебных материалов.

Минусы:

* Нет встроенной ORM (для этого в проекте и используется *Sequelize*);
* Не для CPU-задач.

1. Sequelize — это мощный ORM (Object-Relational Mapping) для Node.js, который позволяет работать с реляционными базами данных (PostgreSQL, MySQL, SQLite, MSSQL) через JavaScript-объекты вместо SQL-запросов.

Плюсы:

* Работа без SQL-запросов;
* Поддержка 6 различных СУБД.

Минусы:

* Нагружает систему;
* Для маленьких проектов легче использовать обычные SQL-запросы.

1. REST API (Representational State Transfer) — это архитектурный стиль для создания веб-сервисов, который использует стандартные HTTP-методы (GET, POST, PUT, DELETE) для взаимодействия с ресурсами.

Плюсы:

* Простота;
* Кэширование;
* Масштабируемость.

Минусы:

* Избыточность данных и ограниченная гибкость(если нужны только пару полей из всех, то сервер всё равно вернёт все поля).

1. PostgreSQL — это система управления базой данных (СУБД) с открытым исходным кодом, известная своей надежностью, расширяемостью и соответствием стандартам SQL.

Плюсы:

* Бесплатный;
* Надёжность;
* Масштабируемость;
* Безопасность.

Минусы:

* Сложность настройки;
* Требует больше памяти, чем MySQL или SQLite;
* Для простых проектов избыточен.
  + 1. **Технологии для разработки клиентской части**

Клиентская часть отвечает за пользовательский интерфейс (UI) и взаимодействие с пользователем в браузере.

1. React.js — это популярная библиотека с открытым исходным кодом, разработанная для создания интерактивных веб-интерфейсов и мобильных интерфейсов. Если нужна скорость и гибкость — он идеален.

Плюсы:

* Легко поддерживать и обновлять код;
* Высокая производительность;
* Работает с другими технологиями (Redux. GraphQL и т.д.).

Минусы:

* Сложен в освоении;
* Частые обновления;
* Только для пользовательского интерфейса (UI).

1. React-Bootstrap — это популярная библиотека компонентов для React. Она позволяет быстро создавать адаптивные интерфейсы без необходимости писать CSS-классы вручную.

Плюсы:

* Готовые компоненты;
* Прост в использовании.

Минусы:

* Сложно менять внутреннюю логику компонентов;
* Не для сложных UI.

1. Axios — это популярная библиотека для выполнения HTTP-запросов из браузера и Node.js. Она предоставляет простой API для работы с RESTful сервисами (REST - архитектурный стиль, основанный на HTTP-протоколе для разработки приложений) и поддерживает все современные возможности работы с сетью.

Плюсы:

* Автоматически преобразует ответ в JSON (поддерживает и другие форматы: XML, FormData и т.д.)
* Прост в использовании;
* Удобная и чёткая обработка ошибок.

Минусы:

* Избыточен для простых задач;
* Необходимо устанавливать в проект.

1. React Router DOM — это популярная библиотека для реализации маршрутизации в React-приложениях. Она превращает ваше SPA (Single Page Application) в приложение, которое ведёт себя как многостраничное, но без перезагрузки.

Плюсы:

* Не требуется перезагрузка страницы;
* Предзагрузка данных.

Минусы:

* Только для React;
* Сложен для новичков;
* Частые обновления.

1. MobX — это библиотека для управления состоянием в JavaScript-приложениях, которая используется для простого, реактивного и эффективного управления данными. Она автоматически отслеживает изменения данных и обновляет интерфейс без лишнего кода.

Плюсы:

* Изменил данные, а компоненты сами обновляются;
* Минимум кода;
* Производительность (не обновляет лишнего).

Минусы:

* Тяжело понять концепцию;
* Нет строгой структуры;
* Сложная отладка.

1. Проектирование и разработка сайта

После того как были определены цели, задачи и выбран технологический стек, начался основной этап работы — непосредственное проектирование и создание интернет-магазина «ГеоMUSIC». Этот процесс был разделен на несколько ключевых этапов: сначала была продумана общая архитектура и структура данных, затем последовательно разработаны серверная и клиентская части, и наконец, было настроено их взаимодействие.

* 1. Проектирование архитектуры и базы данных

На этом этапе были заложены основы будущего приложения. Правильное проектирование — залог того, что система будет стабильной, масштабируемой и простой в поддержке.

