МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

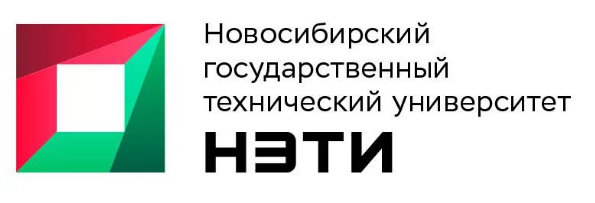
образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**КУРСОВАЯ РАБОТА**

**«*Разработка программного обеспечения для создания и обработки сведений базы данных автосалона*»**

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБc-322», «АВТФ»  *Аврамчук Илья Вячеславович*  «\_\_» декабря 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Доцент кафедры ЗИ*  *Архипова Анастасия Борисовна*  «\_\_» декабря 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

*Защита информации*

Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(полное название кафедры)

##### УТВЕРЖДАЮ

Зав. кафедрой Иванов А. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

**ЗАДАНИЕ  
НА КУРСОВУЮ РАБОТУ**

*Аврамчуку Илье Вячеславовичу*

студенту *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(фамилия, имя, отчество)

*10.05.03 – Информационная безопасность автоматизированных систем*

*10.03.01 – Информационная безопасность*

Направление подготовки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код и наименование направления подготовки бакалавра)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Факультет автоматики и вычислительной техники*

(полное название факультета)

*Разработка программного обеспечения для создания и обработки сведений базы данных автосалона*

Руководитель*Разработка программного обеспечения для создания и ведения кулинарной книги*

Тема\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Календарный план

|  |  |  |
| --- | --- | --- |
| Наименование задач  (мероприятий),  составляющих задание | Дата выполнения задачи  (мероприятия) | Подпись руководителя |
| Развернутая постановка задачи, изучение необходимой научно-технической литературы | 24.06.2025 г. |  |
| Разработка структуры данных и алгоритма решения задачи | 04.07.2025 г. |  |
| Написание текста задачи | 12.07.2025 г. |  |
| Тестирование и отладка программного продукта | 19.08.2025 г. |  |
| Оформление отчета о проделанной работе | 23.08.2025 г. |  |
| Сдача работы руководителю и ее защита | 17.11.2025 г. |  |

Задание согласовано и принято к исполнению.

### Руководитель

### от НГТУ

Архипова Анастасия Борисовна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

к.т.н.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ученая степень, ученое звание)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

### Студент

Аврамчук Илья Вячеславович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия, имя, отчество)

АВТФ, АБс-322

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(факультет, группа)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

СОДЕРЖАНИЕ

Введение6

1. Постановка задачи9
   1. Техническое задание9
2. Алгоритм решения16

2.1. Назначение и характеристики задачи16

2.2. Используемая информация17

2.3. Результаты решения19

2.4. Алгоритм решения19

3. Руководство пользователя22

3.1. Введение22

3.2. Назначение и условия применения22

3.3. Подготовка к работе23

3.4. Описание операций23

3.5. Сообщение пользователю27

3.6. Аварийные ситуации28

4. Руководство системного программиста29

4.1. Общие сведения29

4.2. Структура и описание программы30

4.3. Контейнеризация35

4.4. Описание базы данных36

4.5. Место работы в комплексной защите объекта информации44

5. Контрольный пример51

Заключение63

Список использованных источников65

Приложение66

ВВЕДЕНИЕ

Постоянное развитие веб-технологий и быстрый рост онлайн-сервисов привели к значительному повышению требований к системам управления данными в коммерческих организациях. Особенно это касается предприятий, работающих с большим количеством объектов, имеющих сложную структуру характеристик и высокую стоимость. Одним из таких направлений являются автосалоны и специализированные каталоги техники, где необходимо эффективно хранить, обрабатывать и актуализировать сведения о транспортных средствах.

Современные пользователи ожидают от подобных систем удобства, скорости работы, прозрачности операций и возможности совершать действия онлайн. Однако традиционные подходы, основанные на использовании локальных файлов, разрозненных таблиц и ручной обработки данных, являются время затратными, трудоёмкими и подверженными ошибкам. Особенно остро эта проблема стоит, когда речь идёт о сложных объектах, например, болидах Формулы-1, которые имеют большое количество характеристик, привязку к командам, сезонам, статистике и требуют корректного отслеживания доступности, статуса и истории заявок.

Актуальность разработки программного обеспечения для автоматизации работы автосалона обусловлена необходимостью повышения эффективности обработки данных, снижением вероятности ошибок и улучшением взаимодействия с пользователем. Использование современных технологий, таких как Java, Spring Boot, Docker и PostgreSQL, которые позволяют создавать гибкие, масштабируемые и надёжные приложения, соответствующие требованиям действующих методических указаний по курсовому проектированию.

Целью данной работы является разработка программного обеспечения, обеспечивающего хранение, обработку и управление сведениями об автомобилях и болидах, а также реализацию пользовательского интерфейса, позволяющего осуществлять регистрацию, авторизацию, оформление заявок и просмотр каталога техники.

Для достижения поставленной цели необходимо решить следующие задачи:  
 1. провести анализ существующих методов хранения и обработки данных, используемых в подобных информационных системах;  
 2. разработать структуру базы данных и определить взаимосвязи между объектами предметной области;  
 3. создать серверную часть приложения, включающую реализацию бизнес-логики, обработку запросов пользователей и обеспечение доступа к данным;  
 4. разработать интерфейс для работы администратора и пользователя: просмотр каталога, управление болидами, обработка заявок;  
 5. реализовать механизмы аутентификации и авторизации, обеспечивающие защиту данных;  
 6. выполнить тестирование приложения, оценить его устойчивость и работоспособность в типовых сценариях использования.

Ожидаемым результатом работы является создание программного продукта, предоставляющего удобный и эффективный инструмент для учёта и обработки данных об автомобилях и болидах, а также автоматизацию операций, связанных с их продажей и администрированием. Разрабатываемая система должна быть масштабируемой, удобной в сопровождении, а также расширяемой для дальнейшего развития.

Новизна работы состоит в применении современных веб-технологий для цифровизации процессов, которые ранее выполнялись вручную. Разработка позволяет формализовать процесс обработки заявок, обеспечивать целостность данных и создавать удобную среду взаимодействия между пользователем и системой, что является актуальным направлением в условиях перехода к цифровым сервисам.

Теоретической основой написания курсовой работы являлись работы следующих авторов: Стив Макконелл, Томас Коннолли и Каролин Бегг [1, 2], посвящённые разработке программного обеспечения.

Практической основой написания курсовой работы являлась документация по докеру, PostgreSQL, а также материалы по разработке веб-приложений [3, 4, 5].

1. ПОСТАНОВКА ЗАДАЧИ
   1. Техническое задание
      1. Введение:

Целью данного проекта является создания веб-приложения для автосалона спортивных болидов Formula 1, предоставляющего пользователям возможность просматривать каталог болидов разных сезонов, фильтровать их по стоимости, командам и техническим параметрам, добавлять выбранные модели в корзину и оформлять заявки на покупку.

Приложение обеспечивает безопасную и удобную среду для взаимодействия между пользователями и администраторами, включая добавление новых болидов, управление заказами и обработку заявок.

Актуальность проекта заключается в необходимости цифровизации процесса выбора и покупки уникальных коллекционных машин. Традиционные методы продажи спортивных болидов, основанные на личных показах и офлайн-консультациях, являются недостаточно эффективными. Разработка веб-приложения позволяет упростить взаимодействие между покупателем и продавцом, предоставляет актуальную информацию по наличию и состоянию болидов и исключает ошибки, возникающие при ручной обработке заявок.

Разрабатываемое программное обеспечение обеспечивает пользователю возможность выбора болидов, их сортировки, формирования корзины и создания заявки, а администратору предоставляет удобный инструмент для управления каталогом, пользователями и обработкой заказов.

* + 1. Платформа

Веб-приложение работает в любом современном браузере (Yandex Browser, Google Chrome, Mozilla Firefox, Opera, Edge).

Серверная часть работает на платформе Spring Boot, клиентская часть реализована с использованием HTML и шаблонов Thymeleaf. Данные

хранятся в PostgreSQL.

* + 1. Целевой аудиторией являются:

Коллекционеры и энтузиасты автомобильного спорта, интересующиеся болидами Formula 1, желающие приобрести болиды прошлых сезонов в качестве коллекционных экземпляров.

* + 1. Функциональные требования
       1. Основные функции:
* регистрация и авторизация пользователя;
* добавление и удаление болидов из корзину;
* просмотр карточки болида;
* оформление заявки на покупку болида;
* управление болидами (создание, редактирование, удаление);
* управление пользователями и их заявками.

Рабочее пространство пользователя должно синхронизироваться базой данных, обеспечивая актуальные данные о наличии болидов, статусах заявок и содержимом корзины.

Макет начальной страницы веб-приложения с переходом к регистрации и авторизации пользователя представлен на рисунке 1.1.

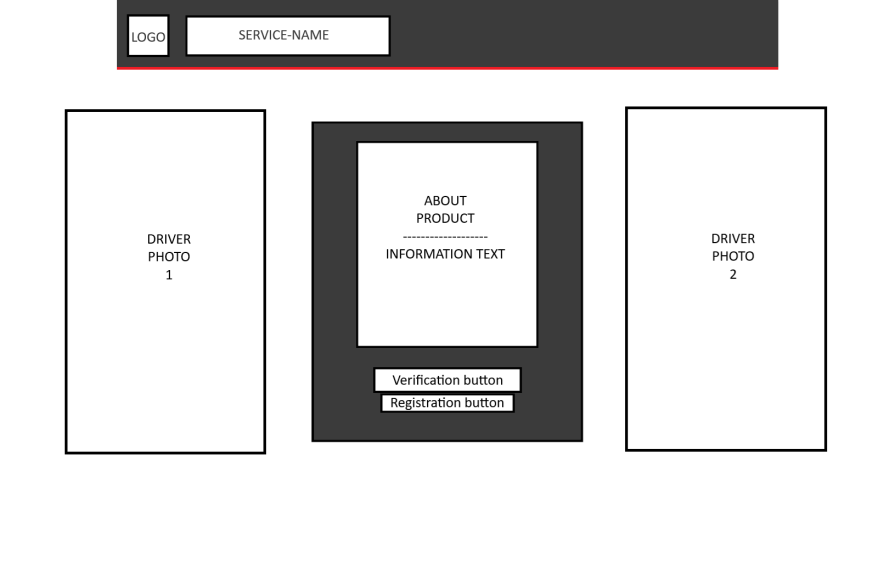


Рисунок 1.1 – Макет начальной страницы

Макет страницы регистрации представлен на рисунке 1.2.

