Содержание

[Введение 3](#_Toc185804505)

[1.2 Конкурентная аналитика 4](#_Toc185804506)

[1.3 Бриф на разработку web-приложения 6](#_Toc185804507)

[2 Проектирование дизайна web-страниц 10](#_Toc185804508)

[2.1 Структура web-приложения 10](#_Toc185804509)

[2.2 Текстовый прототип 10](#_Toc185804510)

[2.3 Вайрфреймы web-страниц 12](#_Toc185804511)

[2.4 Дизайн-макеты web-страниц 12](#_Toc185804512)

[3 Реализация web-приложения 14](#_Toc185804513)

[3.1 Выбор технологий и инструментов 14](#_Toc185804514)

[3.2 Разработка слоя доступа к данным 15](#_Toc185804515)

[3.3 Разработка серверной части 19](#_Toc185804516)

[3.4 Разработка клиентской части 24](#_Toc185804517)

[4 Руководство системного программиста 33](#_Toc185804518)

[5 Описание текстового примера 34](#_Toc185804519)

[Заключение 41](#_Toc185804520)

[Список используемых источников 42](#_Toc185804521)

# Введение

С развитием автомобильной промышленности и увеличением количества автомобилей на дорогах потребность в качественных и доступных запчастях становится всё более значимой. Это обуславливает рост спроса на удобные инструменты и платформы, которые позволяют эффективно искать и заказывать автозапчасти. Однако современные решения зачастую имеют ограниченный функционал или недостаточную систематизацию данных, что усложняет их использование.

Создание веб-приложения, ориентированного на предоставление простого и быстрого доступа к информации о запчастях, становится важной задачей в условиях растущего рынка. Цель данной работы заключается в разработке инструмента, который объединит удобство поиска, структурированность данных и необходимые функциональные возможности для пользователей различных категорий.

Для достижения этой цели в рамках проекта необходимо выполнить ряд задач:

* провести анализ существующих решений, выявив их сильные и слабые

стороны, чтобы определить основные требования к разрабатываемой системе.

* спроектировать архитектуру, обеспечивающую надёжное хранение и

оперативную обработку данных.

* разработать серверную часть для взаимодействия с базой данных и

предоставления информации по запросам пользователей.

* Создать интерфейс, обеспечивающий удобное взаимодействие.
* протестировать систему на различных сценариях использования, чтобы

гарантировать её стабильную и корректную работу.

Ожидаемым результатом станет разработка приложения, способной удовлетворить запросы целевой аудитории. Платформа будет ориентирована на улучшение пользовательского опыта за счёт качественного поиска данных и возможности их систематизации.

Ключевой бизнес-целью является создание конкурентоспособного решения, которое привлечёт и удержит пользователей благодаря удобству использования, надёжности работы и адаптации к их потребностям. Это особенно важно для автомобильного рынка, где скорость поиска и достоверность информации играют решающую роль.

Целевая аудитория платформы охватывает широкий круг пользователей, включая:

* владельцев автомобилей, которым необходим доступ к актуальным

запчастям для ремонта и обслуживания.

* профессионалов, работающих в сфере технического обслуживания и

ремонта транспортных средств.

* автолюбителей, которые ищут надёжные инструменты для поиска

деталей.

Проект направлен на создание универсального решения, которое упростит взаимодействие пользователей с рынком автозапчастей, обеспечит их доступ к необходимой информации и сделает процесс поиска и заказа более комфортным. Такой подход позволит платформе занять достойное место на рынке и удовлетворить потребности широкой аудитории.

## 1.2 Конкурентная аналитика

Для анализа будет рассмотрено три web-приложения:

− Autodoc, https://www.autodoc.ru;

− Exist.ru, https://www.exist.ru;

− Emex, https://www.emex.ru.

В рассматриваемых web-приложения будут проанализированы следующие характеристики: эргономичность дизайна; разделы и структура, функционал, подача информации. Для оценки характеристик будет использоваться 5-бальная шкала, где 1 – функция отсутствует, а 5 – функция есть. Сравнение характеристик рассматриваемых web приложений представлено в таблице 1.1.

Таблица 1.1 – Сравнение характеристик рассматриваемых web-сайтов

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики | Web-приложения | | |
| Autodoc | Exist.ru | Emex |
| Эргономичность дизайна | 5 | 4 (удобный, но немного перегруженный интерфейс с множеством рекламных блоков и мелких шрифтов.) | 3 (дизайн выглядит устаревшим, перегружен рекламой и мелким текстом, что затрудняет использование.) |
| Разделы и структура | 5 | 4(Структура сайта ясная, но из-за большого количества разделов иногда тяжело найти нужную информацию.) | 4 (структура сайта ясная, но обилие рекламы может отвлекать от поиска нужных товаров.) |
| Функционал | 5 | 4 (возможности создания списков избранного ограничены) | 3(Отсутствуют удобные функции, такие как создание списка избранного, возможности фильтрации ограничены.) |
| Подача информации | 5 | 5 | 3 (недостаточное количество информации) |

Оптимальным сайтом является Autodoc, так как он имеет отличный эргономичный дизайн, продвинутое функциональное обеспечение и отличную подачу информации. Главным недостатком является наличие рекламы, но она не мешает основной функциональности сайта.

## 1.3 Бриф на разработку web-приложения

Бриф на разработку web-приложения представлен в таблице 1.2.

Таблица 1.2 – Бриф на разработку

|  |  |
| --- | --- |
| Тема проекта | Приложение для заказа и просмотра совместимости автозапчастей. |
| Категория системы | Web-приложение |
| Автоматизируемая деятельность | Создание персонализированных списков избранных запчастей и совместимости деталей |
| Заинтересованные стороны | Механики, автолюбители. |
| Текущее решение | Различные самостоятельные попытки сохранять информацию о запчастях и их характеристиках, а также их совместимость. |
| Решаемые проблемы | – Отсутствие удобного способа вести списки избранных запчастей.;  – Сложность нахождения хорошо структурированной информации о запчастях. |
| Цель для владельца проекта | – увеличение числа активных пользователей за счет привлекательного и удобного интерфейса;  – повышение уровня удержания пользователей благодаря возможности сохранения избранного и совместимости запчастей; |
| Назначение | – предоставить удобный способ сохранения списков избранных и совместимости деталей;  – предоставить хорошо структурированную информацию о запчастей. |

Продолжение таблицы 1.2

|  |  |
| --- | --- |
| Количество видов пользователей (ролей) | 1. Зарегистрированный пользователь  2. Незарегистрированный пользователь  3. Оператор склада |
| Ключевые свойства и возможности | 1. Поиск и фильтрация деталей  2. Сортировка найденных деталей  3. Заказ запчастей  4. Отмена заказа  5. Просмотр заказов  6. Возможность скачать накладную  7. Отзывы  8. Функция "Избранное"  9.Информация о совместимости деталей  11. Управление запчастями  12. Возможность фильтрации поставщиков  13. Информация о наличии  14. Управление Инвентарём  15.Выбор способа доставки |
| Количество страниц | 1. Авторизация  2. Регистрация  3. Каталог деталей  4. Профиль  5. Мои заказы и отзывы  6. Корзина  7. Избранные  8. Поставщики  9. Запчасти  10. Заказы |
| Количество отчетов | 1. Детали  2. Поставщики  3. Инвентарь |
| Системы-аналоги | 1.Autodoc,  2.Exist.ru  3.Emex |

Подробное описание реализуемых функций и сопоставление с web страницами представлено в таблице 1.3.

Таблица 1.3 – Функции приложения

|  |  |  |
| --- | --- | --- |
| Функция | Описание функции | Web-страница |
| 1 | 2 | 3 |
| Поиск и фильтрация деталей | Пользователь может вводить название деталей для получения списка доступных | Каталог деталей |
| Сортировка найденных деталей | Найденные детали можно отсортировать деталей по цене или популярности | Каталог деталей |
| Заказ запчастей | Возможность заказать деталь с указанием количества и способа доставки | Корзина |
| Просмотр заказов | Функция просмотра истории заказов  запчастей пользователем | Мои заказы и отзывы |
| Возможность скачать накладную | Функция предусматривает возможность скачать накладную в формате PDF | Поставщики, Запчасти |
| Отзывы | Возможность добавить отзыв на деталь | Мои заказы и отзывы |
| Функция "Избранное" | Пользователи могут добавлять интересные детали в избранное для быстрого доступа | Избранные |

Продолжение таблицы 1.3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Информация о совместимости деталей | Предоставление информации о совместимости деталей автомобилей | Каталог деталей |
| Управление деталями | Возможность оператору добавить новую запасами деталь, изменить или удалить старые | Запчасти |
| Возможность фильтрации поставщиков | Возможность отсортировать найденные фильтрации детали по поставщикам | Каталог деталей |
| Информация о наличии | Предоставление  информации о наличии запчастей на складе | Каталог деталей |
| Отмена заказа | Предоставить возможность отменить заказ | Заказы |
| Управление Инвентарём | Оператор может указывать местоположение запчасти и поставщика | Инвентарь |
| Выбор способа доставки | При заказе товара пользователь может выбрать способ доставке | Корзина |

# 2 Проектирование дизайна web-страниц

## 2.1 Структура web-приложения

Структура web-приложения представлена на рисунке 2.1.