**Архитектура приложения**

Для проекта была выбрана классическая клиент-серверная архитектура. Она предполагает разделение приложения на две независимые части:

1. Серверная часть (Backend) — отвечает за всю логику, хранение данных и вычисления. Она не имеет собственного графического интерфейса и общается с пользователем через API.
2. Клиентская часть (Frontend) — это то, что видит и с чем взаимодействует пользователь в браузере. Она отвечает за отображение данных и отправку запросов на сервер.

Клиентская часть реализована как одностраничное приложение (Single Page Application, SPA). Это означает, что пользователь загружает сайт только один раз, а все последующие переходы между страницами (например, из каталога в корзину) происходят мгновенно, без перезагрузки, что делает интерфейс более отзывчивым и быстрым.

**Проектирование базы данных**

База данных — это "сердце" любого интернет-магазина. В ней хранится вся важная информация. Для проекта «ГеоMUSIC» была спроектирована реляционная база данных, управляемая СУБД **PostgreSQL**.

Основные сущности (таблицы) в базе данных:

* **Users** (Пользователи) — хранит данные о зарегистрированных пользователях: id, email, password (в захешированном виде) и role (роль, например, 'USER' или 'ADMIN').
* **Instruments** (Инструменты) — основная таблица с товарами. Содержит id, name (название), price (цена), img (название файла изображения) и info (информация).
* **Categories** (Категории) — справочник категорий товаров (id, name, img), например, "Гитары", "Барабаны".
* **Basket** (Корзина) — таблица, связывающая пользователя с его корзиной.

Связи между таблицами организованы с помощью внешних ключей (foreign keys), что обеспечивает целостность данных. Например, каждый товар в таблице Instruments обязательно связан с определенной категорией category).

*Подробная ER-диаграмма (схема связей) базы данных представлена в Приложении Б.*

* 1. Разработка серверной части

Серверная часть была разработана на платформе **Node.js** с использованием фреймворка **Express.js**. Весь код был структурирован по модульному принципу для удобства чтения и поддержки.

Ключевые реализованные модули:

1. **Маршрутизация (Routing):** С помощью express.Router были созданы группы маршрутов для каждой сущности:
   * /api/user — для регистрации и авторизации.
   * /api/instrument — для получения, добавления и редактирования товаров.
   * /api/category — для получения и добавления категорий.
   * /api/basket — для работы с корзиной пользователя.
2. **Контроллеры (Controllers):** Для каждого маршрута был написан контроллер — функция, которая содержит основную логику обработки запроса. Например, контроллер createInstrument принимает данные о новом товаре из запроса, проверяет их и с помощью Sequelize сохраняет в базу данных.
3. **Аутентификация и авторизация:** Безопасность — один из главных приоритетов.
   * **Регистрация и логин:** При регистрации пароль пользователя не хранится в открытом виде, а хешируется с помощью библиотеки bcrypt.
   * **JWT (JSON Web Token):** После успешной авторизации сервер генерирует для пользователя уникальный токен (JWT), который клиент сохраняет у себя. Этот токен отправляется с каждым последующим запросом, чтобы сервер мог понять, кто этот пользователь и какие у него права.
   * **Промежуточное ПО (Middleware):** Были созданы специальные функции-посредники. Например, authMiddleware проверяет наличие и валидность JWT-токена в каждом запросе. А checkRoleMiddleware проверяет роль пользователя (например, 'ADMIN') и закрывает доступ к определенным маршрутам (например, к добавлению товара) для обычных пользователей.
   1. Разработка клиентской части

Клиентская часть была создана с использованием библиотеки **React.js**. Весь интерфейс построен на **компонентном подходе**, где каждая часть страницы — это отдельный, независимый компонент.

Основные компоненты приложения:

* App.js: Корневой компонент, который управляет маршрутизацией.
* NavBar: Верхняя панель навигации с логотипом, ссылками и кнопкой входа.
* InstrumentList и InstrumenteItem: Компоненты для отображения списка товаров и отдельной карточки товара.
* CategoryList и CategoryItem: Компоненты для отображения списка категорий и отдельной карточки товара.
* Pages: Компоненты, отвечающие за отдельные "страницы" сайта (Shop, AdminPanel).
* Modals: Компоненты для всплывающих окон (например, для авторизации, корзины или добавления нового товара).