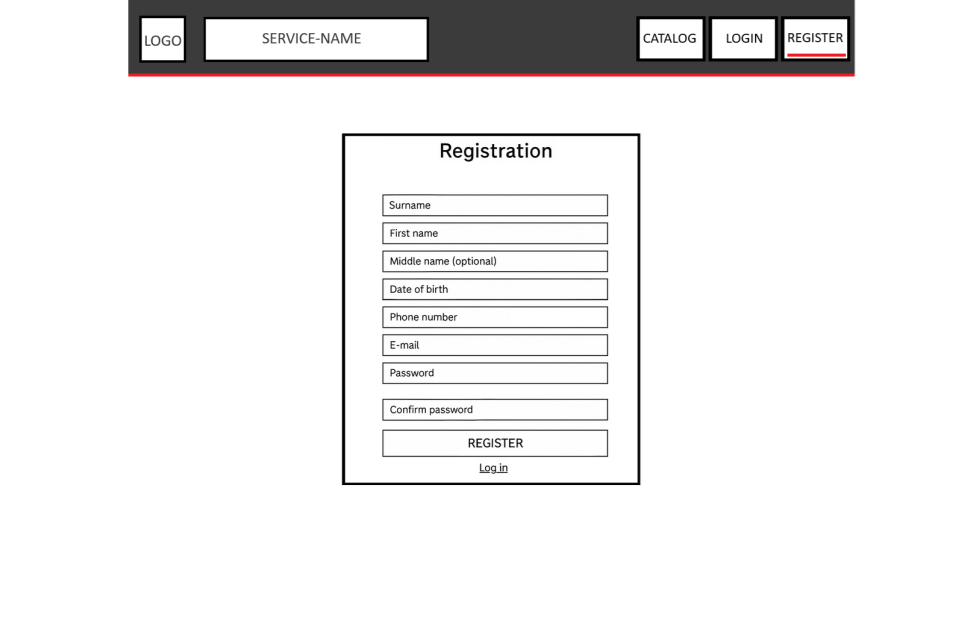


Рисунок 1.2 – Макет страницы регистрации

После успешной регистрации пользователь перенаправляется на страницу авторизации для входа в каталог.

Макет страницы входа представлен на рисунке 1.3.

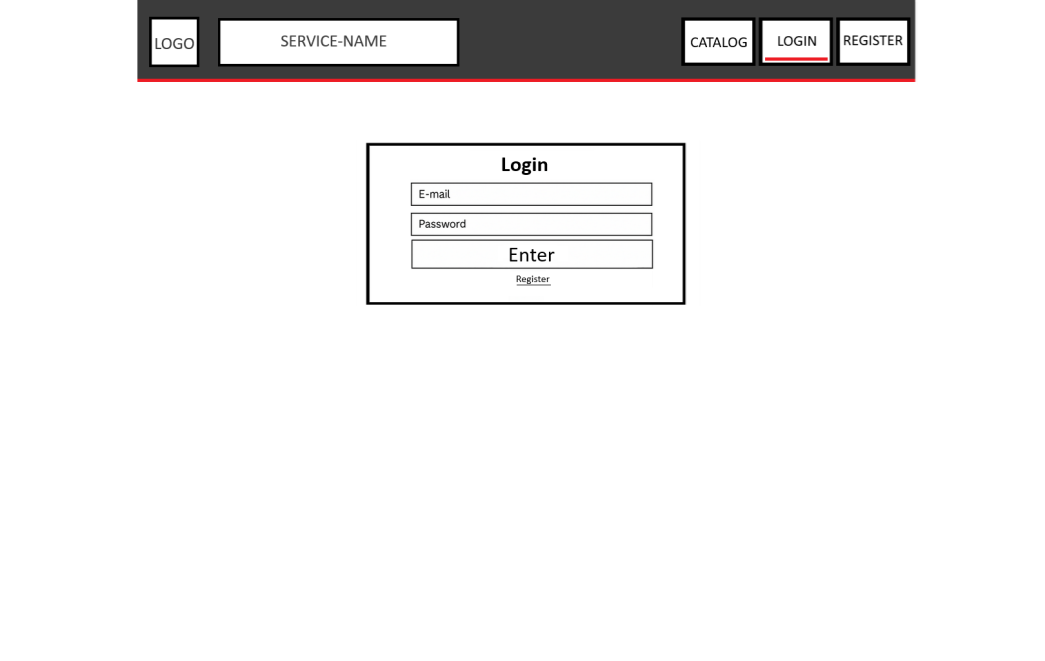


Рисунок 1.3 – Макет страницы входа

После успешной авторизации пользователь перенаправляется на страницу с каталогом болидов. На этой странице пользователь может настроить фильтры на поиск болидов, а в верхней панели навигации доступны элементы управления: каталог, корзина, выйти.

Макет страницы каталога представлен на рисунке 1.4.

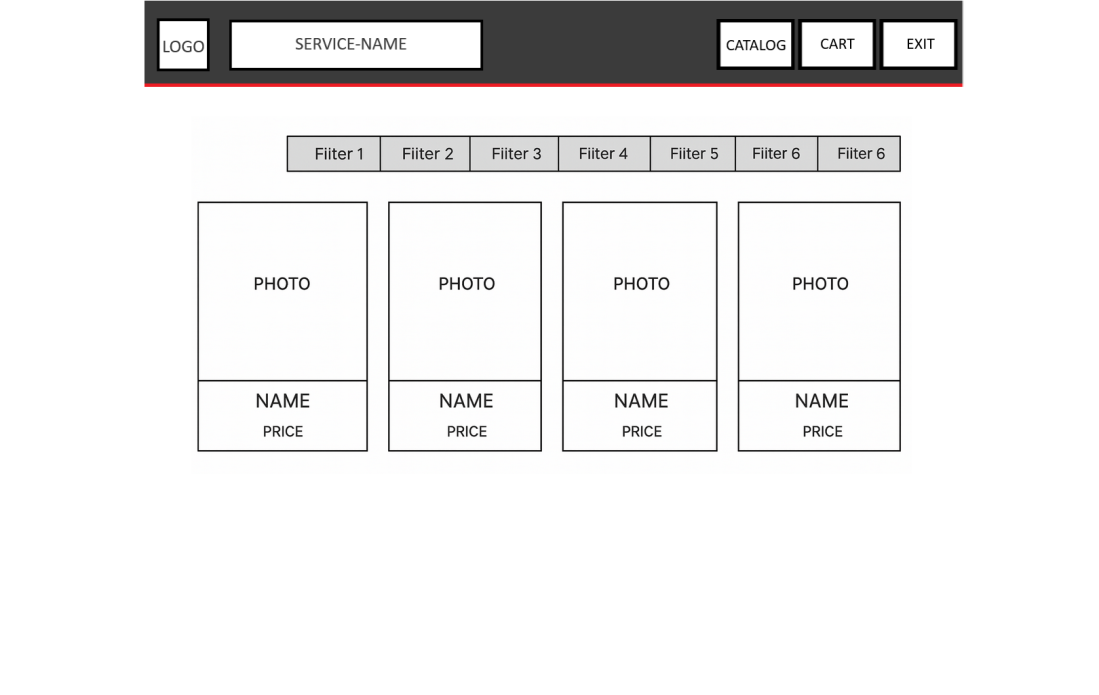


Рисунок 1.4 – Макет страницы каталога

При нажатии пользователем на навигационную кнопку корзины осуществляется переход на страницу управления корзиной.

Макет страницы корзины представлен на рисунке 1.5.

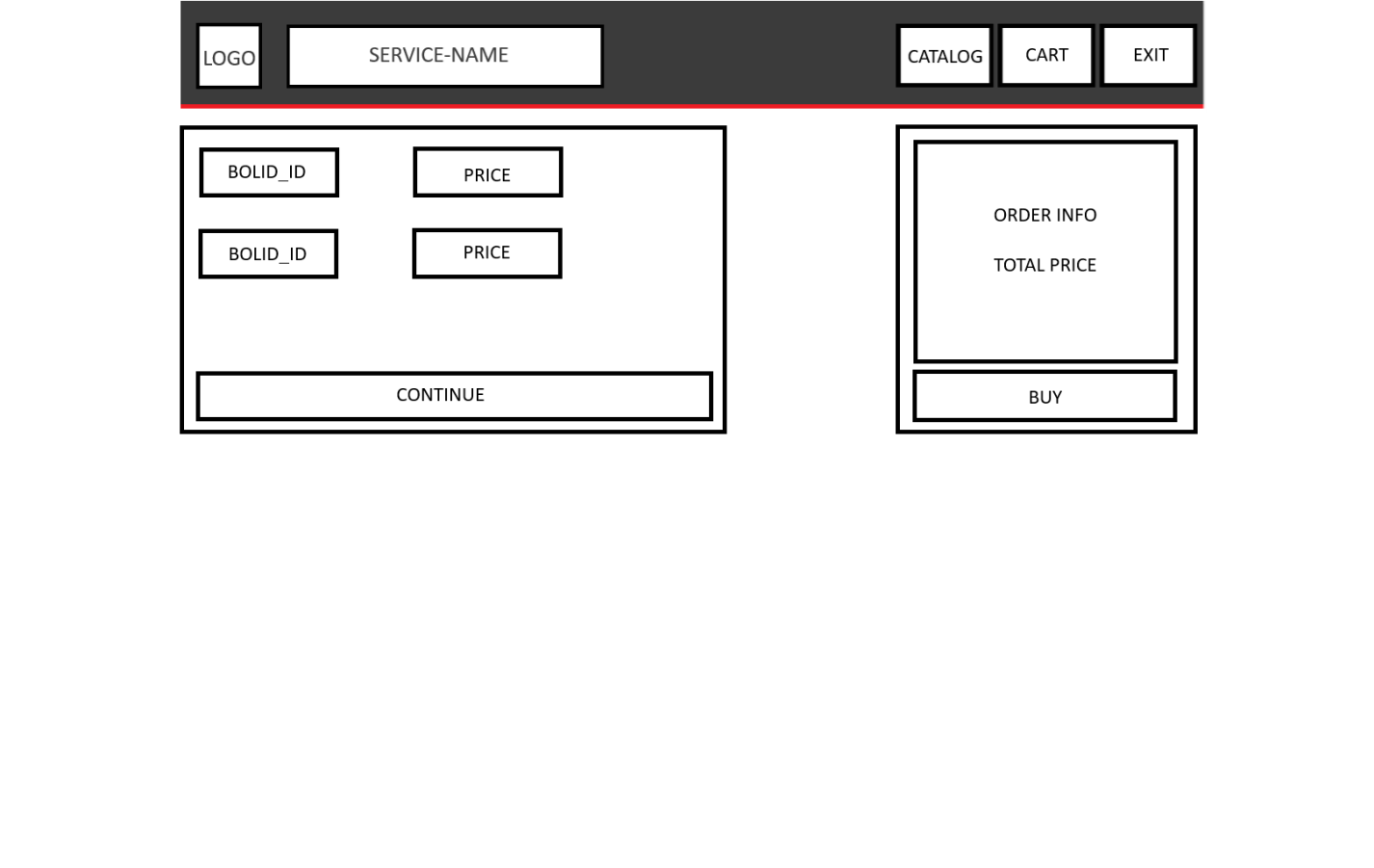


Рисунок 1.5 – Макет страницы корзины

* + 1. Нефункциональные требования

1.1.5.1. Производительность:

* время загрузки не более 3 секунды при стандартной нагрузке;
* обработка пользовательских запросов должна выполняться без заметных задержек.

1.1.5.2. Безопасность:

* хэширование пароля пользователя;
* защита от Cross-Site Scripting;
* защита от SQl-инъекций.