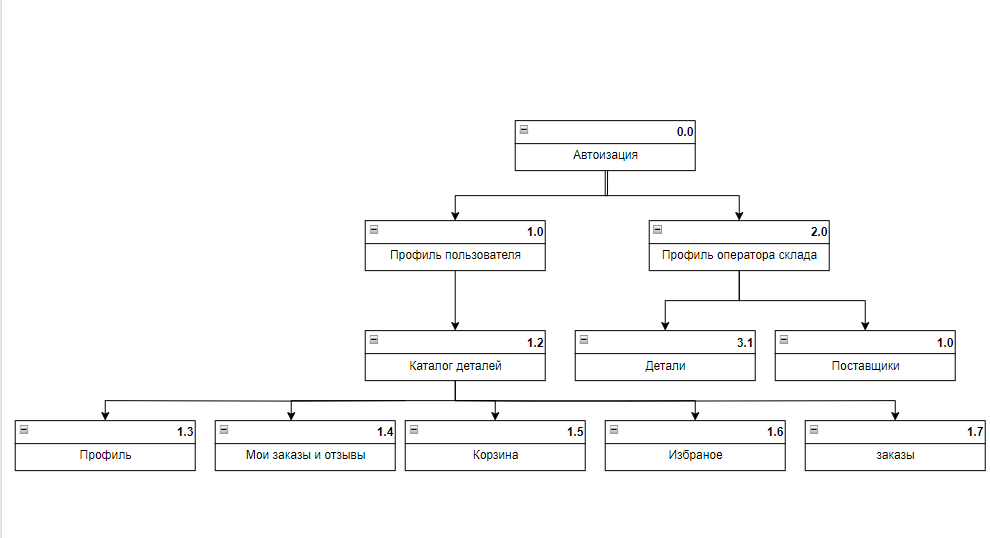


Рисунок 2.1 – Структура web-приложения

## 2.2 Текстовый прототип

Базовая структура сайта: **Авторизация, Каталог деталей, Профиль, Мои заказы и отзывы, Корзина, Избранные, Поставщики, Запчасти, Заказы.**

**2.2.1 Авторизация страница (ссылочный номер 0.0).** Желаемая структура **Авторизация** страницы:

1 Шапка с логотипом, кнопками быстрого перемещения.

2 Основная секция с полем для ввода и кнопкой войти

3 Футер с контактными данными.

**2.2.2 Каталог деталей страница (ссылочный номер 1.2).** Желаемая структура Каталог деталей страницы:

1 Шапка с логотипом, кнопками быстрого перемещения, и кнопкой выйти

2 Основная секция с полем для ввода, меню сортировки, списком деталей

3 Футер с контактными данными.

**2.2.3 Профиль страница (ссылочный номер 1.3).** Желаемая структура Профиль страницы:

1 Шапка с логотипом, кнопками быстрого перемещения, и с кнопкой выйти.

2 Основная секция с полем для изменения данных.

3 Футер с контактными данными.

**2.2.4** **Мои заказы и отзывы страница (ссылочный номер 1.4).** Желаемая структура Мои заказы и отзывы страницы:

1 Шапка с логотипом, кнопками быстрого перемещения, и с кнопкой выхода.

2 Основная секция, по середине должна располагаться информация о заказах и отзывов пользователя.

3 Футер с контактными данными.

**2.2.5 Корзина страница (ссылочный номер 1.5).** Желаемая структура Корзина страницы:

1 Шапка с логотипом, кнопками быстрого перемещения, и с кнопкой выхода.

2 Основная секция, по середине располагаются детали добавленные пользователем.

3 Футер с контактными данными.

**2.2.6 Избранные страница (ссылочный номер 1.6).** 1 Шапка с логотипом, кнопками быстрого перемещения, и с кнопкой выхода.

2 Основная секция, список с избранными деталями.

3 Футер с контактными данными.

**2.2.7 Заказы (ссылочный номер 2.1).** Желаемая структура Favorites страницы:

1 Шапка с логотипом, кнопками быстрого перемещения, и с кнопкой выхода.

2 Основная секция, с заказами и их состоянием.

**2.2.8 Поставщики (ссылочный номер 2.2).** Желаемая структура Favorites страницы:

1 Шапка с логотипом, кнопками быстрого перемещения, и с кнопкой выхода.

2 Основная секция, с поставщиками и возможность их добавлять

3 Футер с контактными данными.

## 2.3 Вайрфреймы web-страниц

Варфреймы всех страниц представлены в графической части.

На рисунке 2.2 представлена карта маршрутав пользователя.

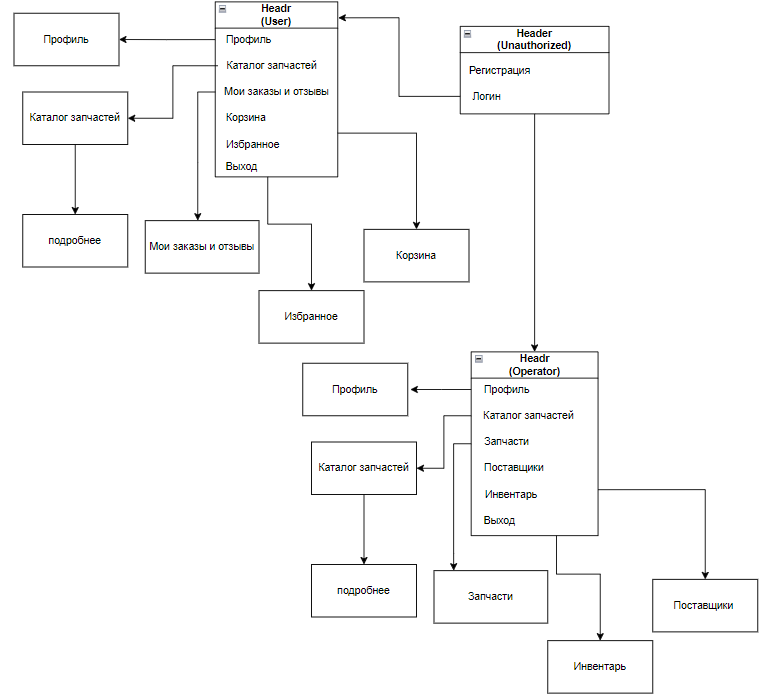


Рисунок 2.2 – Карта маршрутов пользователя

## 2.4 Дизайн-макеты web-страниц

Каталог деталей страница — страница, на которой отображаются все доступные детали в каталоге. В верхней части страницы расположен хедер с логотипом, кнопками быстрого перемещения и кнопкой "Выйти". Основная секция содержит поле для ввода, меню сортировки, фильтры по категории и цене, а также список всех доступных деталей. Внизу страницы размещен футер с контактными данными.

Профиль страница — страница профиля пользователя, где можно изменить личную информацию. В верхней части страницы расположен хедер с логотипом, кнопками быстрого перемещения и кнопкой "Выйти". Основная секция включает поля для изменения личных данных пользователя. Внизу страницы находится футер с контактными данными.

Мои заказы и отзывы страница — страница, на которой отображаются все заказы и отзывы пользователя. В верхней части страницы расположен хедер с логотипом, кнопками быстрого перемещения и кнопкой "Выйти". Основная секция содержит информацию о заказах и отзывах пользователя, расположенную по центру страницы. Внизу страницы находится футер с контактными данными.

Корзина страница — страница корзины, где отображаются добавленные в корзину товары. В верхней части страницы расположен хедер с логотипом, кнопками быстрого перемещения и кнопкой "Выйти". Основная секция содержит перечень добавленных пользователем товаров с изображениями, названиями, количеством и общей ценой. Внизу страницы размещен футер с контактными данными.

Избранные страница — страница избранных товаров пользователя. В верхней части страницы расположен хедер с логотипом, кнопками быстрого перемещения и кнопкой "Выйти". Основная секция включает список избранных товаров. Внизу страницы размещен футер с контактными данными.

Заказы страница — страница заказов, на которой отображаются все активные и завершенные заказы пользователя. В верхней части страницы расположен хедер с логотипом, кнопками быстрого перемещения и кнопкой "Выйти". Основная секция отображает все заказы пользователя и их состояние (активные или завершенные). Внизу страницы размещен футер с контактными данными.

Поставщики страница — страница поставщиков, на которой можно просматривать и добавлять новых поставщиков. В верхней части страницы расположен хедер с логотипом, кнопками быстрого перемещения и кнопкой "Выйти". Основная секция содержит перечень поставщиков и возможность добавления новых. Внизу страницы размещен футер с контактными данными.

Регистрация страница — страница регистрации нового пользователя. Основная секция содержит форму с полями для ввода имени, фамилии, email, пароля и кнопку "Зарегистрироваться". Внизу страницы имеется ссылка на страницу авторизации для уже зарегистрированных пользователей.

Авторизация страница — страница авторизации, где пользователи могут войти в систему. Основная секция содержит форму для ввода email, пароля и кнопку "Войти". Внизу страницы доступна ссылка на страницу регистрации для новых пользователей.

# 3 Реализация web-приложения

## 3.1 Выбор технологий и инструментов

Для разработки веб-приложения «AutoPart» были выбраны современные технологии и инструменты, которые обеспечивают удобство разработки, надежность и масштабируемость приложения.

Клиентская часть разработана с использованием следующих технологий:

* React – библиотека для создания пользовательских интерфейсов, она позволяет создавать быстрые, интерактивные и адаптивные интерфейсы благодаря своей компонентной структуре. Использование React упрощает управление состоянием и обновление пользовательского интерфейса;
* React Router – библиотека для маршрутизации внутри приложения, которая позволяет создавать удобную навигацию и многопотоковую архитектуру страниц. React Router обеспечивает плавный переход между разделами приложения, улучшая пользовательский опыт;
* Redux Toolkit – менеджер состояний. Это решение позволяет централизовать управление состоянием приложения, делая его устойчивым и прогнозируемым, что особенно важно для приложения с большим объемом данных;
* Bootstrap – опулярный CSS-фреймворк, который облегчает создание адаптивных и мобильных интерфейсов. Включает в себя готовые компоненты и сеточную систему.
* Axios – API для взаимодействия с сервером. Она обеспечивает высокую производительность и упрощает процесс конфигурации запросов к серверу;
* Xlsx – Библиотека для работы с файлами Excel (XLSX). Позволяет читать и записывать данные в формате Excel, что полезно для обработки таблиц и отчетов.
* Webpack – это мощный инструмент для сборки модулей JavaScript, который используется для управления зависимостями и оптимизации ресурсов в веб-приложениях. Sass – препроцессор, использующийся для стилизации компонентов, он расширяет возможности CSS и делает код стилей более структурированным.

Для серверной части веб-приложения были выбраны следующие технологии:

* jsonwebtoken: Библиотека для работы с JSON Web Token (JWT) для создания и проверки токенов.

Для взаимодействия с базой данных и управления данными использовались следующие инструменты:

* PostgreSQL – СУБД. Она поддерживает ACID-транзакции и обеспечивает стабильную работу с большими объемами данных. PostgreSQL также предоставляет возможности для работы с JSON, что позволяет гибко хранить данные;
* Sequelize – ORM. Sequelize обеспечивает удобное и безопасное взаимодействие с базой данных, облегчает написание запросов и поддерживает миграции данных;
* pg и pg-hstore – библиотеки, использующиеся для подключения и обработки данных в PostgreSQL, что упрощает реализацию и интеграцию различных данных в ORM.