Для управления данными в приложении использовалась библиотека **MobX**. Были созданы так называемые "хранилища" (stores):

* UserStore: Хранит информацию об авторизованном пользователе и его роли.
* InstrumentStore: Хранит список товаров, категорий, а также информацию о выбранной категории.

Когда пользователь совершает какое-то действие (например, выбирает категорию), соответствующий компонент вызывает функцию в InstrumentStore, та обновляет данные, а MobX автоматически заставляет все "подписанные" на эти данные компоненты перерисоваться.

* 1. Взаимодействие серверной и клиентской части

Связью между независимыми Frontend и Backend частями служит **REST API**. Клиентская часть не имеет прямого доступа к базе данных. Вместо этого она отправляет HTTP-запросы на определенные адреса (эндпоинты) на сервере.

Для отправки запросов использовалась библиотека **Axios**. Рассмотрим типичный сценарий взаимодействия на примере добавления товара в корзину:

1. **Действие пользователя:** Пользователь нажимает кнопку "Добавить в корзину" на карточке товара.
2. **Frontend (React):** Срабатывает onClick обработчик, который вызывает метод addToBasket() в InstrumentStore.
3. **HTTP-запрос (Axios):** addToBasket() формирует и отправляет POST-запрос на сервер по адресу /api/basket. В теле запроса передается ID товара, а в заголовках — id пользователя для идентификации пользователя.
4. **Backend (Express):** Сервер получает запрос. authMiddleware проверяет токен. Если все в порядке, запрос передается в basketController.
5. **Логика и БД (Sequelize):** Контроллер использует модели Sequelize, чтобы добавить запись в таблицу Basket, связывая ID товара с ID корзины текущего пользователя.
6. **HTTP-ответ:** Сервер отправляет обратно ответ, например, со статусом 200 OK, подтверждая, что товар успешно добавлен.
7. **Обновление интерфейса:** Frontend (Axios) получает ответ. MobX обновляет состояние корзины, и React автоматически перерисовывает иконку корзины, показывая, что в ней появился новый товар.

Таким образом, четкое разделение на клиент и сервер, и их общение через API, позволило создать гибкую и надежную систему.

1. Тестирование и демонстрация работы

После завершения разработки ключевых модулей приложения был проведен этап тестирования. Его главная цель — убедиться, что все функции интернет-магазина «ГеоMUSIC» работают корректно, без сбоев, и соответствуют изначальным требованиям. Качественное тестирование помогает выявить и исправить ошибки до того, как с ними столкнутся реальные пользователи.

* 1. Стратегии и методы тестирования

Для проверки работоспособности сайта была выбрана стратегия **ручного тестирования**. Этот подход позволяет имитировать действия реального пользователя, проверять удобство интерфейса и выявлять не только функциональные ошибки, но и проблемы с логикой или дизайном.

Тестирование проводилось по методу **«черного ящика»**. Это означает, что тесты выполнялись без знания о внутренней структуре кода. Проверялось только то, соответствует ли поведение системы ожидаемому результату. Например, при нажатии на кнопку «Добавить в корзину» проверяется, что товар действительно появился в корзине, а не то, какая именно функция в коде за это отвечает.

Процесс тестирования был разделен на несколько видов:

1. **Функциональное тестирование:** Проверка того, что каждая функция сайта работает так, как задумано. Например, работает ли регистрация, добавляются ли товары, правильно ли считается сумма в корзине.
2. **Тестирование пользовательского интерфейса (UI):** Проверка внешнего вида сайта. Корректно ли отображаются все элементы (кнопки, картинки, текст), нет ли "поехавшей" верстки на разных экранах, удобна ли навигация.
3. **Тестирование совместимости:** Проверка работы сайта в разных веб-браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge), чтобы убедиться в кроссбраузерной поддержке.
4. **Тестирование безопасности:** Проверка базовых аспектов безопасности, например, невозможность доступа к панели администратора без авторизации.
   1. Результат тестирования основного функционала

Для систематизации процесса были разработаны **тест-кейсы** — сценарии, описывающие шаги для проверки конкретной функции и ожидаемый результат. Ниже приведены результаты тестирования ключевого функционала сайта.