1.1.5.3. Удобство использования:

* простой и интуитивно понятный интерфейс, не требующий подготовки для работы;
* валидация форм при вводе данных, отображение ошибок и подсказок пользователю;
* логичное расположение кнопок добавления, редактирования и удаления болидов;
* единый стиль отображения карточек, заявок и форм.

1.1.6. Технологические требования:

– frontend: HTML, CSS, Thymeleaf, JavaScript;

– backend: Java 17, Spring Boot;

– база данных: PostgreSQL 16;

– система сборки: Maven;

– контейнеризация: Docker.

1.1.7. Дизайн

* + - 1. Визуальный стиль:
* минималистичный интерфейс в спортивной стилистике Formula 1;
* основная палитра использует темные цвета и акцентные красные элементы для управления;
* ключевые элементы выделены контрастными цветами.
  + - 1. Макеты

Основные макеты:

* страница с формой регистрации;
* страница с формой входа;
* страница каталога болидов;
* страница детального просмотра болида;
* страница корзины;
* страница оформление заявки;
* страница панели администратора;
* страница профиля пользователя.

1.1.8. Этапы разработки:

1.1.8.1 Сбор требований и проектирование интерфейса

Разрабатывается архитектура приложения, макеты страниц, логика переходов между ними. Определяются сущности базы данных (пользователи, болиды, команды, заявки), формируется структура проекта.

1.1.8.2 Разработка прототипа и его тестирование

Создается первичный работающий каркас сайта с базовой навигацией и шаблонами страниц. Проводится ручное тестирование пользовательских сценариев.

1.1.8.3 Реализация функционала

Реализуются основные функции: регистрация и авторизация пользователей, управление болидами, корзина, оформление заявок, панель администратора. Настраивается взаимодействие frontend и backend.

1.1.8.4 Тестирование системы (функциональное, нагрузочное)

Проводится функциональное тестирование (проверка бизнес логики) и нагрузочное тестирование (поведение при большом количестве болидов, пользователей и заявок).

1.1.9. Сроки выполнения

Проект необходимо выполнить в течение 2 месяцев с момента начала разработки.

1.1.10. Заключение

Разработка веб-приложения для автосалона формульных болидов позволяет пользователям удобно просматривать ассортимент, изучать характеристики моделей и оформлять заявки на покупку. Система предоставляет корзину, подробные карточки болидов и механизм резервирования. Административная часть обеспечивает управление пользователями, болидами и заявками, что делает приложение полноценным инструментов для работы автосалона. Удобный интерфейс и современный технологический стек делают процесс взаимодействия с системой простым, быстрым и эффективным.

2. АЛГОРИТМ РЕШЕНИЯ

2.1 Назначение и характеристика задачи

Целью настоящей работы является разработка веб-приложения, предназначенного для просмотра, фильтрации и оформления заявок на легендарные болиды Формулы-1. Приложение предоставляет пользователю возможность просматривать каталог болидов, изучать подробную информацию о каждом экземпляре, добавлять их в корзину и оформлять заявку на приобретение. Для администратора предусмотрен расширенный функционал управления моделями: создание, редактирование и удаление карточек болидов.

Основные задачи разработки:

– разработка структура реляционной базы данных;

– разработка пользовательского интерфейса и его ключевых модулей;

– реализация функционала взаимодействия с каталогом (просмотр, фильтрация, сортировка);

– разработка корзины и системы оформления заявок;

– реализация административной панели для управления болидами и пользователями;

– обеспечение аутентификации и авторизации, защита данных пользователей;

– разработка удобного, интуитивно понятного интерфейса, не требующего предварительного обучения.

Пользователи состояния объектов:

– корректность данных (валидация форм, отсутствие ошибок при вводе);

– время отклика системы при выполнении операций фильтрации и загрузке элементов каталога.

Выходная информация:

– список всех доступных болидов;

– карточка болида с подробным описанием и характеристиками;

– данные зарегистрированных пользователей;

– содержимое корзины;

– оформленные заявки.

Сбор информации:

– данные о пользователях вводятся при регистрации и авторизации;

– данные о болидах добавляются администратором;

– данные о корзине и заявках формируются пользователем и процессе взаимодействия с корзиной.

Контроль и корректировка:

– валидация всех вводимых данных;

– вывод сообщений об ошибках и подсказок при неверном вводе;

– автоматическое обновление данных каталога, корзины и истории заявок.

2.2 Используемая информация

2.2.1. Анализ аналогичных платформ туристических агенств.

На рисунке 2.1 представлен скриншот скриншот сайта «f1authentics.com» [6]. Этот сайт предоставляет каталог болидов Формулы 1, шоу-моделей и коллекционных экземпляров, лицензированных как оригинальные болиды или аналогичные версии. На сайте реализованы фильтры по году выпуска, команде, цене и статусу.

Преимуществом данного решения является наличие развитой системы представления данных: каждая карточка болида содержит технические характеристики, фотографию, описание состояния, год выпуска, принадлежность команде и цену. Дополнительно используется фильтрация по команде и коллекции, что облегчает поиск нужного болида. Также важным аспектом является высокий уровень визуального оформления и акцент на коллекционную ценность товара.

Однако система «F1 Authentics» ориентирована на коммерческий рынок элитных коллекционных товаров. Соответственно, функционал включает элементы, не требуемые в рамках разрабатываемого веб-приложения (аукционные лоты, продажа экипировки, сувенирной продукции, предметов из боксов команд). Это делает интерфейс платформы более сложным и перегруженным для конечного пользователя, заинтересованного в конкретных моделях болидов.

В отличие от неё, разрабатываемое приложение имеет узкую специализацию, оно предназначено для отображения каталога болидов различных сезонов и команд, оформления заявок на покупку, а также управления корзиной. Интерфейс веб-приложения выполнен в минималистичной форме, без лишних элементов, что повышает удобство взаимодействия с системой и снижает когнитивную нагрузку на пользователя.

Таким образом, платформа «F1 Authentics» не полностью соответствует целям проекта. Она ориентирована на коммерческую продажу реальных болидов и коллекционных предметов, имеет сложную структуру, большое количество категорий и маркетинговых блоков. Это усложняет навигацию и отвлекает пользователя от основной задачи, а именно поиск болида и отправка заявки. Разрабатываемое приложение решает эти задачи более локально и эффективно, предоставляя лёгкий доступ к каталогу, корзине и функционалу оформления заявки.

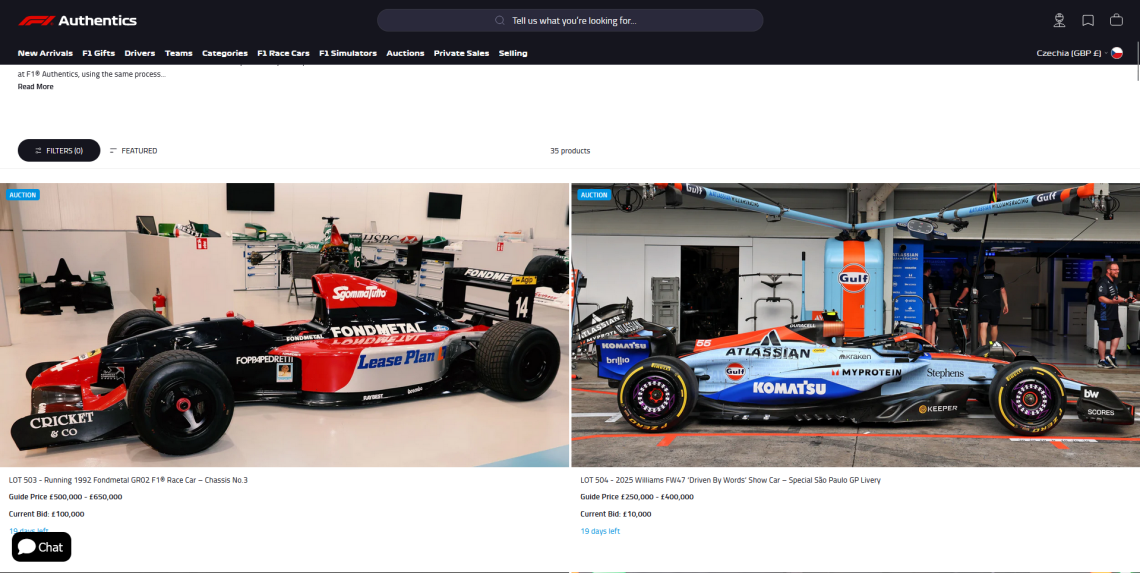


Рисунок 2.1 – скриншот сайта «f1authentics.com»

2.3 Результаты решения

Результатом решения поставленной задачи является реляционная база данных, обеспечивающая хранение информации о болидах, пользователях, корзине и заявках. Система реализует:

– сохранение и изменение данных о болидах (команда, сезон, цена, пробег, состояние доступности);

– хранение пользовательских данных и их статусов;

– фиксацию содержимого корзины и привязку её к конкретному пользователю;

– регистрацию заявок с фиксацией параметров выбранного болида;

– оформленные заявки;

– автоматическое обновление состояния болида при оформлении заявки.

2.4 Алгоритм решения

2.4.1 Серверная часть:

– клонирование репозитория;

– переход в рабочую директорию приложения;

– запуск серверной части командой (docker-compose up --build);

– запуск контейнера PostgreSQL;

– автоматическая инициализация база данных, создание необходимых таблиц;

– установление соединения между сервером приложения и базой данных;

– дальнейшая работа осуществляется до момента остановки контейнера или включения рабочей станции;

2.4.2 Клиентская часть:

1. Регистрация пользователя.

1.1. Ввести фамилию, имя отчество, дату рождения, номер телефона, выбрать пол, e-mail и пароль;

1.2. Нажать кнопку «Зарегистрироваться».

2. Авторизация в системе.

2.1. Ввести e-mail и пароль;

2.2. Нажать кнопку «Войти».

3. Работа с каталогом болидов.

3.1. Настроить фильтры по сезонам, именам, цене, годам или наличию;

3.2. Нажать кнопку «Применить фильтр»;

3.3. Выбрать болид и нажать кнопку «Подробнее».

4. Просмотр карточки болида.

4.1. Для добавления болида в корзину нужно нажать кнопку «Добавить в корзину».

4.2. Для выхода из карточки болида нужно нажать на кнопку «Каталог».

5. Работа с корзиной.

5.1. Нажать кнопку «Корзина» в верхней панели;

5.2. Просмотреть список добавленных болидов;

5.3. При необходимости удалить отдельные позиции нажав на кнопку «Удалить»;

5.4. При необходимости полностью очистить корзину нужно нажать на кнопку «Очистить заявку»;

5.5. Чтобы вернуться обратно в каталог нужно нажать кнопку «Продолжить покупки»;

5.4. Нажать кнопку «Оформить заявку» для фиксации выбора.

6. Оформление заявки.

6.1. В заявке фиксируется модель болида, цена болида, дата оформления заявки и номер заявки;

6.2. Статус болида изменяется на «Недоступен»;

6.3. Отображается уведомление об успешной отправке заявки;

6.4. Корзина очищается.

7. Выход из системы.

7.1. Нажатие на кнопку «Выйти» приводит к завершению пользовательской сессии и возвращению на начальную страницу.