## 3.2 Разработка слоя доступа к данным

Структура базы данных представлена в графической части на листе 1.

Структуры таблиц базы данных представлены в таблицах 3.1 – 3.9.

Таблица 3.1 – Структура таблицы users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование атрибута | Хранимая информация | Тип данных | NOT NULL | Тип |
| id | идентификатор пользователя | UUID | true | primary\_key |
| username | имя пользователя | string | true |  |
| email | email | string | true | unique\_constraint |
| password | пароль | string | true |  |
| role | роль | ENUM |  |  |

Таблица 3.2 – Структура таблицы part

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование атрибута | Хранимая информация | Тип данных | NOT NULL | Тип |
| id | идентификатор фильма | UUID | true | primary\_key |
| name | Название детали | string | true |  |
| description | описание | text | false |  |
| price | цена | decimal | true |  |
| compatibility | совместимость | json | false |  |
| image | название изображения | string | false |  |
| stock | склад | integer | true |  |

Таблица 3.3 – Структура таблицы Supplier

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование атрибута | Хранимая информация | Тип данных | NOT NULL | Тип |
| id | идентификатор поставщиков | UUID | true | primary\_key |
| name | Название поставщика | string | true |  |
| contact\_info | Контактная информация | string | false |  |
| rating | Рейтинг поставщика | Decimal  (3, 2) | false |  |

Таблица 3.4 – Структура таблицы Order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование атрибута | Хранимая информация | Тип данных | NOT NULL | Тип |
| id | идентификатор заказа | UUID | true | primary\_key |
| status | Статус заказа | ENUM('pending', 'completed', 'cancelled') | false |  |
| total\_price | Общая цена заказа | Decimal  (10, 2) | false |  |
| delivery\_method | Метод доставки | string | false |  |
| address | Адрес доставки | string | true |  |

Таблица 3.5 – Структура таблицы OrderItem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование атрибута | Хранимая информация | Тип данных | NOT NULL | Тип |
| id | идентификатор | UUID | true | primary\_key |
| quantity | количество | integer | true |  |
| price | цена | Decimal  (10, 2) | true |  |

Таблица 3.6 – Структура таблицы Review

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование атрибута | Хранимая информация | Тип данных | NOT NULL | Тип |
| id | идентификатор | UUID | true | primary\_key |
| rating | рейтинг | integer | true |  |
| comment | комментарий | text | false |  |

Таблица 3.7 – Структура таблицы Favorite

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование атрибута | Хранимая информация | Тип данных | NOT NULL | Тип |
| id | идентификатор | UUID | true | primary\_key |

Таблица 3.8 – Структура таблицы Inventory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование атрибута | Хранимая информация | Тип данных | NOT NULL | Тип |
| id | идентификатор | UUID | true | primary\_key |
| quantity | количество | integer | true |  |
| location | местоположение | string | true |  |

## 3.3 Разработка серверной части

Структура серверной части приложения представлена на рисунке 3.2.

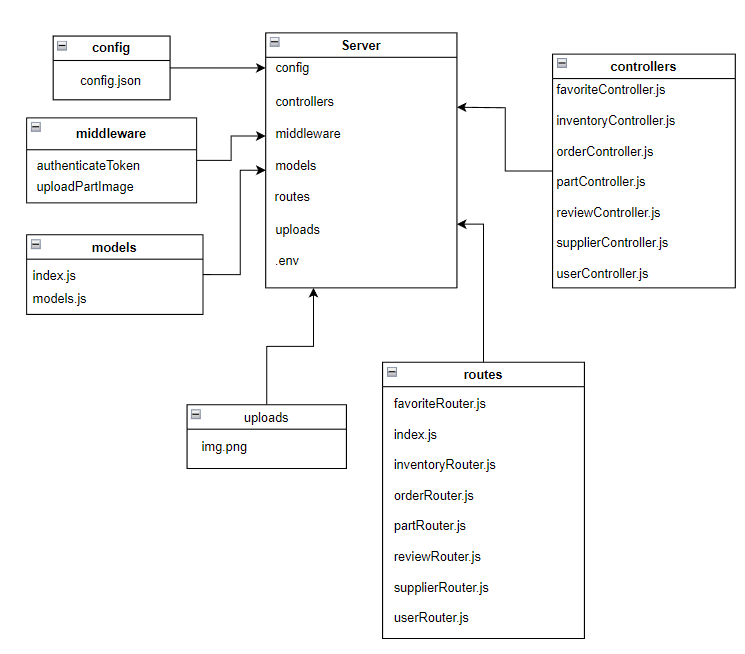


Рисунок 3.2 – Структура серверной части

Ниже описаны обработчики конечных точек сервера.

POST /api/suppliers

Добавляет нового поставщика. Принимает параметры: name (название), rating (рейтинг от 0 до 5), contact\_info (контактная информация).

GET /api/suppliers

Возвращает список всех поставщиков. Может принимать дополнительные параметры: page — текущая страница, limit — количество элементов на странице, sortBy — критерий сортировки, sortOrder — порядок сортировки, search — поисковый запрос.

GET /api/suppliers/$id

Возвращает информацию о поставщике по заданному id. Если поставщик с таким id не найден, возвращает 404: Not Found.

PUT /api/suppliers/$id

Обновляет информацию о поставщике по id. Принимает параметры: name — название поставщика, rating — рейтинг, contact\_info — контактная информация. Если поставщик с таким id не найден, возвращает 404: Not Found.

POST /api/suppliers/export/excel

Экспортирует список поставщиков в формате Excel.

POST /api/suppliers/export/docx

Экспортирует список поставщиков в формате DOCX.

POST /api/suppliers/contact\_info

Добавляет новое поле контактной информации для поставщика. Принимает параметры: key — ключ, value — значение.

DELETE /api/suppliers/contact\_info/$key

Удаляет контактную информацию для заданного key у поставщика. Если контактная информация с таким key не найдена, возвращает 404: Not Found.

POST /api/inventories

Создает новый инвентарь. Требует аутентификацию через токен. Принимает данные для нового инвентаря.

GET /api/inventories

Обработчик возвращает список всех инвентарей, доступных для текущего пользователя. Для выполнения этого запроса также требуется аутентификация с использованием токена.

GET /api/inventories/:id

Обработчик возвращает подробную информацию о конкретном инвентаре по его id. Этот запрос требует аутентификации с использованием токена. Если инвентарь с указанным id не найден, возвращается HTTP-ответ 404: Not Found.

PUT /api/inventories/:id

Обработчик обновляет данные инвентаря по заданному id. Для выполнения запроса требуется аутентификация с использованием токена. В теле запроса передаются новые данные инвентаря, такие как количество товара, описание или другие параметры. Если инвентарь с указанным id не существует, возвращается HTTP-ответ 404: Not Found. После успешного обновления инвентаря возвращается обновленная информация о нем.

DELETE /api/inventories/:id

Обработчик удаляет инвентарь по заданному id. Для выполнения этого запроса требуется аутентификация с использованием токена. Если инвентарь с указанным id не существует, возвращается HTTP-ответ 404: Not Found. После успешного удаления инвентаря возвращается подтверждение удаления или пустой ответ с кодом 200: OK.

POST /api/orders

Обработчик создает новый заказ. Для выполнения запроса требуется аутентификация с использованием токена

GET /api/orders

Обработчик возвращает список всех заказов пользователя. Для выполнения запроса требуется аутентификация с использованием токена.

GET /api/orders/:id

Обработчик возвращает информацию о заказе по заданному id. Для выполнения запроса требуется аутентификация с использованием токена. Если заказ с указанным id не найден, возвращается HTTP-ответ 404: Not Found.

PUT /api/orders/:id/status

Обработчик обновляет статус заказа по заданному id. Для выполнения запроса требуется аутентификация с использованием токена. В теле запроса передается новый статус заказа. Если заказ с указанным id не найден, возвращается HTTP-ответ 404: Not Found. После успешного обновления возвращает обновленную информацию о заказе.

PUT /api/orders/:id/cancel

Обработчик отменяет заказ по заданному id. Для выполнения запроса требуется аутентификация с использованием токена. Если заказ с указанным id не найден или не может быть отменен, возвращается HTTP-ответ 404: Not Found. После успешной отмены заказа возвращается обновленная информация о заказе с пометкой об отмене.

POST /api/parts

Обработчик создает новую запчасть. Для выполнения запроса требуется аутентификация с использованием токена

GET /api/parts

Обработчик возвращает список всех доступных запчастей. Запрос не требует аутентификации, так как доступен для всех пользователей.

PUT /api/parts/:id

Обработчик обновляет информацию о запчасти по заданному id. Для выполнения запроса требуется аутентификация с использованием токена. В теле запроса передаются обновленные данные для запчасти. Если запчасть с таким id не существует, возвращается HTTP-ответ 404: Not Found.

DELETE /api/parts/:id

Обработчик удаляет запчасть по заданному id. Для выполнения запроса требуется аутентификация с использованием токена. Если запчасть с таким id не существует, возвращается HTTP-ответ 404: Not Found. После успешного удаления возвращается подтверждение удаления или пустой ответ с кодом 200: OK.

POST /api/parts/:id/image

Обработчик загружает изображение для запчасти. Для выполнения запроса требуется аутентификация с использованием токена Загружаемое изображение прикрепляется к запчасти с заданным id. Если загрузка изображения не удалась, возвращается HTTP-ответ 400: Bad Request. После успешной загрузки изображения возвращается информация о запчасти с привязанным изображением.

POST /api/reviews

Обработчик создает новый отзыв. Для выполнения запроса требуется аутентификация с использованием токена

GET /api/reviews/:id

Обработчик возвращает отзыв по заданному id. Этот запрос требует аутентификацию с использованием токена. Если отзыв с таким id не найден, возвращается HTTP-ответ 404: Not Found.

DELETE /api/reviews/:id

Обработчик удаляет отзыв по заданному id. Для выполнения запроса требуется аутентификация с использованием токена.Если отзыв с таким id не существует, возвращается HTTP-ответ 404: Not Found.