**Тест-кейс 1: Регистрация и авторизация пользователя**

* **Шаги:**
  1. Открыть сайт. Нажать кнопку "Войти".
  2. Во всплывающем окне перейти на вкладку "Регистрация".
  3. Ввести валидные email и пароль. Нажать "Зарегистрироваться".
  4. Попробовать войти с новыми данными.
* **Ожидаемый результат:** Пользователь успешно регистрируется. После авторизации кнопка "Войти" меняется на "Личный кабинет" и "Выйти".
* **Фактический результат:** **Успешно.** Функция работает корректно.

**Тест-кейс 2: Просмотр и фильтрация товаров**

* **Шаги:**
  1. Открыть главную страницу (каталог).
  2. Выбрать одну из категорий (например «Гитары»)
* **Ожидаемый результат:** Открывается список гитар.
* **Фактический результат:** **Успешно.**

**Тест-кейс 3: Добавление товара в корзину**

* **Шаги:**
  1. Перейти на страницу карточки товара
  2. Нажать кнопку "Добавить в корзину".
  3. Открыть модальное окно корзины.
* **Ожидаемый результат:** Выбранный товар появляется в корзине с указанием цены. Итоговая сумма заказа корректно рассчитана.
* **Фактический результат:** **Успешно.** Товар добавляется, сумма обновляется.

**Тест-кейс 4: Доступ к панели администратора**

* **Шаги:**
  1. Авторизоваться под учетной записью с ролью 'USER'.
  2. Попробовать перейти по прямому URL-адресу /admin.
  3. Авторизоваться под учетной записью с ролью 'ADMIN'.
  4. Нажать кнопку "Панель администратора".
* **Ожидаемый результат:** Обычный пользователь не может попасть в админ-панель. Администратор получает доступ к функциям добавления товаров.
* **Фактический результат:** **Успешно.** Система ролей работает правильно, доступ разграничен.

**Итоги тестирования:** В ходе ручного тестирования были выявлены и устранены следующие ошибки:

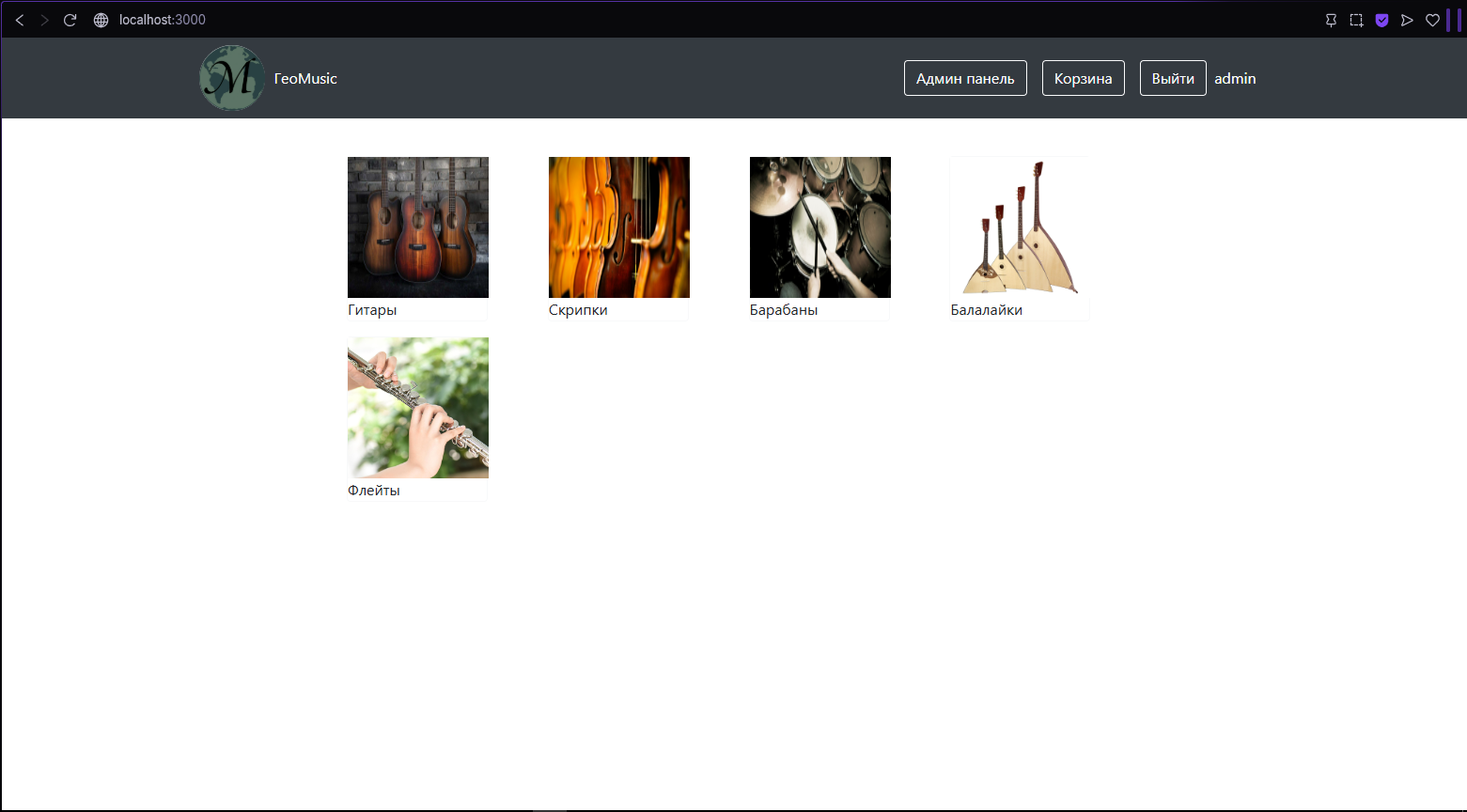
* При нажатии на кнопку «Выход» не работала функция выхода из учётной записи
* Страница инструмента с подробной информацией не загружала данные из таблицы
* Неправильный вывод инструментов по их id