3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

3.1 Введение

3.1.1 Назначение системы

Система AVRAMCHUK BOLID MOTORSPORT предназначена для оформления заявок на покупку болидов Формулы-1, просмотра каталога автомобилей, управления персональной корзиной и взаимодействия с функционалом администратора. Пользователи могут просматривать доступные болиды, фильтровать их по сезонам, производителям и стоимости, изучать карточки моделей, а также формировать заявки на покупку. Приложение предоставляет удобный интерфейс, не требующий специальных навыков, и обеспечивает безопасную работу с пользовательскими данными.

3.1.2 Программа предоставляет пользователю следующие возможности:

* просмотр каталога болидов;
* фильтрация болидов по параметрам (сезон, производитель, цена, наличие);
* отображение детальной карточки выбранного болида;
* добавление болидов в корзину и её последующее редактирование;
* оформление заявки на покупку нескольких болидов;
* регистрация, авторизация и выход из профиля;
* для администратора – просмотр заявок всех пользователей.

3.2 Назначение и условия применения

3.2.1 Целевая аудитория

Система предназначена для частных пользователей, желающих приобрести болид формульной команды, а также для администратора, который управляет каталогом автомобилей и обрабатывает заявки.

3.2.2 Условия применения

Для использования системы необходимо:

* базовые навыки работы с компьютером;
* умение пользоваться веб-браузером;
* наличие интернет-соединения;
* наличие зарегистрированной учётной записи для взаимодействия с каталогом и оформлением заявок.

3.3 Подготовка к работе

Чтобы начать работу с веб-приложением, пользователю необходимо:

1. Перейти по адресу веб-сервера;
2. На начальной странице нажать кнопку Регистрация или Вход;
3. Авторизоваться, после чего автоматически открывается страница каталога болидов.

3.4 Описание операций

3.4.1 Введение

В разделе представлены основные функции пользовательского интерфейса и краткое описание каждого окна системы. Для каждой формы указаны её элементы и назначение.

3.4.2 Основные окна

После перехода на веб-приложение пользователь попадает на начальную страницу (Рисунок 3.1).

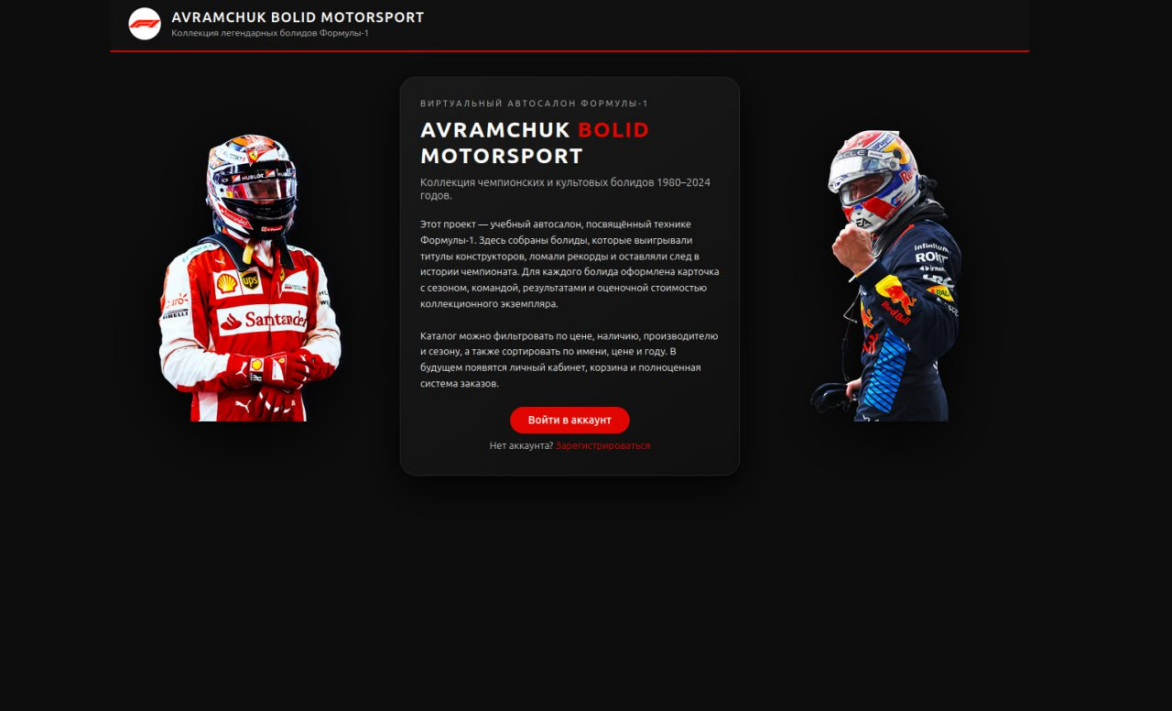


Рисунок 3.1 – Начальная страница

Если у пользователя уже имеется учётная запись, он может перейти на страницу входа в систему (Рисунок 3.2).

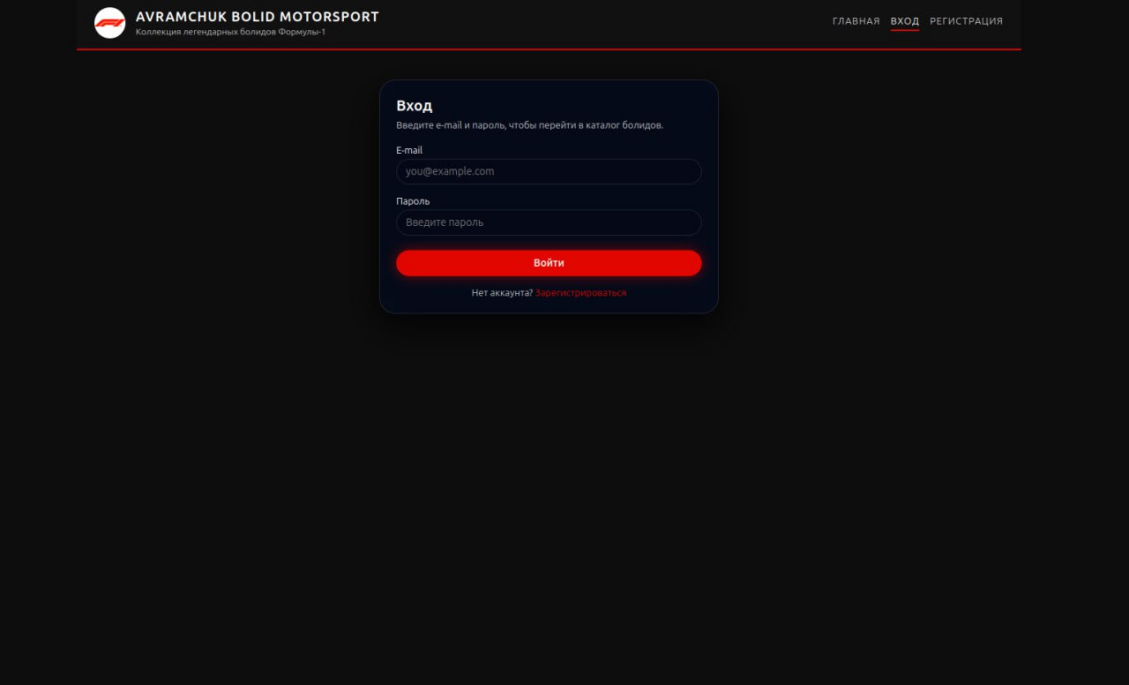


Рисунок 3.2 – Страница авторизации

Если пользователь ранее не проходил регистрацию, ему необходимо перейти на страницу регистрации, где требуется заполнить обязательные поля (Рисунок 3.3).

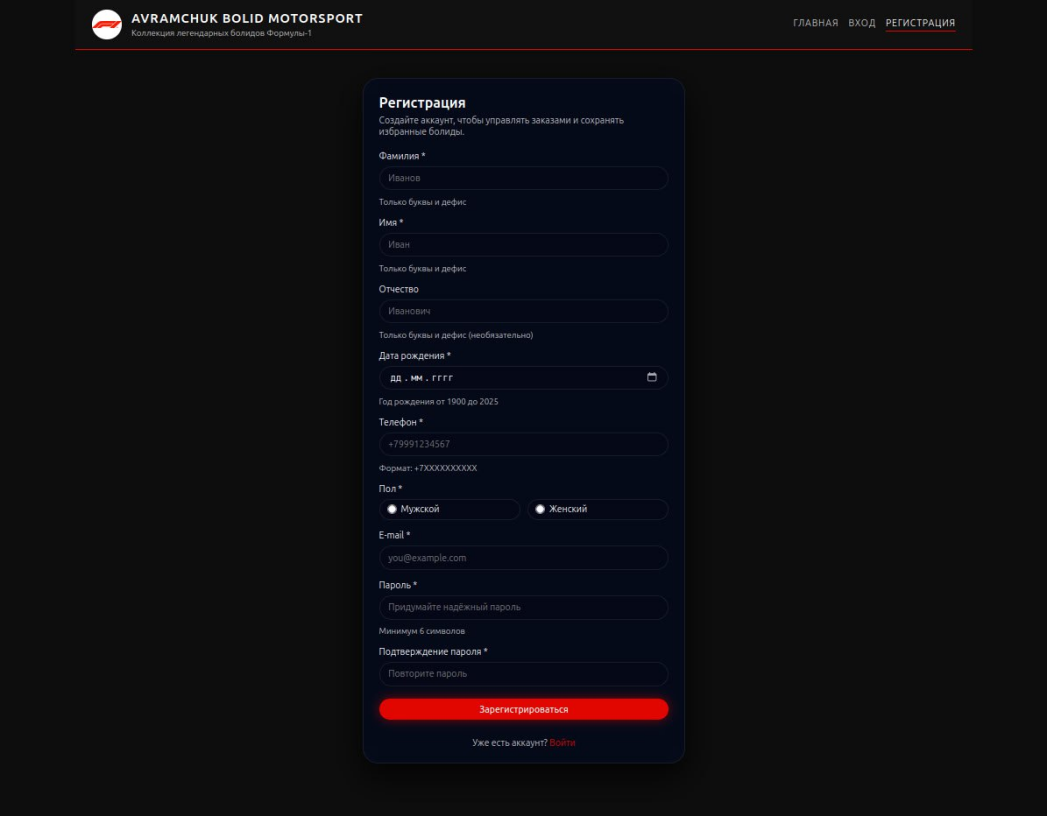


Рисунок 3.3 – Страница регистрации

После успешной авторизации пользователь попадает на страницу каталога болидов (Рисунок 3.4)