POST /api/users/register

Обработчик регистрирует нового пользователя. Принимает в теле запроса данные, такие как имя, электронная почта и пароль.

POST /api/users/login

Обработчик выполняет вход пользователя. Принимает в теле запроса электронную почту и пароль, а затем возвращает jwt токен, который может быть использован для аутентификации при дальнейшем взаимодействии с сервером. Если данные некорректны, возвращает 404: Not Found или 401: Unauthorized.

## 3.4 Разработка клиентской части

Структура клиентской части приложения представлена на рисунке 3.3.

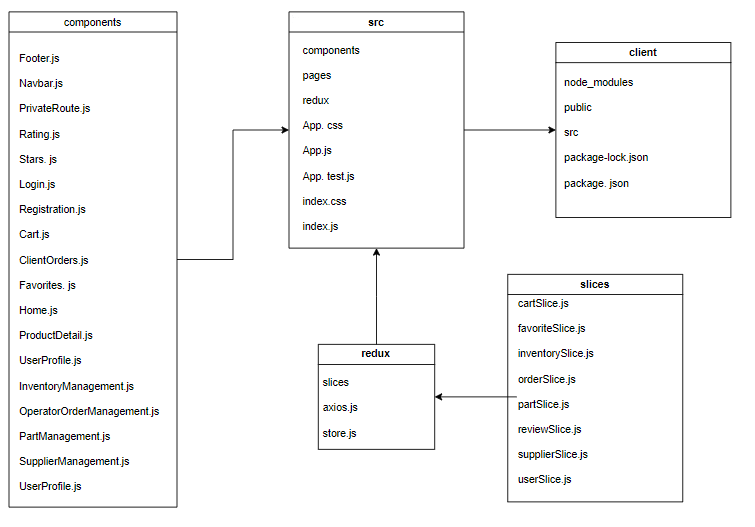


Рисунок 3.3 – Структура клиентской части

Внутри проекта идёт разделение на следующие пакеты:

* client – Этот раздел содержит корневую структуру проекта.;
* redux – Содержит файл axios.js, в котором настроен клиент для работы с API, включая базовый URL и обработку запросов.components – содержит компоненты и папку slice
* slice – Этот раздел содержит все slice
* src – Основная папка с исходным кодом приложения;
* components - Этот раздел содержит все React-компоненты, которые используются в приложении.;

В таблице 3.10 приведены разработанные компоненты.

Таблица 3.10 – Компоненты

|  |  |
| --- | --- |
| Компонент | Описание |
| Footer | Компонент футера, отображающий контактные данные и дополнительные ссылки, такие как политика конфиденциальности, условия использования и контактная информация. Обычно размещается внизу страницы и присутствует на всех основных страницах сайта. |
| Navbar | Компонент навигационной панели, содержащий логотип, ссылки на основные разделы сайта и кнопки для входа/выхода, регистрации и личного кабинета. Обеспечивает переход между страницами и улучшает пользовательский интерфейс. |
| PrivateRoute | Компонент для защиты маршрутов. Проверяет, аутентифицирован ли пользователь. Если пользователь не авторизован, компонент перенаправляет его на страницу авторизации. В случае загрузки данных отображает индикатор загрузки, а если пользователь авторизован, рендерит дочерние компоненты. |
| Rating | Компонент для отображения рейтинга детали |
| Stars | Компонент для визуализации рейтинга в виде звезд. |
| Login | Компонент для страницы авторизации, где пользователи могут ввести свой email и пароль для входа в систему. После успешной аутентификации пользователь перенаправляется на соответствующую страницу. |
| Registration | Компонент для страницы регистрации нового пользователя. Включает форму с полями для ввода имени, email и пароля. Проверяет корректность введенных данных и отправляет их на сервер для создания нового аккаунта. |
| Cart | Компонент корзины, отображающий добавленные товары. Включает информацию о количестве и стоимости каждого товара, а также общую стоимость корзины. Пользователи могут изменить количество товаров, удалить их и перейти к оформлению заказа. |

Продолжение таблицы 3.10

|  |  |
| --- | --- |
| ClientOrders | Компонент для отображения и управления заказами клиента. Позволяет просматривать историю заказов, их статус. |
| Favorites | Компонент для отображения списка избранных товаров пользователя. Позволяет добавлять и удалять товары из избранного. |
| ProductDetail | Компонент для отображения подробной информации о продукте. Включает изображение, описание, цену, рейтинг и кнопки для добавления в корзину или избранное. Позволяет пользователю узнать все детали о товаре. |
| UserProfile | Компонент профиля пользователя. Позволяет пользователю просматривать и редактировать свою личную информацию, такую как имя, email и пароль. |
| BurgerMenu | Бургер меню с ссылками на страницы |
| InventoryManagement | Компонент для управления инвентарем. Включает инструменты для отслеживания запасов. |
| OperatorOrderManagement | Компонент для управления заказами оператором. Позволяет изменять статус заказов, удалять их и просматривать информацию о пользователях, которые оформили заказы. Также может поддерживать экспорт данных. |
| PartManagement | Компонент для управления запчастями. |
| SupplierManagement | Компонент для управления поставщиками. |
| UserProfile | Компонент профиля пользователя, отображающий и редактирующий личные данные пользователя |
| Home | Компонент содержит информацию про товары |

Логика работы страницы Authorization:

* пользователь вводит свои данные в соответствующие поля форм;
* нажимает кнопку Войти;
* из клиентского кода web-приложения уходит HTTP-запрос в backend (метод login);
* если данные некорректны, выводиться сообщение об ошибке
* в случае успешной авторизации, backend возвращает данные пользователя и jwt-токен, который используется в других запросах
* jwt-токен сохраняется в localStorage (метод setToken);
* пользователь переадресуется на страницу Каталог деталей;

Логика работы страницы Регистрация:

* пользователь вводит свои данные в соответствующие поля форм;
* нажимает кнопку Зарегистрироваться;
* из клиентского кода web-приложения уходит HTTP-запрос в backend (метод registration);
* если данные некорректны, выводиться сообщение об ошибке
* в случае успешной регистрации, backend возвращает данные пользователя и jwt-токен, который используется в других запросах
* jwt-токен сохраняется в localStorage (метод setToken);
* пользователь переадресуется на страницу Котолог запчастей;

Логика работы страницы Каталог запчастей:

* из клиентского кода web-приложения уходит HTTP-запрос в backend (метод getParts);
* полученные запчасти отображаются на странице;
* пользователь вводит поисковый запрос;
* из клиентского кода web-приложения уходит HTTP-запрос в backend (метод getParts);
* полученные запчасти отображаются на странице;
* пользователь выбирает вариант сортировки;
* из клиентского кода web-приложения уходит HTTP-запрос в backend (метод getParts);
* полученные запчасти отображаются на странице;
* пользователь нажимает на название детали;
* пользователь переадресовывается на страницу детали;
* пользователь нажимает на значок избранное;
* из клиентского кода web-приложения уходит HTTP-запрос в backend (метод addPartToFavorites);
* деталь добавлена в избранное;

Логика работы страницы Корзины:

* из клиентского кода web-приложения уходит HTTP-запрос в backend для получения данных корзины (метод getCartItems).;
* полученные товары в корзине отображаются на странице;
* пользователь изменяет количество товара в корзине;
* из клиентского кода web-приложения уходит HTTP-запрос в backend для обновления количества товара (метод updateQuantity);
* после обновления количество товара отображается на странице;
* пользователь нажимает кнопку Удалить на товаре;
* из клиентского кода web-приложения уходит HTTP-запрос в backend для удаления товара из корзины (метод removeFromCart);
* товар исчезает из корзины, и корзина обновляется;

Пользователь выбирает способ доставки:

* изменения в способе доставки сохраняются в локальном состоянии и отображаются на странице;

Пользователь вводит адрес доставки (если выбран способ доставки курьер):

* адрес сохраняется в локальном состоянии;
* пользователь нажимает на кнопку Оформитьзаказ;
* из клиентского кода web-приложения уходит HTTP-запрос в backend для создания заказа (метод createOrder);

Если заказ успешно создан:

* отображается сообщение об успешном заказе;
* корзина очищается с помощью действия clearCart;
* адрес доставки очищается;

Если произошла ошибка:

* отображается сообщение об ошибке;
* общая стоимость корзины отображается на странице;
* кнопка Оформитьзаказ блокируется, если заказ уже в процессе оформления;

Логика работы страницы Избранное

Загрузка избранных запчастей:

* при загрузке страницы компонент FavoritesPage отправляет HTTP-запрос в backend (метод getFavorites) для получения списка избранных запчастей;
* запрос включает параметры пагинации: текущая страница (currentPage) и количество элементов на странице (limit);
* если запрос успешен, полученные избранные запчасти сохраняются в состоянии компонента favorites, а также обновляется общее количество страниц totalPages для пагинации;
* если произошла ошибка при загрузке данных, отображается ошибка с сообщением, полученным из ответа сервера или с дефолтным текстом;
* Отображение избранных запчастей:
* Если данные загружены успешно и есть элементы в списке избранных запчастей:
* для каждого товара отображается его название, описание, цена, совместимость (если доступно) и количество в запасе;
* для каждого элемента списка предоставляется кнопка Удалить, при нажатии на которую отправляется HTTP-запрос в backend (метод removeFavorite) для удаления запчасти из избранного;

Удаление запчасти из избранного:

* при нажатии на кнопку Удалить отправляется запрос на сервер для удаления товара из избранного. Это делается через вызов метода handleRemoveFavorite, который:
* Отправляет DELETE-запрос на сервер по адресу /favorites/{partId};
* после успешного удаления обновляется локальное состояние favorites, и запчасть исчезает из таблицы;
* Также отображается сообщение об успешном удалении запчасти;
* В случае ошибки отображается сообщение об ошибке удаления;

Логика работы страницы Каталог Запчатей:

Загрузка данных:

* при загрузке компонента отправляется HTTP-запрос в backend (метод getSuppliers) для получения списка поставщиков и в метод getParts для получения запчастей с фильтрами и пагинацией;
* данные о поставщиках сохраняются в состоянии suppliers, а данные о запчастях в parts;
* если запросы успешны, отображаются полученные данные. В случае ошибки отображается сообщение об ошибке;

Поиск по названию:

* пользователь вводит текст в поле Поиск по названию;
* каждый раз, когда пользователь изменяет значение поиска, состояние search обновляется, и при отправке формы (нажатии кнопки Применитьфильтры) отправляется HTTP-запрос с параметром search, который передается на сервер для фильтрации запчастей по названию;

Фильтрацияпоцене**:**

* пользователь вводит минимальную и максимальную цену в поля Мин. цена и Макс. Цена;
* эти значения сохраняются в состояниях minPrice и maxPrice;
* при отправке формы, эти параметры также включаются в запрос, чтобы фильтровать запчасти по цене;

Фильтрация по поставщику:

* пользователь выбирает поставщика из выпадающего списка Поставщик;
* при изменении выбора, значение supplierId обновляется, и при отправке формы происходит фильтрация по выбранному поставщику;

Фильтрация по совместимости:

* пользователь вводит данные о совместимости в поле Совместимость;
* это значение сохраняется в состоянии compatibility, и при отправке формы передается в запрос на сервер для фильтрации по совместимости;

Сортировка:

* пользователь выбирает критерий сортировки в поле Сортировать по (например, по цене, названию или дате добавления);
* значение для сортировки сохраняется в состоянии sortBy, а sortOrder управляет направлением сортировки (по возрастанию или убыванию);
* при изменении этих значений, компонент отправляет обновленный запрос с новыми параметрами сортировки;

Логика работы страницы Запчати

Добавление/редактирование запчасти:

* при нажатии кнопки Добавить запчасть открывается модальное окно с формой для ввода данных новой запчасти;
* при редактировании запчасти (нажатие на кнопку Редактировать) заполняются данные формы, и состояние isEditing устанавливается в true;
* в форме можно вводить название, описание, цену, количество на складе и совместимость. Пользователь может добавить несколько полей для совместимости, а также загрузить изображение для запчасти;
* после отправки формы, данные запчасти отправляются на сервер через POST (для добавления новой запчасти) или PUT (для редактирования существующей);
* если добавляется или редактируется изображение, оно отправляется с помощью FormData и метода POST;

Обработка ошибок и загрузки:

* если во время добавления или редактирования запчасти возникает ошибка, она отображается в виде Alert;
* в процессе отправки данных, кнопка отправки и форма могут быть заблокированы, чтобы предотвратить повторную отправку данных;
* если данные успешно сохранены, форма закрывается, и список запчастей обновляется;

Удаление запчасти:

* При нажатии на кнопку Удалить для запчасти, появляется подтверждение действия. Если пользователь подтверждает, отправляется DELETE-запрос на сервер для удаления этой запчасти.
* После успешного удаления, список запчастей обновляется.
* Экспорт данных:
* Реализована возможность экспорта данных о запчастях в два формата: Excel и DOCX.
* Для экспорта в Excel используется библиотека XLSX, которая конвертирует данные о запчастях в формат таблицы и сохраняет файл.
* Для экспорта в DOCX используется библиотека docx, которая создает таблицу с данными запчастей и сохраняет файл в формате .docx.

Логика работы страницы Поставщики.

Добавление и редактирование поставщиков:

* при нажатии на кнопку Добавить поставщика открывается модальное окно с формой для ввода данных нового поставщика;
* при редактировании существующего поставщика (нажатие на кнопку Редактировать), форма заполняется данными этого поставщика, а состояние isEditing устанавливается в true;
* в форме можно вводить название, рейтинг и контактную информацию поставщика. Пользователь может добавить несколько полей для контактной информации;
* после отправки формы, данные поставщика отправляются на сервер через метод POST (для добавления нового поставщика) или PUT (для редактирования существующего);
* если добавляется или редактируется контактная информация, она передается в формате объекта.

Обработка ошибок и состояний загрузки:

* в процессе добавления или редактирования поставщика, если возникает ошибка, она отображается в виде Alert;
* если запрос на добавление или редактирование успешен, форма закрывается, а список поставщиков обновляется;
* во время загрузки данных или сохранения, кнопки отправки и форма могут быть заблокированы для предотвращения повторной отправки данных;

Удаление поставщика:

* при нажатии на кнопку Удалить для конкретного поставщика, появляется окно подтверждения действия;
* если пользователь подтверждает удаление, отправляется DELETE-запрос на сервер для удаления поставщика;
* после успешного удаления, список поставщиков обновляется.

Экспорт данных:

* реализованы функции экспорта данных о поставщиках в два формата: Excel и DOCX;
* для экспорта в Excel используется библиотека XLSX: данные о поставщиках конвертируются в формат таблицы и сохраняются в файл suppliers.xlsx;
* для экспорта в DOCX используется библиотека docx: данные о поставщиках конвертируются в таблицу и сохраняются в файл suppliers.docx.

# 4 Руководство системного программиста

На компьютере должна быть установлена платформа node.js версии не ниже 20.9.0. Официальные установочные пакеты для всех основных платформ можно скачать с сайта https://nodejs.org. Установить PostgreSQL. Официальные установочные пакеты для всех основных платформ можно скачать с сайта <https://www.postgresql.org/download/>.

Алгоритм установки программного обеспечения на локальный компьютер:

* распаковать архив с программным обеспечением;
* в среде разработки Visual Studio Code перейти в папку client;
* установить зависимости командой npm install;
* перейти в папку server;
* установить зависимости командой npm install;
* выполнить импорт базы данных autoparts\_marcetplace в PgAdmin 4;
* выполнить команду nodemon index.js;
* перейти в папку auto-parts-dismantling;
* выполнить команду npm start;

# 5 Описание текстового примера

Чтобы запустить web-приложение необходимо в адресной строке ввести <http://localhost:3000>

Откроется страницаКаталог Запчастей. Результат представлен на рисунке 4.1.

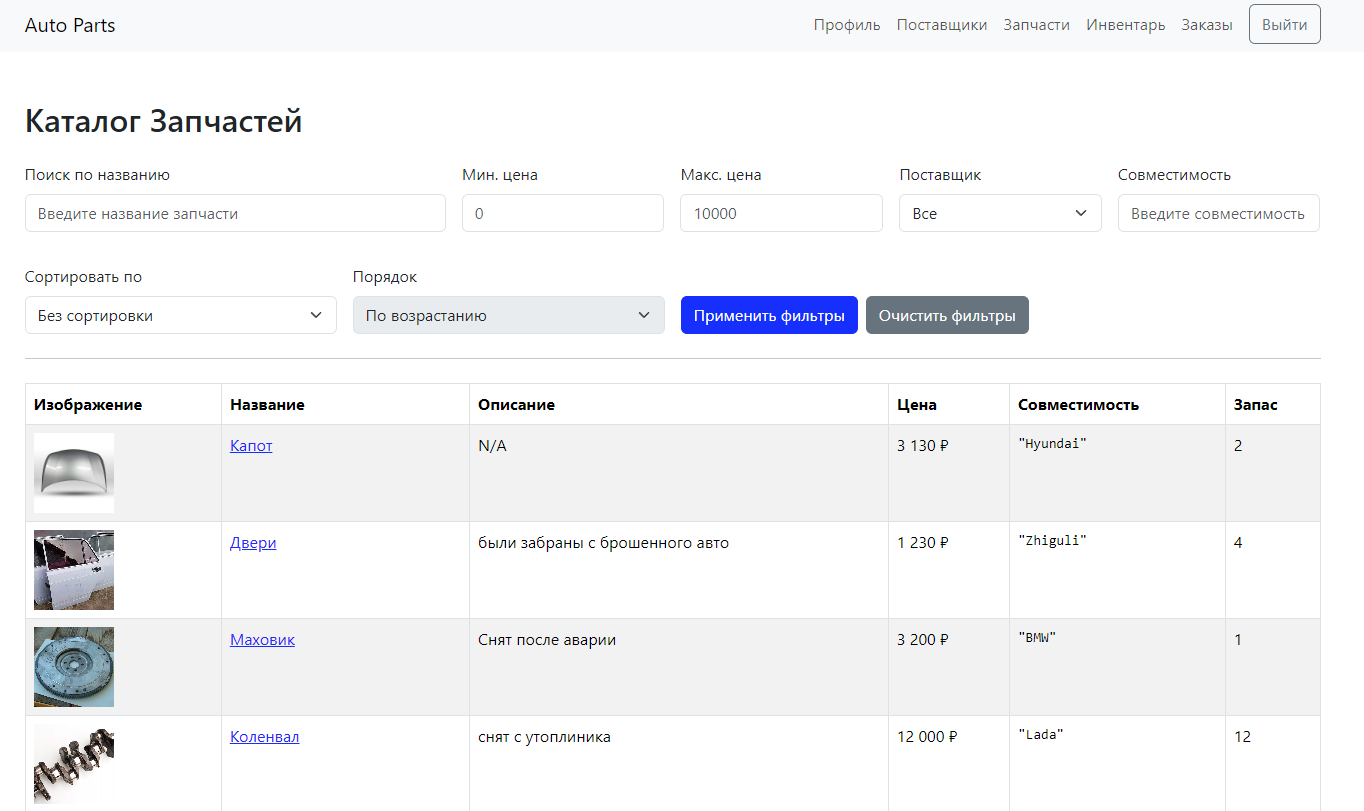


Рисунок 4.1 – Страница Каталог Запчастей

Можно ввести поисковой запрос. Результат представлен на рисунке 4.2.

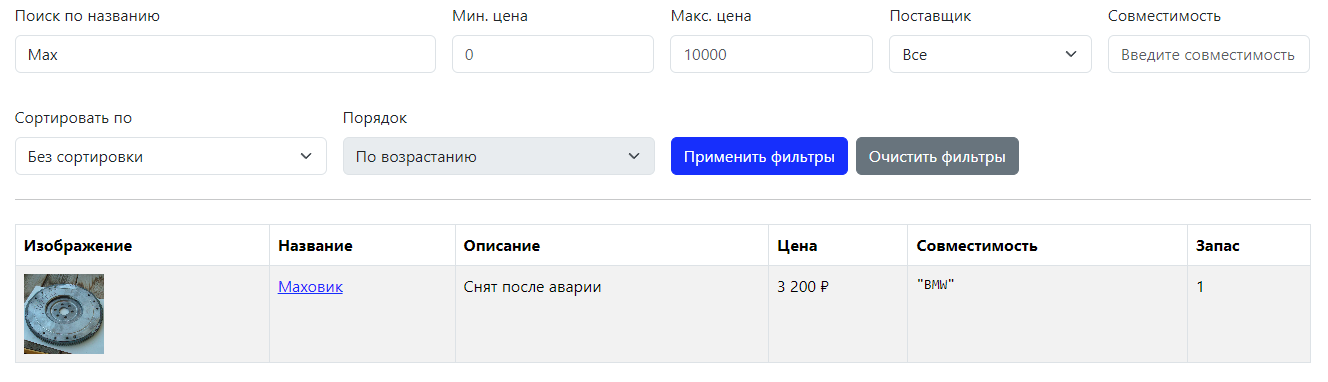


Рисунок 4.2 – Поисковой запрос

Можно выбрать опцию сортировки. Результат представлен на рисунке 4.3.