Весь основной функционал интернет-магазина «ГеоMUSIC» работает стабильно и соответствует заявленным требованиям.

* 1. Демонстрация работы сайта

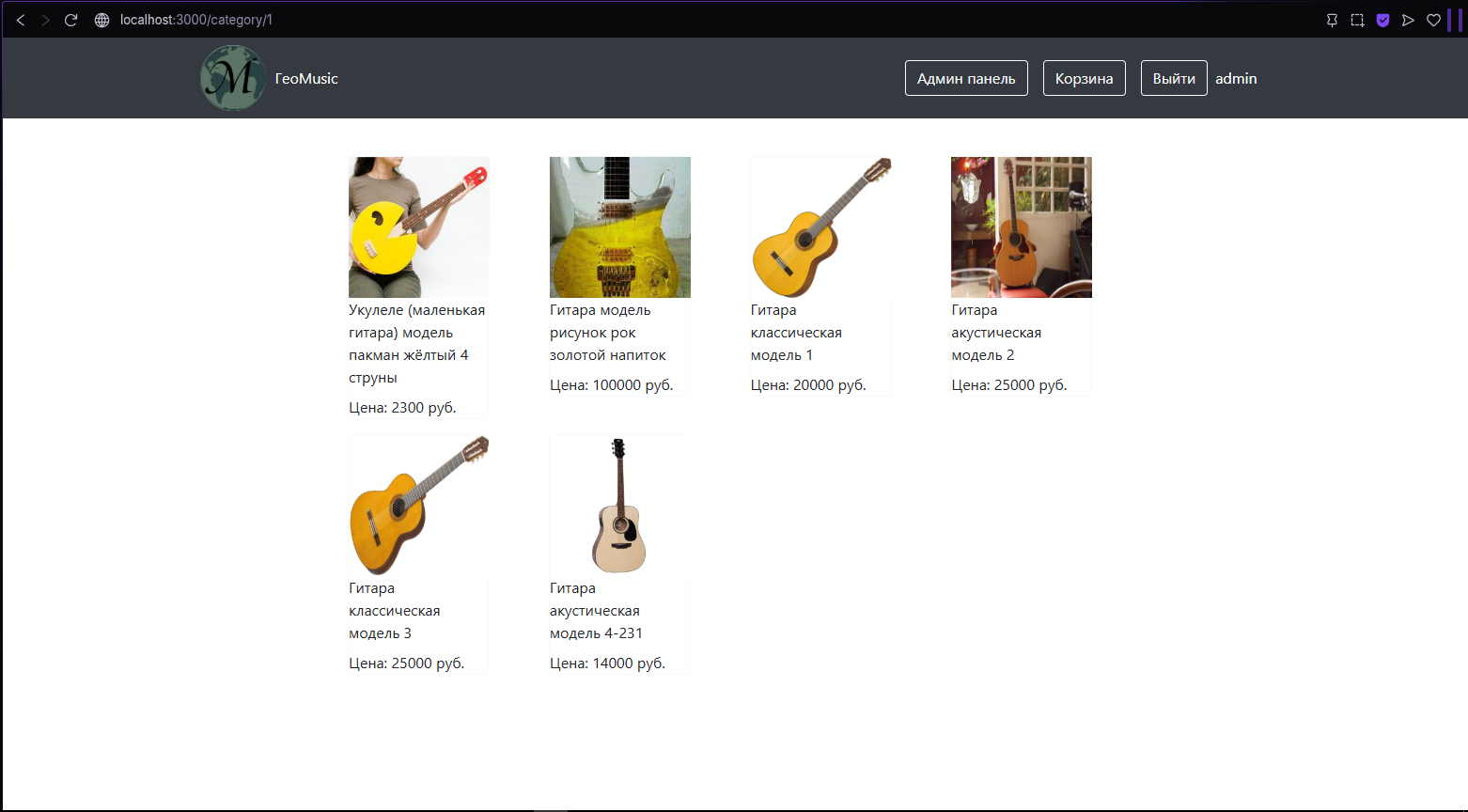
Для наглядной демонстрации работы разработанного приложения ниже представлены скриншоты ключевых страниц и интерфейсов.