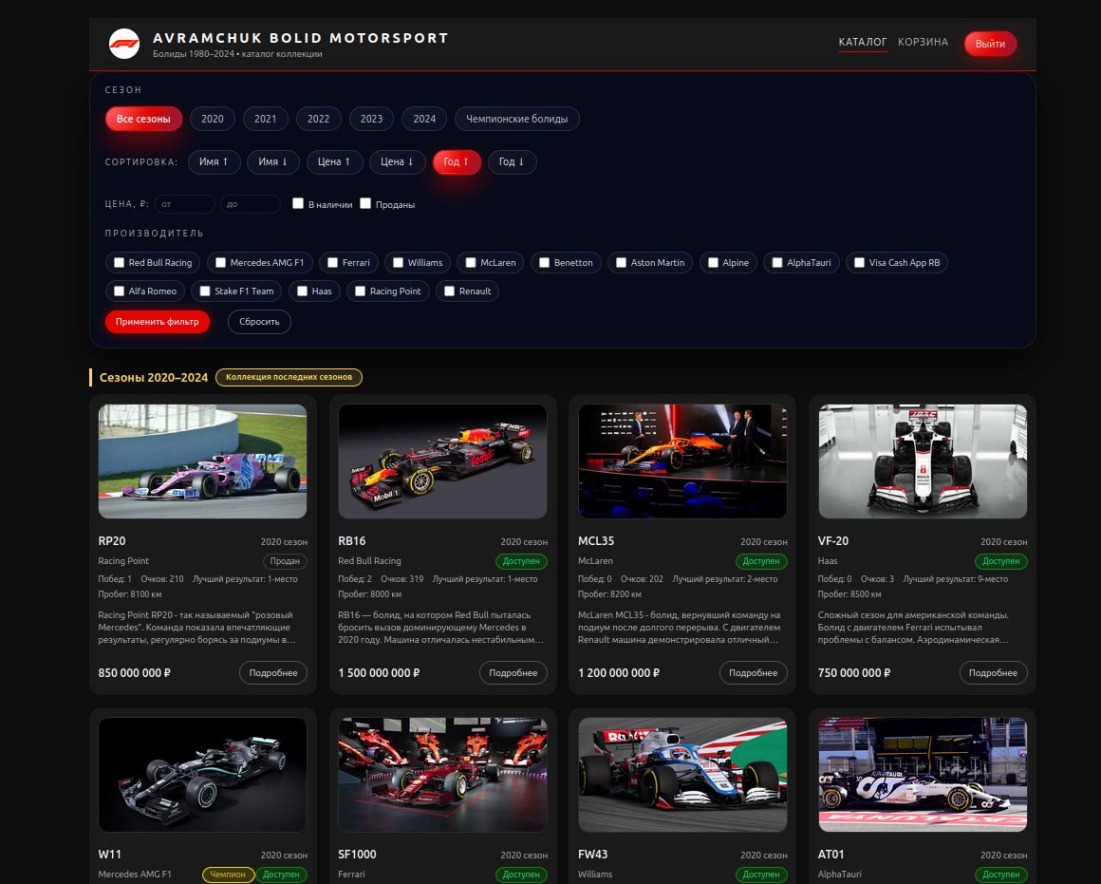


Рисунок 3.4 – Страница каталога

После выбора модели из каталога пользователь нажимает на кнопку «Подробнее» и перенаправляется на страницу с подробной карточкой болида, где болид можно добавить в корзину (Рисунок 3.5).

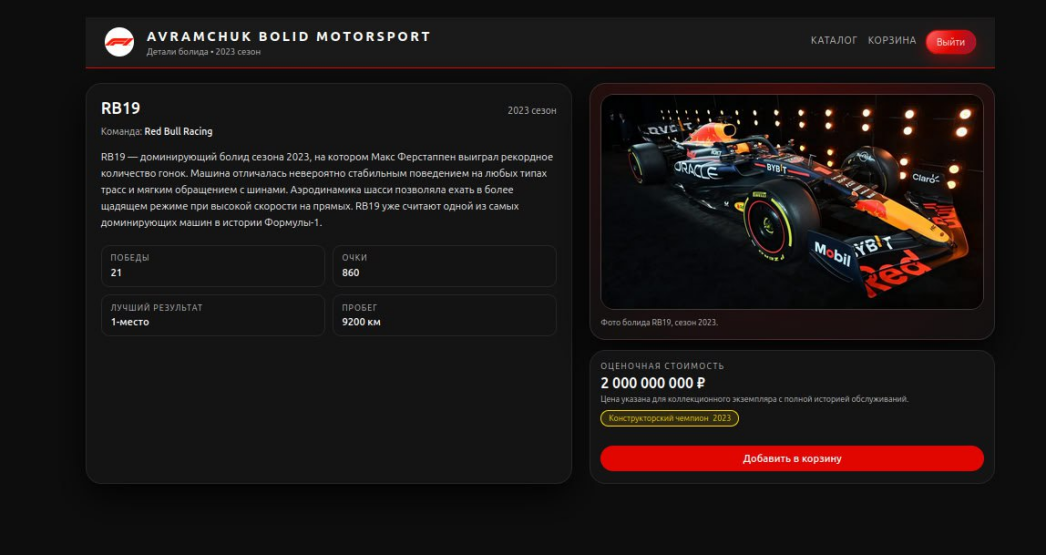


Рисунок 3.5 – Страница карточки болида

При нажатии на кнопку «Корзина» в шапке сайта пользователь переходит к странице корзины, в которой можно оформить заявку (Рисунок 3.6).

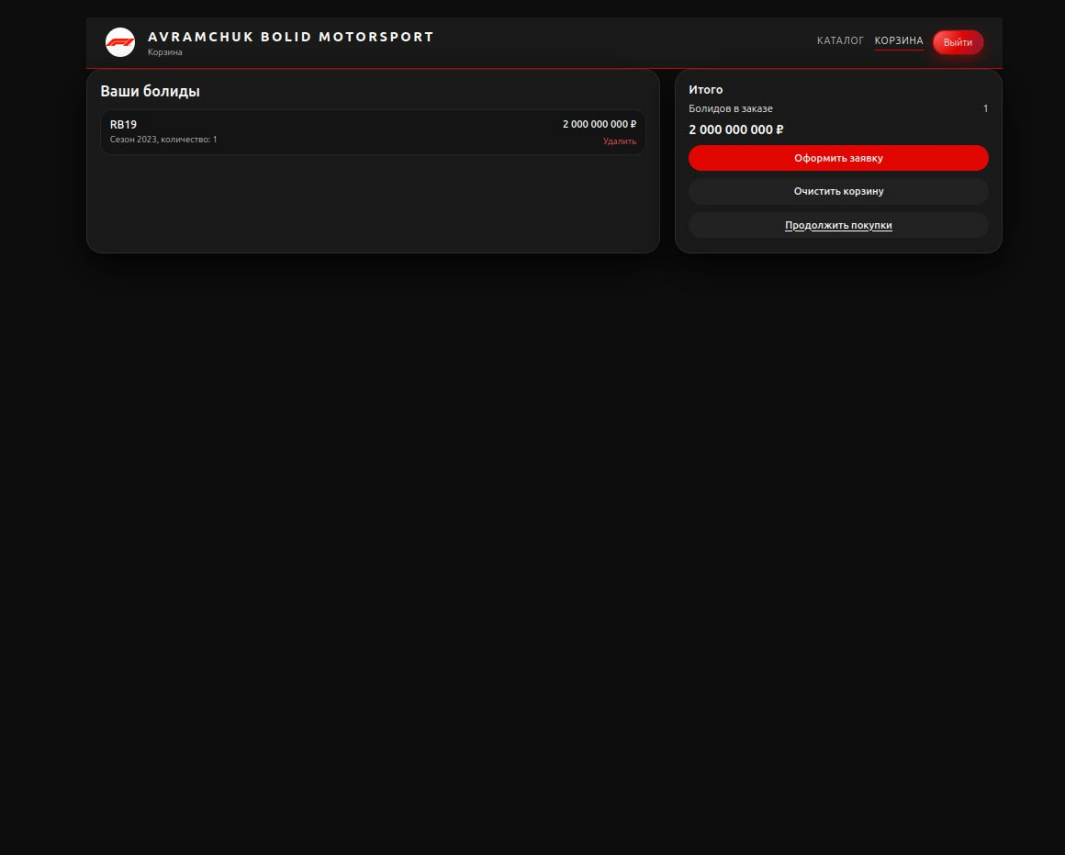


Рисунок 3.6 – Страница корзины

3.5. Сообщение пользователю

При работе с программой могут появиться следующие сообщения, представленные в таблице 3.1.

Таблица 3.1 – Сообщение пользователю

|  |  |
| --- | --- |
| **Текст сообщения программы** | **Ответ пользователя** |
| Неверный email или пароль | Ввести корректные учётные данные |
| Болид не найден | Вернуться в каталог и выбрать актуальный болид |
| Болид добавлен в корзину | Продолжить покупки |
| Этот болид уже находится в корзине | Товар не дублируется |
| Этот болид уже недоступен для покупки. | Добавить болиды перед оформлением |
| **Болид недоступен для покупки.** | Выбрать другой болид |
| **Корзина пуста, оформить заявку нельзя.** | Администратор должен сначала удалить или завершить заявку |
| **Корзина очищена.** | Нет действий |
| **Заявка отправлена. Болиды зарезервированы за вами.** | Ожидать подтверждения от администратора |
| **Пользователь не найден.** | Авторизоваться заново |
| **Не удалось определить текущего пользователя.** | Авторизоваться заново |
| **Болид … уже недоступен для покупки.** | Убрать из корзины и выбрать другой |

3.6 Аварийные ситуации

К аварийным ситуациям относятся:

* отсутствие соединения с интернетом;
* недоступность серверной части;
* сбой подключения к базе данных;
* отсутствие таблиц или нарушение целостности данных;
* попытка оформления заявки на недоступный болид.

В случае возникновения аварийной ситуации работа системы прекращается.

4. РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА

4.1 Общие сведения

Серверная часть программного обеспечения реализована на языке Java с использованием фреймворка Spring Boot и модулей Spring Web, Spring Data JPA, Spring Security и шаблонизатора Thymeleaf. Проект собирается при помощи системы сборки Maven.

В качестве системы управления базами данных используется PostgreSQL (версии 15/16, в зависимости от установленной конфигурации). Для развёртывания сервиса базы данных применяется Docker и Docker Compose, что обеспечивает удобство настройки и воспроизводимость окружения.часть написана на JavaScript с JSX; с использованием React и CSS.

Клиентская часть реализована в виде серверно-генерируемых HTML-страниц на основе шаблонов Thymeleaf и обрабатывается в браузере пользователя. Доступ к функционалу осуществляется через стандартный веб-браузер (Yandex, Chrome, Firefox и др.).

Перед началом работы необходимо убедиться, что на рабочей станции или сервере установлены следующие компоненты:

* Java Development Kit (JDK) не ниже версии 17;
* Maven;
* Docker;
* Docker Compose;
* PostgreSQL.

Инструкция по установке:

* sudo apt update;
* sudo apt install docker.io -y;
* sudo apt install docker-compose -y;
* sudo apt install postgresql postgresql-contrib -y.

Работа с базой данных может осуществляться через консольную утилиту psql либо через графический клиент pgAdmin.

Для контейнеризации и управления зависимостями используется Docker и Docker Compose.

Основные внешние библиотеки серверной части:

* spring-boot-starter-web – обработка HTTP-запросов, реализация MVC-логики;
* spring-boot-starter-thymeleaf – серверная генерация HTML-представлений;
* spring-boot-starter-data-jpa – доступ к базе данных через JPA и Hibernate;
* spring-boot-starter-security – аутентификация и авторизация пользователей;
* postgresql – JDBC-драйвер для взаимодействия с PostgreSQL;
* bcrypt –– библиотека для безопасного хэширования паролей;
* spring-boot-starter-validation – валидация данных форм (аннотации @NotNull, @Size, @Min, @Max и др.);
* spring-boot-starter-test – модуль для тестирования.