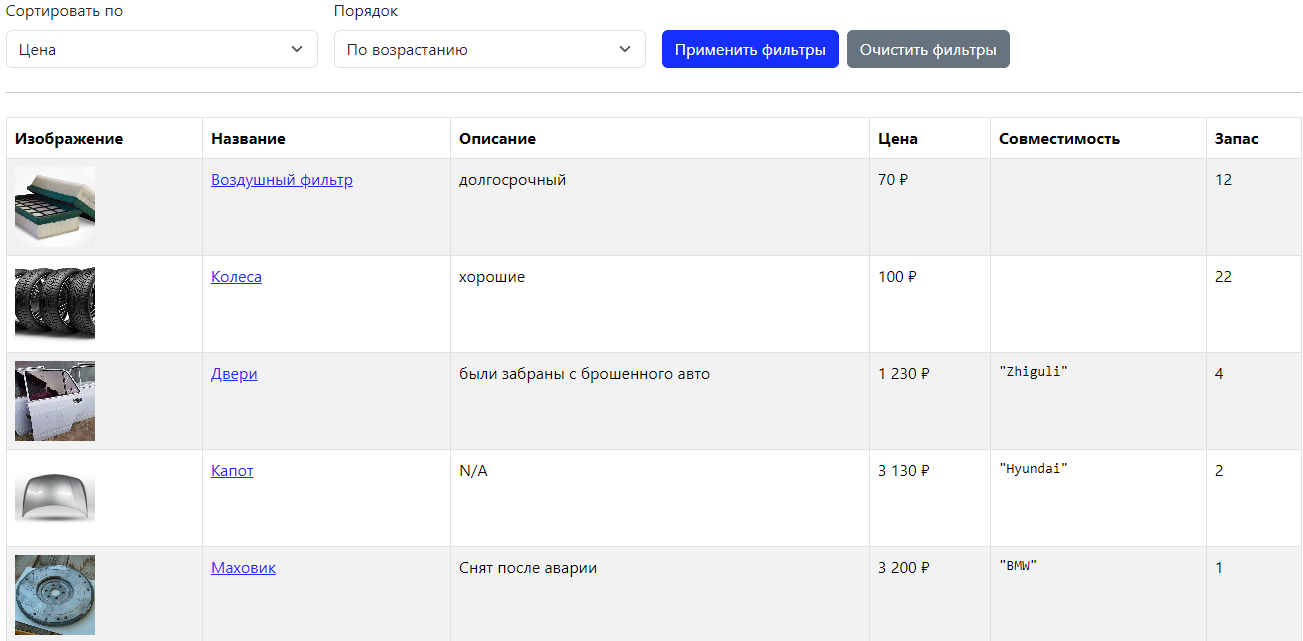


Рисунок 4.3 – Сортировка

Для перехода на страницу авторизации необходимо нажать на кнопку Вход. Результат представлен на рисунке 4.4.

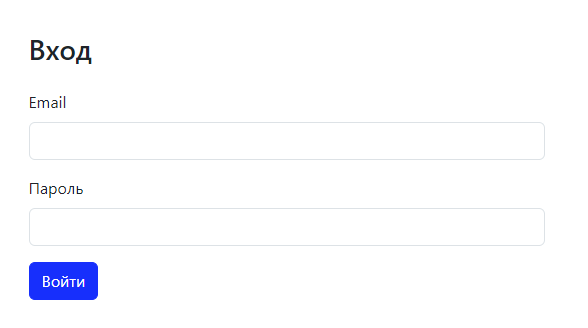


Рисунок 4.4 – Страница Авторизации

Для авторизации необходимо ввести данные в соответствующие поля. Далее нажать на кнопку Войти, после чего будет осуществлено перенаправление на страницу Каталог Запчастей.

Для перехода на страницу регистрации необходимо нажать на кнопку Регистрация на хедаре. Результат представлен на рисунке 4.5.



Рисунок 4.5 – Страница Регистрация

Для регистрации необходимо ввести данные в соответствующие поля. Далее нажать на кнопку Зарегистрироваться после чего будет осуществлено перенаправление на страницу Каталог Запчастей.

Для добавления запчасти в избранное необходимо на странице подробной информации о детали нажать на иконку. Результат представлен на рисунке 4.6.

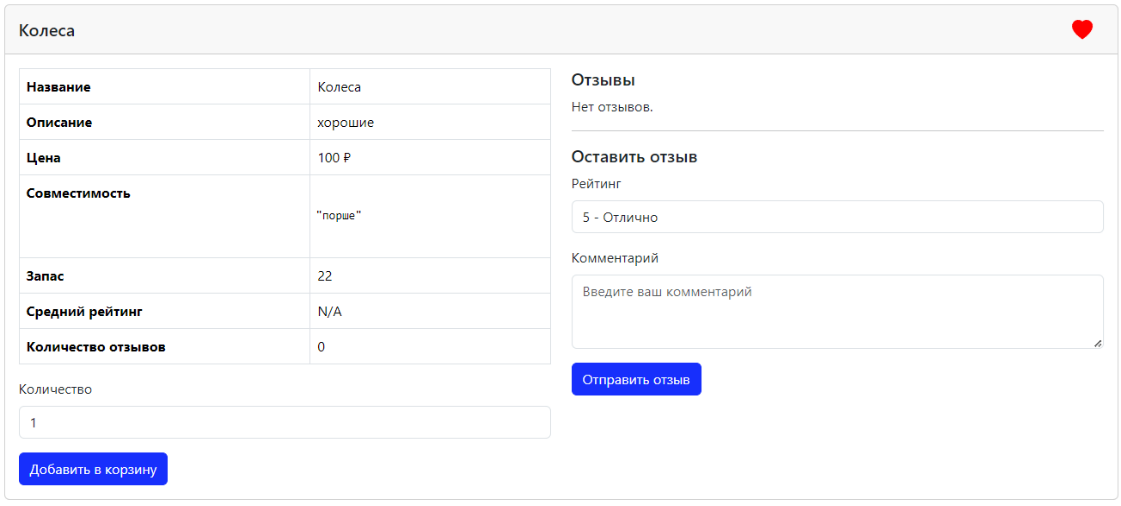


Рисунок 4.6 – Добавление в избранное

Для просмотра детальной информации о запчасти, необходимо нажать на карточку фильма. Результат представлен на рисунке 4.7.

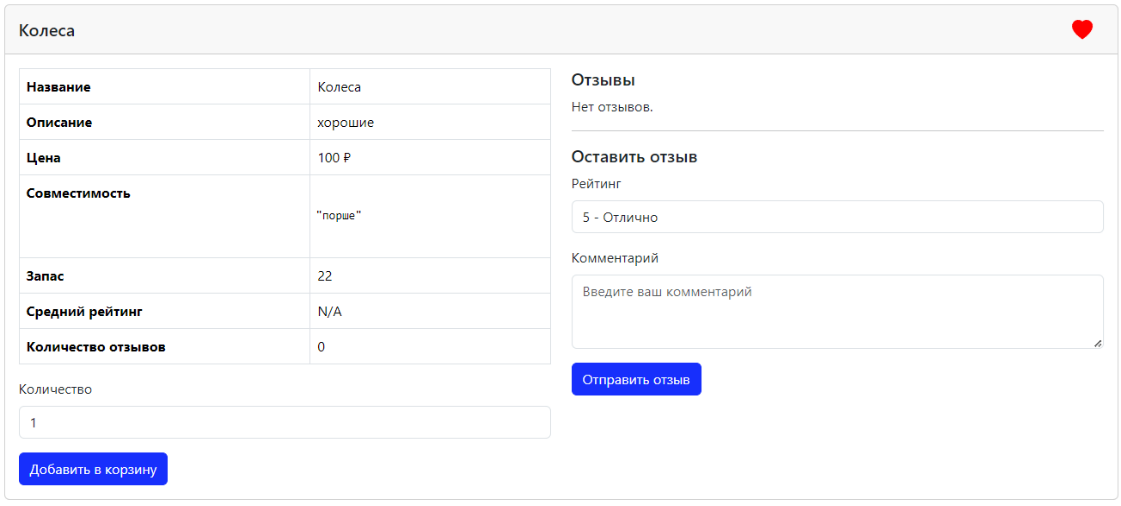


Рисунок 4.7 – Страница Детальное описание детали

Для перехода на страницу корзина необходимо нажать на ссылку Корзина хедере. Результат представлен на рисунке 4.8.