**1. Главная страница (каталог товаров)**  
На главной странице отображаются категории товаров.

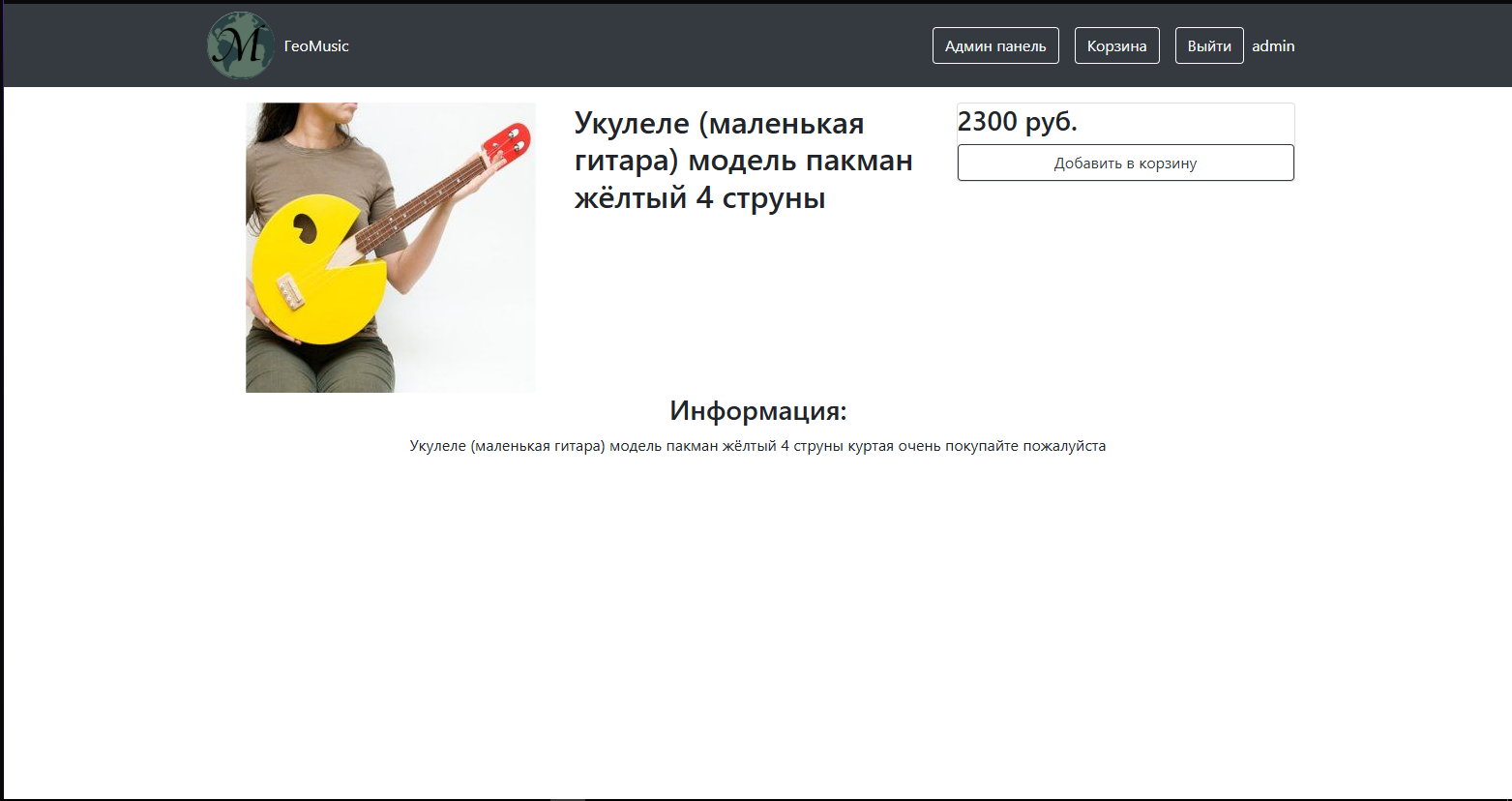
**

**2. Страница инструментов определённой категории**

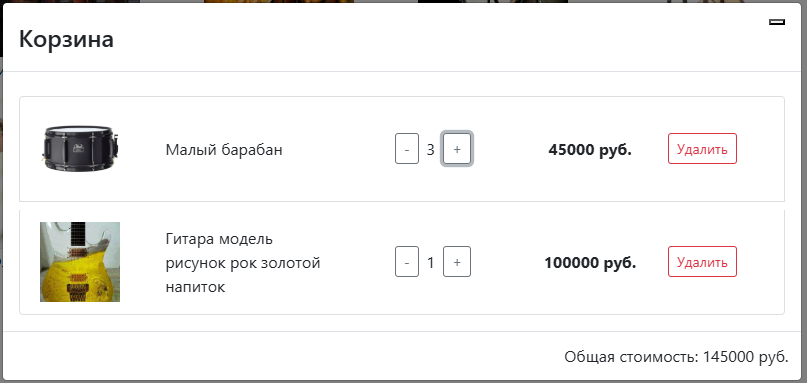