4.2 Структура и описание программы

Программное обеспечение представляет собой веб-приложение для управления каталогом болидов Формулы-1 и оформления заявок на их покупку. Программа работает по архитектурной модели «клиент – сервер», где серверная часть реализована на Java (Spring Boot), а клиентская часть формируется на базе серверных HTML-шаблонов Thymeleaf.

4.2.1 Основные пакеты серверной части

Программы включает следующие пакеты серверной части:

– com.coursework.autosalon.web.model – классы доменной модели (сущности JPA)у;

– com.coursework.autosalon.web.repository – интерфейсы репозиториев Spring Data JPA;

– com.coursework.autosalon.web.controller – веб-контроллеры, обрабатывающие HTTP-запросы;

– com.coursework.autosalon.web.config и com.coursework.autosalon.web.security – конфигурация безопасности и интеграция со Spring Security;

– ресурсные файлы src/main/resources/templates – HTML-шаблоны страниц с использованием Thymeleaf.

4.2.2 Основные модули серверной части

Приложение включает следующие основные подсистемы:

1. Подсистема аутентификации и авторизации

– обеспечивает регистрацию пользователей, вход в систему, защиту административных разделов и разграничение прав доступа.

1. Подсистема каталога болидов

– каталог с фильтрами;

– карточка болида;

– поиск по параметрам;

– отображение ограниченных моделей (только для администратора).

1. Подсистема корзины и оформления заявок

– пользователь может добавлять болиды в корзину, просматривать её, удалять позиции и оформлять заявку на покупку.

1. Административная подсистема

– создание, редактирование и удаление болидов;

– просмотр пользователей;

– просмотр и обработка заявок;

– защита от удаления болидов, участвующих в активных заявках.

1. Подсистема доступа к базе данных

– реализована через Spring Data JPA, включает репозитории для всех основных сущностей.

4.2.3 Навигационная схема программы

Все страницы приложения объединяются единой логикой переходов:

1. Главная страница

– доступна неавторизованным пользователям. Содержит ссылки на вход и регистрацию.

1. Страница регистрации

– позволяет создать личный кабинет. После успешного создания — переход на страницу входа.

1. Страница авторизации

– при успешном входе пользователю открывается каталог болидов;

– в случае ошибки отображается сообщение о неправильных данных.

1. Каталог болидов

– фильтры (цена, бренд, сезон, команда, наличие);

– динамический список болидов;

– пункты меню: Каталог, Корзина, Выход.

1. Карточка болидов

– отображает детальные характеристики болида и кнопку «Добавить в корзину».

1. Корзина

– просматривать выбранные модели;

– удалять позиции;

– оформлять заявку.

1. Административная панель

– управление болидами;

– список пользователей;

– список заявок;

– детальная карточка пользователя;

– интерфейс создания и редактирования болида.

4.2.4 Внутренняя архитектура

Приложение построено на основе слоистой архитектуры:

1. Модельный слой (Model)

– User;

– Car;

– Team;

– Driver;

– UserCartItem;

– PurchaseRequest.

1. Слой доступа к данным (Repository)

– UserRepository;

– CarRepository;

– TeamRepository;

– DriverRepository;

– UserCartRepository;

– PurchaseRequestRepository.

1. Сервисный слой (Service)

– обработку заявок;

– управление корзиной;

– проверку возможности удаления болидов;

– регистрацию пользователей;

– фильтрацию и поиск болидов.

1. Веб-слой (Controller)

– AuthController;

– CarController;

– CartController;

– OrderController;

– AdminCarController;

– AdminUserController;

– AdminRequestController.

1. Слой безопасности (Security)

– защиту административных маршрутовсписок заявок;

– BCrypt хэширование паролей;

– механизм авторизации через форму;

– автоматический выход по ссылке /logout.

1. Слой представления (View)

– HTML-страницы расположены в src/main/resources/templates

и генерируются сервером через шаблоны Thymeleaf.

4.2.5 Логика работы приложения

1. Пользователь проходит регистрацию, далее данные записываются в таблицу users.

2. Пользователь авторизуется, затем Spring Security создаёт сессию.

3. Открывается каталог, после чего выполняется запрос к таблице cars, фильтры применяются через JPA-запросы.

4. При добавлении болида в корзину создаётся запись в user\_cart\_items.

5. При оформлении заявки данные фиксируются в purchase\_requests.

6. Администратор может обрабатывать заявки и управлять каталогом.

7. Удаление болида возможно только если он не участвует в активных заявках (реализовано через бизнес-логику в контроллере).

4.2.6 Клиент-серверная архитектура и взаимодействие подсистем

Разрабатываемое веб-приложение функционирует в архитектуре «клиент – сервер», где серверная часть отвечает за обработку бизнес-логики, взаимодействие с базой данных и обеспечение безопасности, а клиентская часть отвечает за отображение данных пользователю и обработку его действий.

1. Серверная часть (Backend)

Сервер реализован на Java с использованием Spring Boot и включает следующие компоненты:

1.1 Контроллеры

Контроллеры принимают HTTP-запросы клиента, выполняют обработку данных и формируют ответы или HTML-страницы.

Основные контроллеры:

– AuthController – регистрация, авторизация;

– CarController – каталог болидов, карточка болида;

– CartController – работа с корзиной;

– AdminCarController – добавление, редактирование и удаление болидов;

– AdminUserController – управление пользователями;

– AdminRequestController – управление заявками.

1.2 Сервисный слой

Отвечает за бизнес-логику:

– проверка возможности удаления болида;

– формирование заявки на покупку;

– добавление болида в корзину;

– фильтрация каталога;

– обработка статусов заявок;

– регистрация пользователя (валидация, хэширование пароля).

1.3 Репозитории

Модуль доступа к данным, реализованный через Spring Data JPA:

– UserRepository;

– CarUserRepository;

– TeamUserRepository;

– UserCartItemUserRepository;

– PurchaseRequestUserRepository.

Они предоставляют CRUD-операции, а также выполняют SQL-запросы через Hibernate.

1.4 Слой безопасности

Отвечает за:

– разграничение доступа (USER, ADMIN);

– обработку сессий;

– защиту административных маршрутов;

– хэширование паролей через BCrypt.

1.5 Шаблоны HTML

Страницы генерируются на сервере с помощью Thymeleaf:

– каталог;

– карточка болида;

– корзина;

– редактирование болида

– управление пользователями.

2. Клиентская часть (Frontend)

Клиентская часть представляет собой набор серверно-генерируемых HTML-страниц, которые рендерятся в браузере пользователя.

2.1 Основные функции клиента

– отображение страниц каталога;

– отправка запросов к серверу через формы и кнопки;

– отображение ошибок валидации;

– динамическое изменение интерфейса в зависимости от роли пользователя;

– отображение содержимого корзины;

– защита от повторных добавлений болида в корзину;

– отображение административных функций при наличии прав.

2.2 Валидация на стороне клиента

Выполняется HTML5-валидацией и простыми JS-проверками:

– корректность формата email;

– проверка заполненности обязательных полей;

– запрет отправки формы регистрации с ошибками;

– проверка отсутствия пустых значений.

3. Поток данных

Ниже – описание того, как проходят данные через систему.

3.1 Регистрация

1. Форма клиента отправляет данные на сервер.

2. Backend валидирует (email, пароль).

3. Пароль хэшируется (BCrypt).

4. Данные сохраняются в таблице users.

5. Клиент перенаправляется на /login.

3.2 Добавление болида в корзину

1. Клиент отправляет запрос в /cart/add.

2. Backend проверяет:

– существует ли болид;

– доступен ли он;

– не добавлен ли он повторно.

3. Создаётся запись в user\_cart\_items.

4. Клиент получает обновлённую корзину.

3.3 Оформление заявки

1. Клиент отправляет POST /checkout.

2. Backend:

– фиксирует цену и название болида (snapshot!);

– создаёт запись в purchase\_requests;

– обновляет статус болида

3. Клиенту отображается подтверждение.

Связь модулей программы между собой представлена на рисунке 4.1.

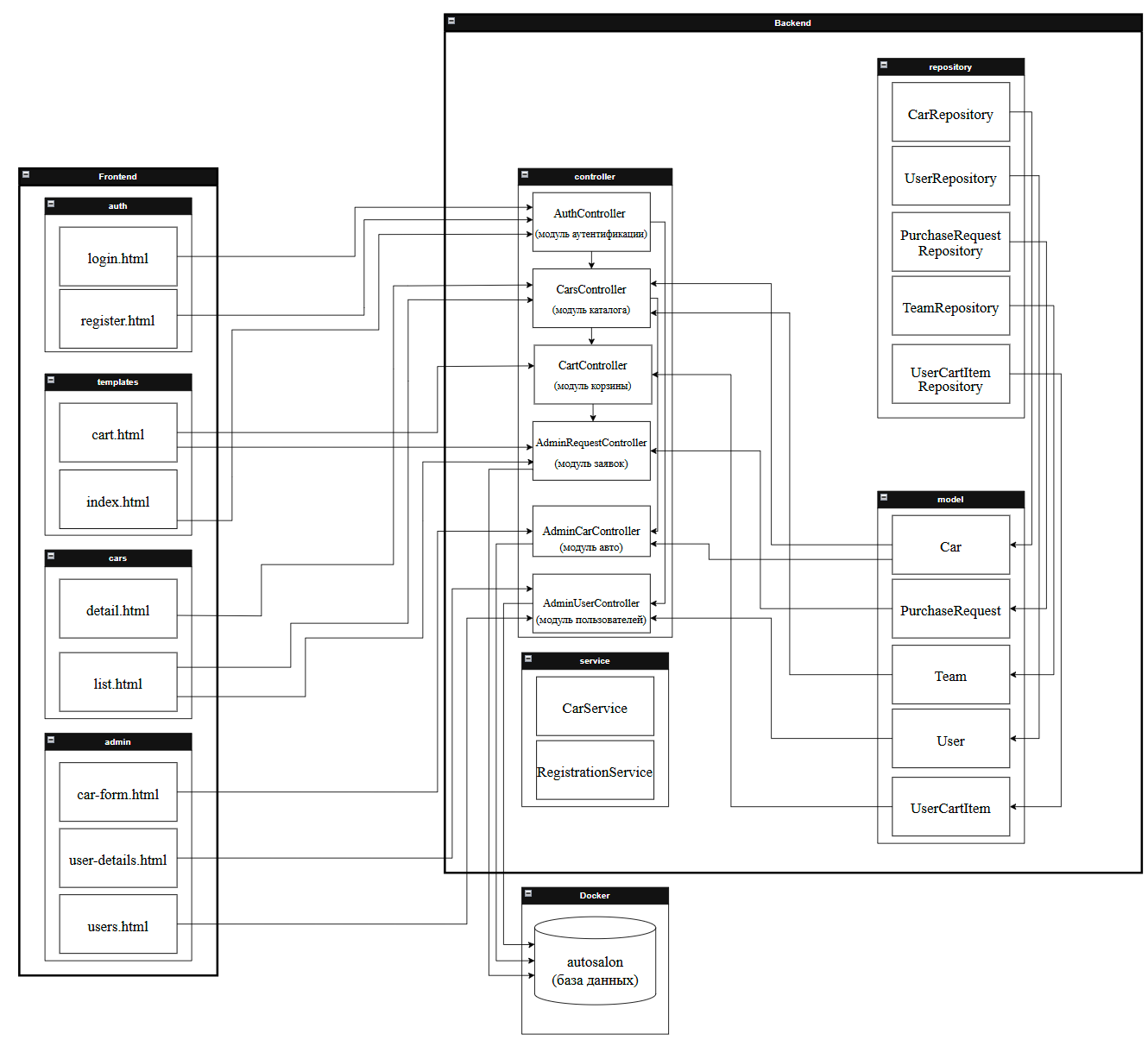


Рисунок 4.1 – UML-диаграмма связи модулей программы

4.3 Контейнеризация

В проекте предусмотрена контейнеризация сервиса базы данных PostgreSQL. Для этого используется файл docker-compose.yml в корне подпроекта autosalon-web.