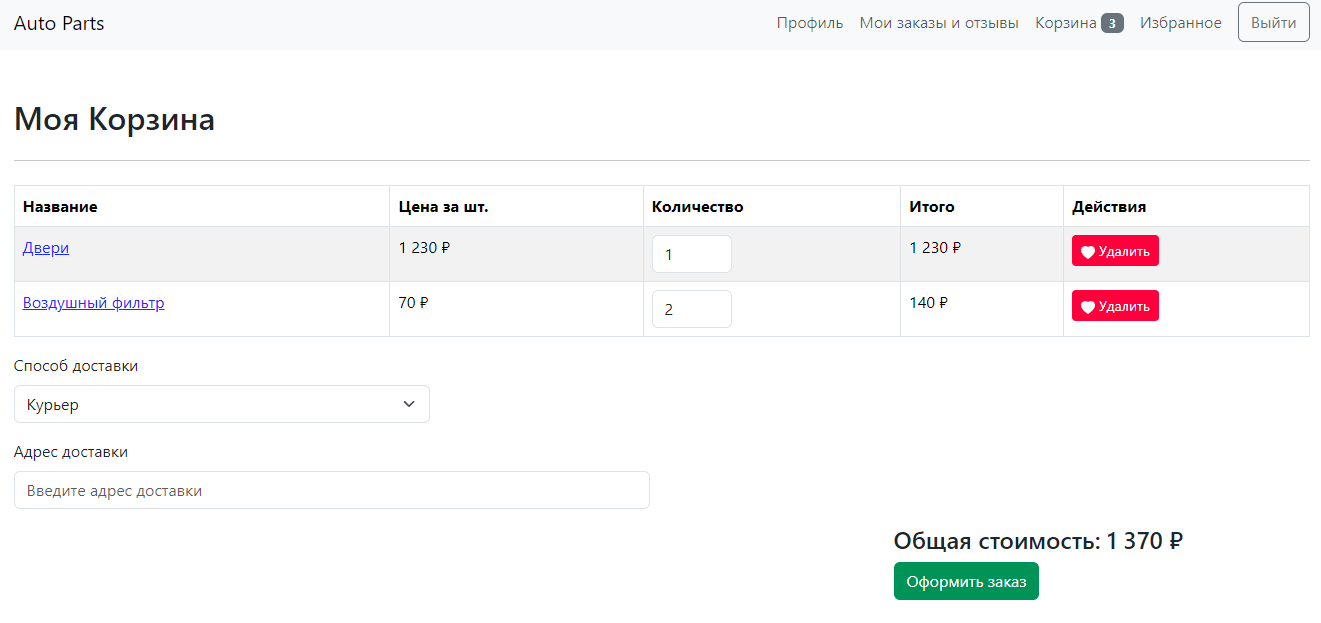


Рисунок 4.8 – Страница Корзина

Для просмотра заказов и отзывов необходимо нажать в хедере на ссылку Мои заказы и отзывы. Результат представлен на рисунке 4.9.

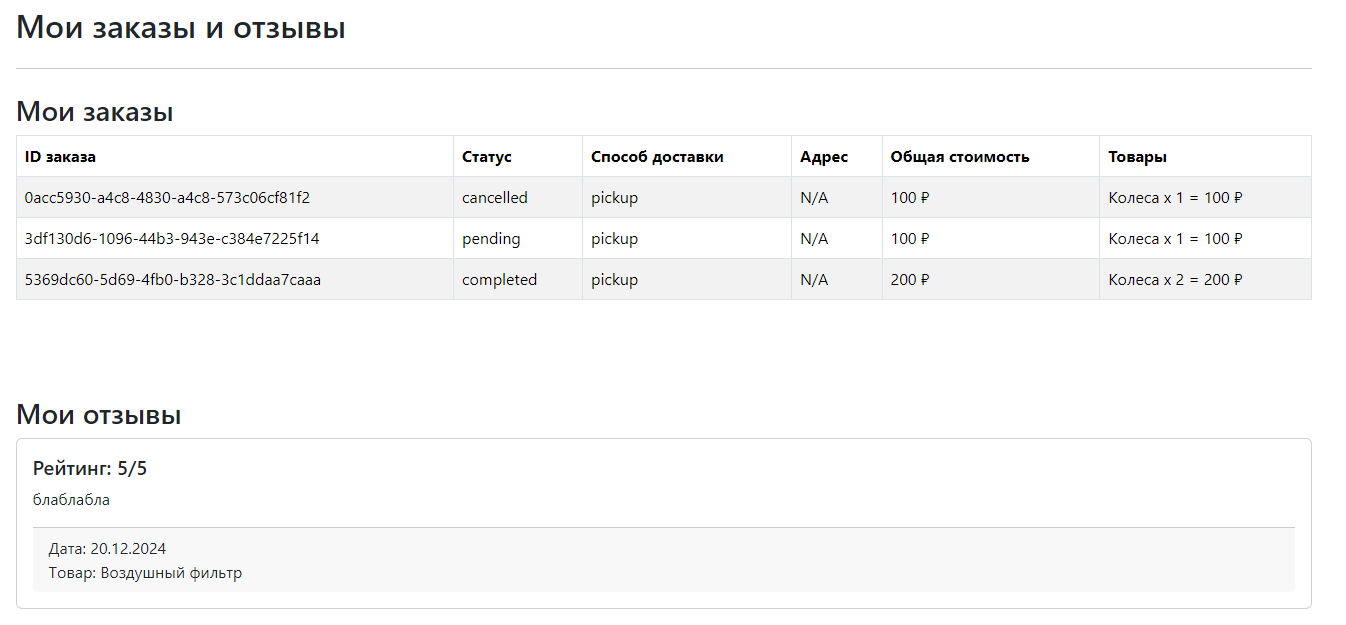


Рисунок 4.9 – Страница Мои заказы и отзывы

Для перехода на страницу Профиль необходимо нажать на ссылку Профиль в хедере. Результат представлен на рисунке 4.10.

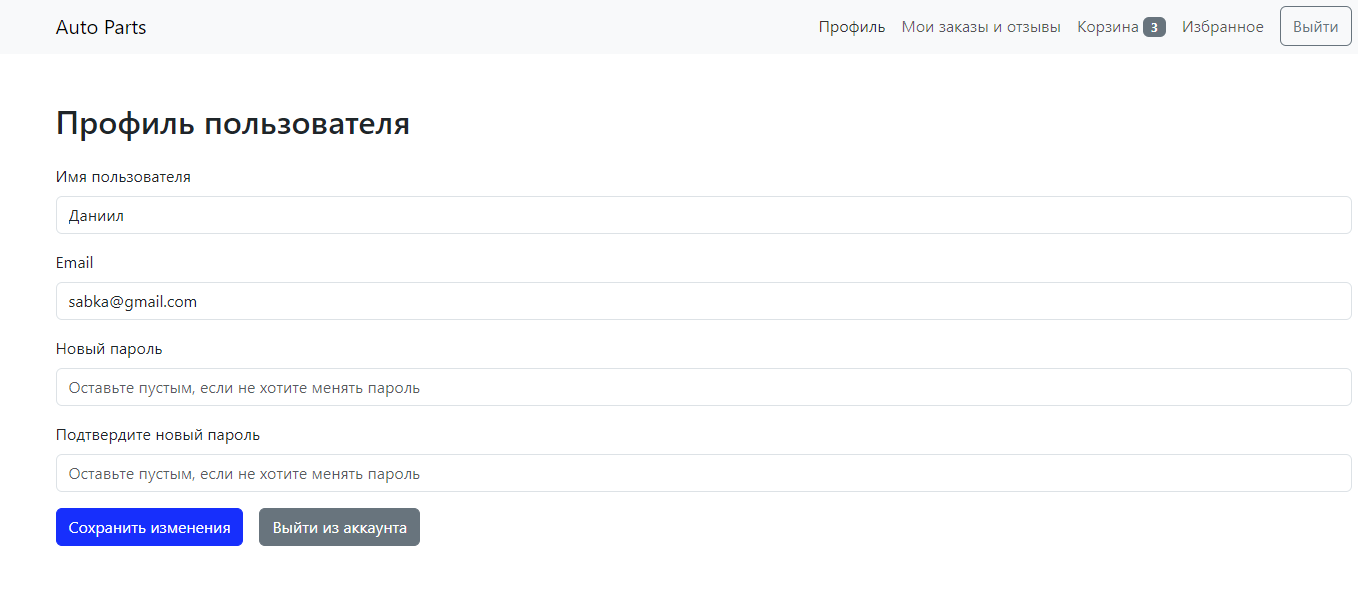


Рисунок 4.10 – Страница Профиль

Для управления запчастями необходимо нажать на ссылку в хедере Запчасти. Результат представлен на рисунке 4.11.

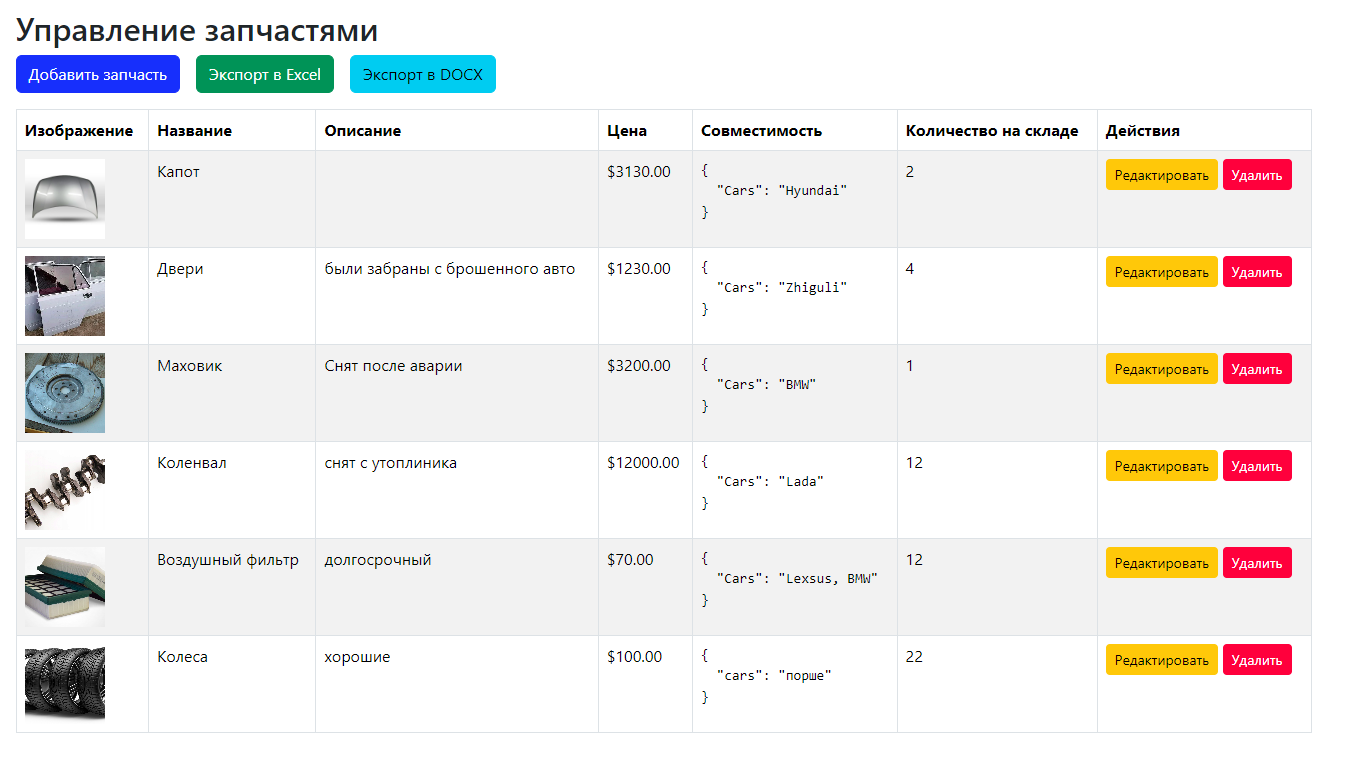


Рисунок 4.11 – Страница Запчасти

Для перехода на страницу Избранное необходимо нажать на кнопку в правом углу в хедере. Результат представлен на рисунке 4.12.