На странице инструментов отображаются товары и их цены



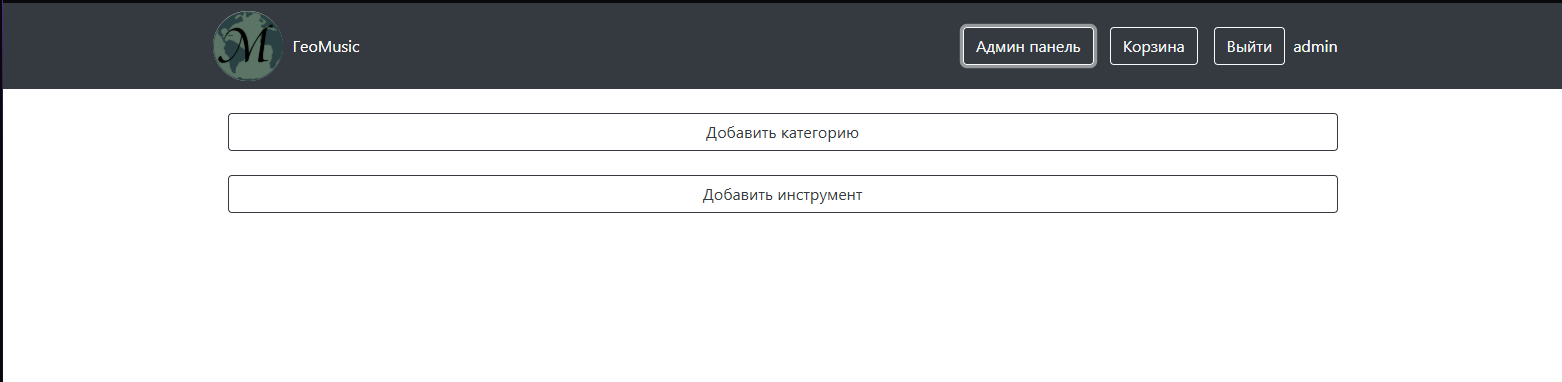
**3. Карточка товара и добавление в корзину**  
При клике на товар открывается страница с подробным описанием, характеристиками и кнопкой добавления в корзину.