В docker-compose.yml описан сервис:

контейнер с образом PostgreSQL, в котором:

* создаётся база данных приложения;
* настраиваются параметры подключения (имя базы, пользователь, пароль);
* при необходимости могут применяться SQL-скрипты инициализации.

Для запуска службы базы данных в контейнере необходимо выполнить команду из каталога, где расположен docker-compose.yml: «docker-compose up -d»

После запуска контейнера приложение Spring Boot подключается к БД PostgreSQL по параметрам, указанным в файле application.properties или application.yml:

* URL подключения (JDBC);
* имя пользователя базы данных;
* пароль;
* стратегия создания / обновления схемы (spring.jpa.hibernate.ddl-auto).

Серверная часть приложения (Spring Boot) запускается вне контейнера стандартной командой Maven: «mvn spring-boot:run»

либо через скомпилированный JAR: «java -jar autosalon-web-0.0.1-SNAPSHOT.jar»

Таким образом, контейнеризация базы данных упрощает развёртывание и снижает зависимость проекта от локальной конфигурации СУБД.

4.4 Описание базы данных

Приложение использует реляционную базу данных PostgreSQL. Структура базы данных соответствует сущностям доменной модели и обеспечивает хранение информации о пользователях, болидах, командах, пилотах, позициях корзины и заявках на покупку.

4.4.1 Сущности

Перечень основных сущностей базы данных представлен в таблице 4.1.

Таблица 4.1 – Сущности базы данных

|  |  |
| --- | --- |
| Сущность | Определение |
| users | Данные зарегистрированных пользователей. Хранит учетные данные, ФИО, контактную информацию и роль |
| roles | Справочник ролей пользователей. Может использоваться для разграничения прав доступа |
| teams | Справочник команд Формулы 1. Содержит название команды и страну |
| **drivers** | Справочник пилотов. Хранит ФИО, страну и количество чемпионских титулов |
| **cars** | Основная сущность системы. Хранит информацию о болидах, их стоимости, статистике и принадлежности |
| **user\_cart\_items** | Позиции пользовательских корзин. Связь между пользователем и выбранными им болидами |
| **orders** | Заказы пользователей, оформленные на основе содержимого корзины |

Продолжение таблицы 4.1

|  |  |
| --- | --- |
| **1** | 2 |
| **orders\_items** | Строки заказов. Связывают заказ с конкретными болидами и их ценой на момент оформления |
| **purchase\_requests** | Заявки на покупку болида. Фиксируют привязку пользователю, болиду и цену на момент создания заявки |

4.4.2 Атрибуты сущности users

Таблицы 4.2 – атрибуты users

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | | Обязательно к заполнению | |
| id | BIGINT | Простой | Целое число | | Да | |
| email | character varying | Простой | Электронная почта | | Да | |
| password\_hash | character varying | Простой | Латиница, спецсимволы | | Да | |
| last\_name | character varying | Простой | Кириллица, латиница | | Да | |
| first\_name | character varying | Простой | Кириллица, латиница | | Да | |
| middle\_name | character varying | Простой | Кириллица, латиница | Нет | |
| birth\_date | VARCHAR | Простой | Дата | Да | |

Продолжение таблицы 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| phone | VARCHAR | Простой | Телефон | Да |
| role | VARCHAR | Простой | Латиница | Да |
| gender | VARCHAR | Простой | true/false | Да |
| enabled | BOOLEAN | Простой | true/false | Да |

4.4.3 Атрибуты сущности cars

Таблица 4.3 – атрибуты cars

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| id | BIGSERIAL | Простой | Целое число | Да |
| available | BOOLEAN | Простой | true/false | Да |
| brand | VARCHAR | Простой | Кириллица, латиница | Да |
| milleage\_km | INTEGER | Простой | Целое число | Да |
| price | INTEGER | Простой | Целое число | Да |
| team\_id | BIGINT | Простой | Целое число | Да |

Продолжение таблицы 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| season\_year | INTEGER | Простой | Целое число | Да |
| description | TEXT | Простой | TEXT | Нет |
| total\_points | INTEGER | Простой | Целое число | Нет |
| best\_driver\_name | VARCHAR | Простой | Латиница, кириллица | Нет |
| best\_driver\_titles | INTEGER | Простой | Целое число | Нет |
| best\_result | VARCHAR | Простой | TEXT | Нет |
| wins | INTEGER | Простой | Целое число | Нет |
| image\_url | VARCHAR | Простой | URL | Нет |
| is\_constructors\_champion | BOOLEAN | Простой | true/false | Да |
| best\_driver\_id | INTEGER | Простой | Целое число | Да |
| hidden | BOOLEAN | Простой | true/false | Да |

4.4.4 Атрибуты сущности teams

Таблицы 4.4 – атрибуты teams

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| id | BIGSERIAL | Простой | Целое число | Да |
| country | VARCHAR | Простой | Латиница, кириллица | Да |
| name | VARCHAR | Простой | Латиница, | Да |
| name | VARCHAR | Простой | Латиница, кириллица | Да |

4.4.5 Атрибуты сущности drivers

Таблицы 4.5 – атрибуты drivers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| id | BIGSERIAL | Простой | Целое число | Да |
| full\_name | VARCHAR | Простой | Латиница, кириллица | Да |
| country | VARCHAR | Простой | Латиница, кириллица | Да |
| championships\_count | INTEGER | Простой | Целое число | Да |
| is\_active | BOOLEAN | Простой | true/false | Да |

4.4.6 Атрибуты сущности purchase\_requests

Таблица 4.6 – атрибуты purchase\_requests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| id | BIGSERIAL | Простой | Целое число | Да |
| user\_id | BIGINT | Простой | Целое число | Да |
| car\_id | BIGINT | Простой | Целое число | Да |
| car\_brand\_snapshot | VARCHAR | Простой | Латиница, кириллица | Да |
| price\_snapshot | NUMERIC | Простой | Целое число | Да |
| created\_at | TIMESTAMPZ | Составной | Дата и время | Да |
| processed | BOOLEAN | Простой | true/false | Да |

4.4.7 Атрибуты сущности user\_cart\_items

Таблицы 4.7 – атрибуты user\_cart\_items

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| id | BIGSERIAL | Простой | Целое число | Да |
| user\_id | BIGINT | Простой | Целое число | Да |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |

Продолжение таблицы 4.7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| car\_id | BIGINT | Простой | Целое число | Да |
| created\_at | TIMESTAMPZ | Составной | Дата и время | Да |

4.4.8 Атрибуты сущности user\_cart\_items

Таблицы 4.8 – атрибуты user\_cart\_items

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип | Вид | Домен | Обязательно к заполнению |
| id | BIGSERIAL | Простой | Целое число | Да |
| name | VARCHAR | Простой | Латиница, кириллица | Да |

Связь таблиц базы данных представлена на рисунке 4.2.

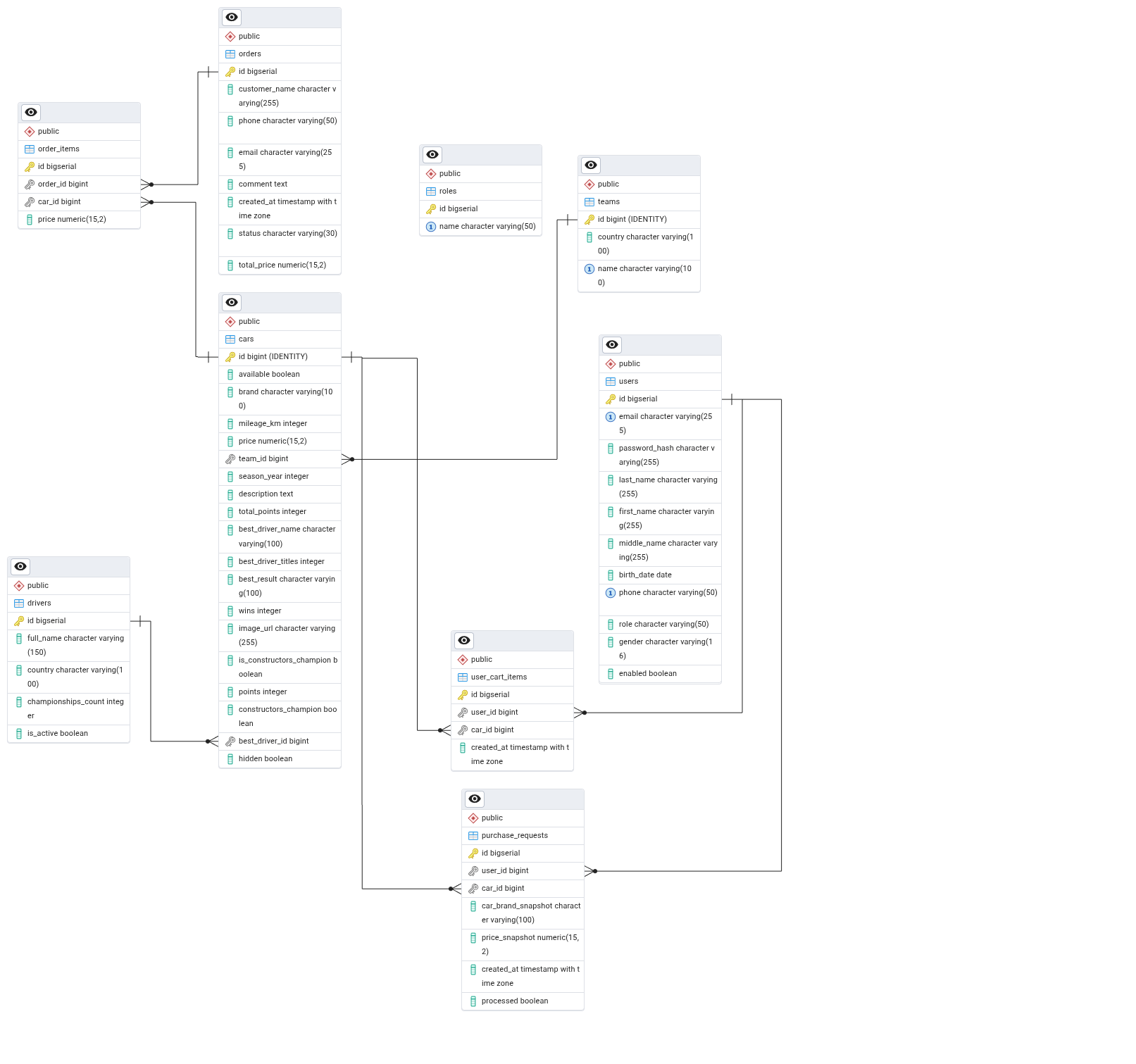


Рисунок 4.2 – ERD-схема связи между сущностями

4.5 Место работы в комплексной защите объекта информации

Комплексная защита веб-приложения автосалона болидов Формулы-1 включает в себя меры по защите данных в базе данных, обеспечение безопасной передачи информации, защиту серверной части, а также контроль входных данных на клиентской стороне. Реализованные механизмы направлены на предотвращение несанкционированного доступа к персональным данным пользователей, обеспечивают целостность заказов и корректность работы всех модулей системы.