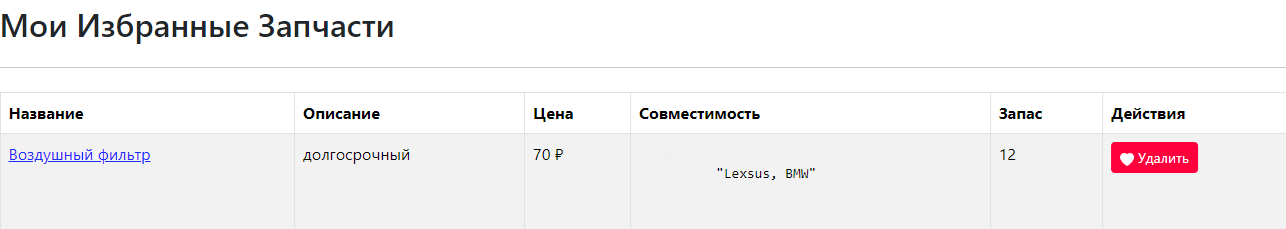


Рисунок 4.12 – Страница Избранное

Можно выбрать совместимость запчастей. Например, для просмотра запчастей которые совместимы с авто необходимо ввести его в фильтре. Результат представлен на рисунке 4.13.

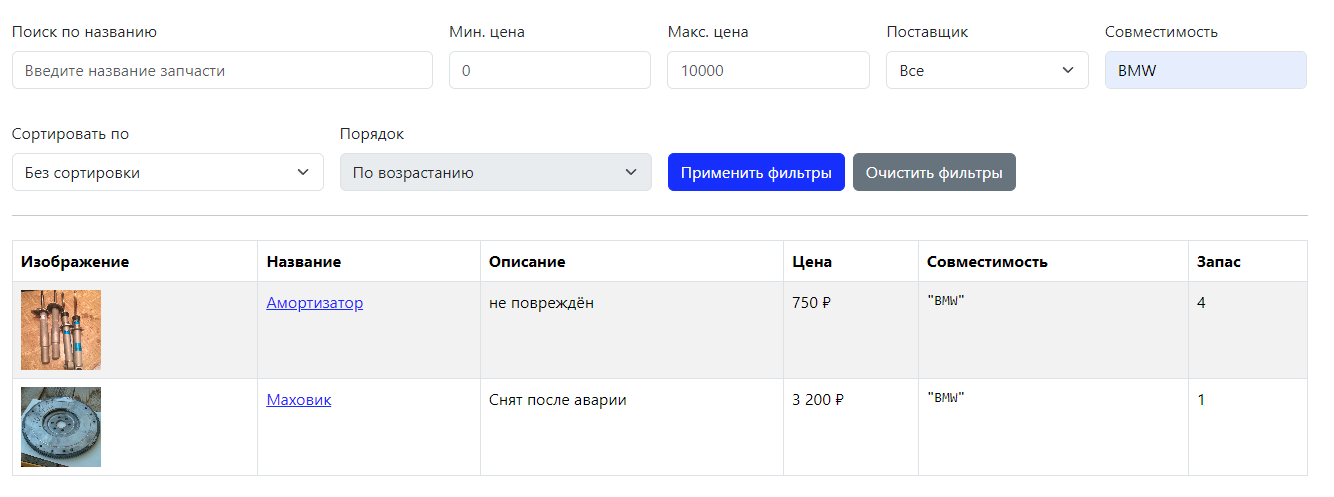


Рисунок 4.13 – совместимость запчастей

Для перехода на страницу Инвентарь необходимо нажать на ссылку Инвентарь. Результат представлен на рисунке 4.14.

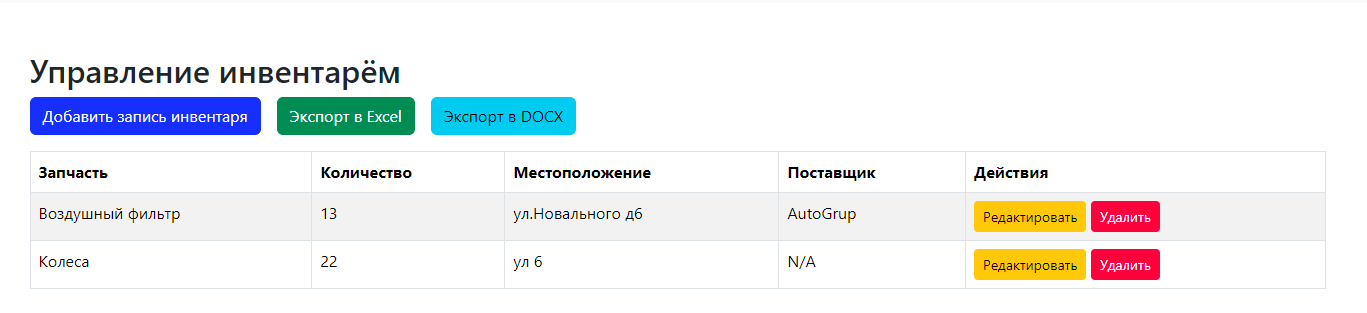


Рисунок 4.14 – Страница Account

Для перехода на страницу поставщиков нужно в хедере нажать на ссылку Поставщики Результат представлен на рисунке 4.15.

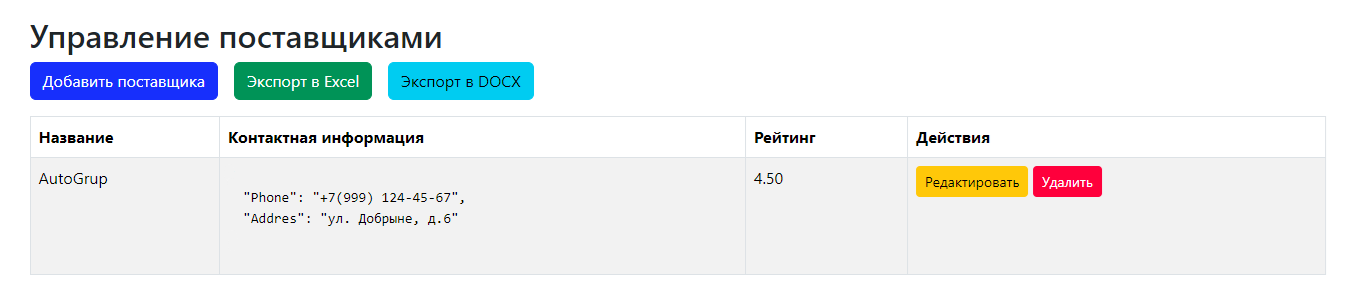


Рисунок 4.14 – Страница Управление поставщиками

# Заключение

В рамках выполнения курсового проекта была успешно достигнута основная цель — разработка веб-приложения «AutoParts», предназначенного для упрощения процесса поиска и заказа автозапчастей. Приложение предлагает пользователям интуитивно понятный интерфейс, который включает функционал регистрации, авторизации, управления профилем, добавления товаров в корзину и оформления заказов.

Кроме того, система предоставляет администраторам возможности эффективного управления товарным ассортиментом и обработкой заказов через специализированную панель. Все поставленные задачи, включая анализ аналогичных решений, выбор технологий и инструментов, проектирование и реализацию клиентской и серверной частей, а также интеграцию с базой данных PostgreSQL, были успешно выполнены.

Основные преимущества приложения заключаются в его удобной структуре и высокой функциональности, а также в использовании современных технологий, таких как React и Axios, что обеспечивает производительность и масштабируемость. Дополнительные функции, такие как экспорт данных и модуль для отзывов, значительно улучшают пользовательский опыт.

Перспективы развития проекта включают:

* расширение функционала управления заказами с автоматическими

уведомлениями и прогнозами сроков выполнения;

* улучшение интерфейса с элементами более сложной визуализации

данных.

Работа над проектом позволила углубить знания в области веб-разработки, включая использование React, работу с PostgreSQL и разработку REST API. Это создало прочный фундамент для дальнейших профессиональных достижений в сфере информационных технологий.

# Список используемых источников

1 React [Электронный ресурс]. Режим доступа: https://react.dev/

2 Built-in React Hooks [Электронный ресурс]. Режим доступа: https://react.dev/reference/react/hooks

3 React Router: Docs [Электронный ресурс]. Режим доступа: https://reactrouter.com/en/main

4 NestJS - A progressive Node.js framework [Электронный ресурс]. Режим доступа: https://nestjs.com/

5 Sequelize | Feature-rich ORM for modern TypeScript & JavaScript [Электронный ресурс]. Режим доступа: https://sequelize.org/

6 React Hook Form [Электронный ресурс]. Режим доступа: https://www.react-hook-form.com/

7 Getting Started | Axios Docs [Электронный ресурс]. Режим доступа: https://axios-http.com/docs/intro

8 Redux Toolkit [Электронный ресурс]. Режим доступа: https://redux-toolkit.js.org/

9 How to Use Redux and Redux Toolkit – Tutorial for Beginners [Электронный ресурс]. Режим доступа: https://www.freecodecamp.org/news/redux-and-redux-toolkit-for-beginners/

10 Vite | Next Generation Frontend Tooling [Электронный ресурс]. Режим доступа: https://vite.dev/