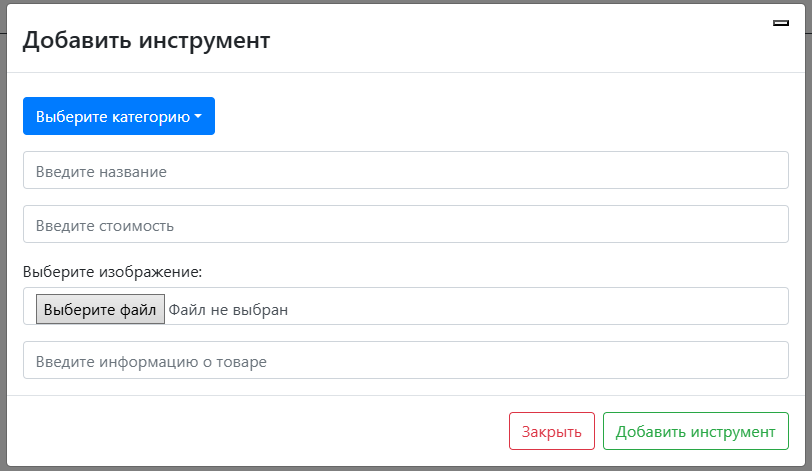
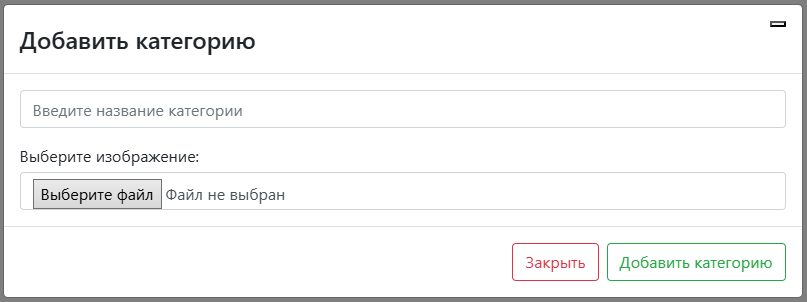


**4. Модальное окно корзины**  
После авторизации становится доступна кнопка «Корзина», при нажатии на которую открывается модальное окно на котором отображена информация о товарах в корзине и цене.

**

**5. Панель администратора**  
После авторизации с правами администратора становится доступна панель, где можно добавлять новые категории и товары через удобные модальные окна.

**

**

Представленные скриншоты иллюстрируют работоспособность основного функционала и демонстрируют готовый к использованию пользовательский интерфейс интернет-магазина «ГеоMUSIC».

Заключение

В рамках этого дипломного проекта был с нуля разработан интернет-магазин музыкальных инструментов «ГеоMUSIC». Цель работы — изучить современные технологии создания сайтов и создать современный, удобный сайт интернет-магазина — была успешно достигнута.

В ходе работы были выполнены все поставленные задачи:

* Сначала мы спланировали, как будет устроен сайт, и выбрали подходящие технологии: связку Node.js для сервера и React для клиентской части.
* Затем была спроектирована и создана база данных в PostgreSQL для хранения всей информации.
* Была написана серверная часть, которая обрабатывает все запросы, отвечает за безопасность (логин, пароли) и предоставляет данные для сайта.
* Была создана клиентская часть — то, что видит пользователь. Реализован каталог с фильтрами, корзина, а также удобная панель для администратора, где можно управлять товарами.
* Было проведено тестирование, которое показало, что все функции работают стабильно и без ошибок.

В итоге мы получили готовый продукт: полнофункциональный интернет-магазин. Он быстро работает благодаря технологии одностраничного приложения (SPA), имеет понятный интерфейс и обеспечивает разделение прав доступа для обычных пользователей и администраторов.

Эта работа имеет практическую пользу: созданное приложение можно использовать как основу для настоящего коммерческого проекта, чтобы помочь бизнесу выйти в онлайн.

В будущем проект можно развивать и дальше: добавить онлайн-оплату, систему отзывов и оценок от покупателей или даже создать мобильное приложение для еще большего удобства.

Таким образом, все задачи, поставленные в начале, были решены, а результат работы представляет собой готовый и полезный программный продукт.

Список использованных источников

Приложение А

(обязательное)

Программный код

Приложение Б

(обязательное)

СХЕмы, диаграммы