4.5.1 Защита данных в базе данных

1. Параметризованные SQL-запросы (JPA/Hibernate или JDBC):

* все операции записи и чтения данных выполняются через ORM-слой или безопасные параметризованные запросы;
* использование параметров исключает возможность SQL-инъекции при передаче идентификаторов болидов, пользователей, заказов и содержимого корзины.

2. Ролевое разграничение доступа:

Таблица users содержит атрибут role (например, USER, ADMIN), что позволяет:

* ограничить доступ к административным страницам;
* запретить обычным пользователям оформлять заявки за других пользователей;
* разрешить только ADMIN выполнять операции модерации заявок.

3 Целостность данных через связи FK:

* удаление болида невозможно, пока он используется в таблицах purchase\_requests, user\_cart\_items или order\_items;
* это предотвращает появление «битых» заявок.

4 Хранение паролей в виде хэшей:

* в таблице users хранится только password\_hash, создаваемый алгоритмом BCrypt;
* пароли в открытом виде не сохраняются и не передаются.

4.5.2 Защита данных при передаче

1. Транспортировка данных по HTTP/HTTPS:

* при развёртывании в промышленной среде рекомендуемое использование HTTPS с TLS-шифрованием;
* это исключает перехват данных авторизации и личной информации пользователя.

2. Проверка принадлежности операций пользователю:

* корзина, заявки и заказы привязаны к user\_id;
* сервер проверяет, что пользователь может управлять только своей корзиной и своими заявками.

4.5.3 Защита серверной части

1. Контейнеризация Docker:

* PostgreSQL и серверная часть приложения развёрнуты в отдельных контейнерах;
* изоляция контейнеров снижает риск влияния внешнего ПО на работоспособность базы данных или сервера.

2. Ограничение доступа к БД:

* в docker-compose.yml БД доступна только по локальному порту 543;
* внешние подключения запрещены, если не требуется pgAdmin;
* пароли БД задаются через переменные среды и не хранятся в исходниках.

3. Обработка ошибок и валидация на бэкенде:

* пароли БД задаются через переменные среды и не хранятся в исходниках;
* при ошибках сервер возвращает информативные сообщения.

4. Журналирование:

* логи фиксируют все обращения к защищённым ресурсам;
* при возможных попытках взлома можно определить источник угрозы.

4.5.4 Защита клиентской части

1. Валидация пользовательских данных:

* проверяется корректность полей регистрации: email, пароль, фамилия, имя, телефон, дата рождения;
* ошибки выводятся пользователю без передачи неверных данных на сервер.

2. Исключение дублирования элементов в корзине:

* клиентская часть проверяет, что болид не добавляется повторно;
* это уменьшает нагрузку на сервер и БД..

3. Скрытие элементов управления, недоступных пользователю:

* если пользователь не ADMIN, то скрыты возможности управления заявками, добавления болида и редактированием болидов.

4.5.5 Обеспечение целостности и непрерывности работы:

* система корректно обрабатывает отключение сервера или повреждение соединения;
* при потере связи отображается сообщение о недоступности БД;
* содержимое корзины хранится в таблице user\_cart\_items, поэтому не исчезает при обновлении страницы или перезапуске сервера.

5. КОНТРОЛЬНЫЙ ПРИМЕР

После обращения к адресу размещения веб-приложения открывается главная страница, содержащая краткую информацию о сервисе, логотип, а также элементы навигации «Вход» и «Регистрация» (Рисунок 5.1).



Рисунок 5.1 – Начальная страница веб-приложения

Пользователь выбирает пункт «Регистрация». Открывается форма создания нового аккаунта, содержащая поля для ввода персональных данных: фамилии, имени, отчества, даты рождения, телефона, пола, адреса электронной почты и пароля. При некорректном заполнении хотя бы одного поля система отображает уведомление о нарушении требований валидации (Рисунок 5.2).

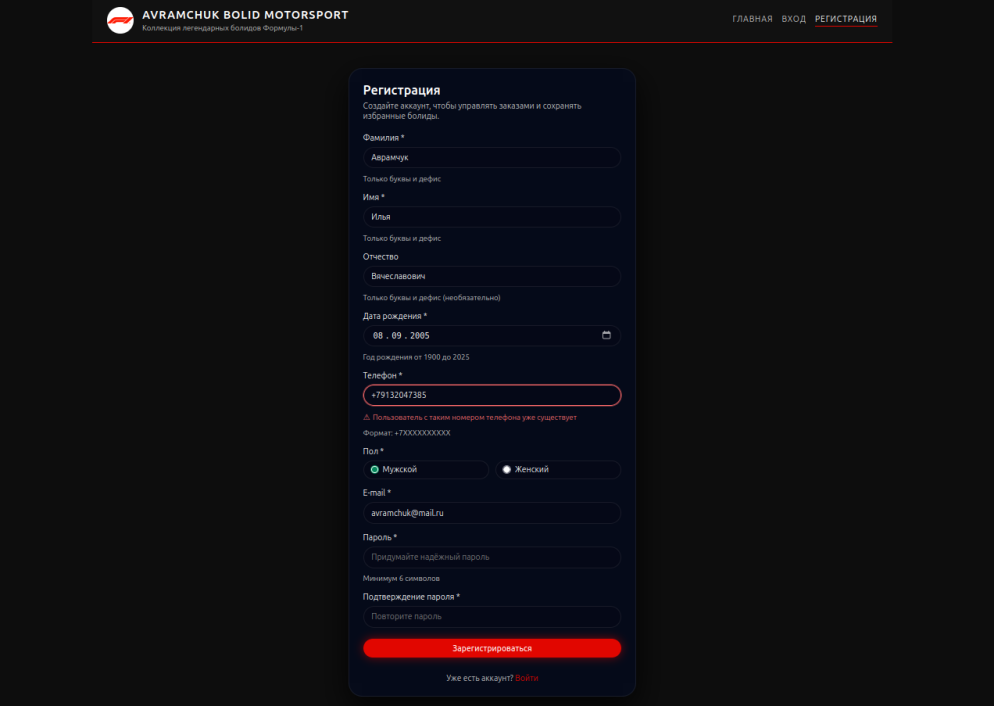


Рисунок 5.2 – Окно регистрации

После регистрации пользователь выбирает пункт «Вход» и вводит свои учетные данные для авторизации. При корректном вводе система перенаправляет его в каталог болидов (Рисунок 5.3).

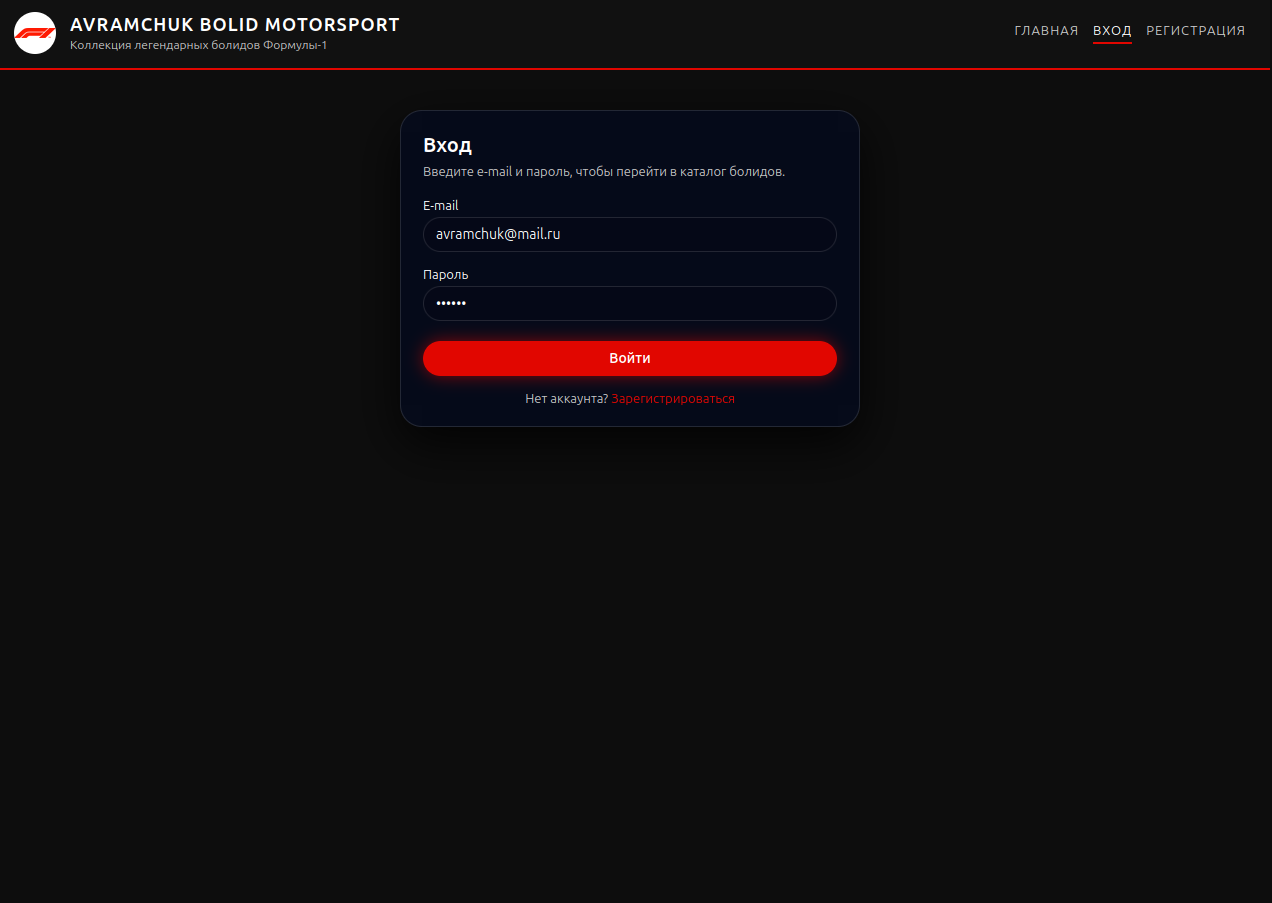


Рисунок 5.3 – Окно входа

После успешной авторизации отображается каталог доступных болидов с возможностью фильтрации и сортировки (Рисунок 5.4).

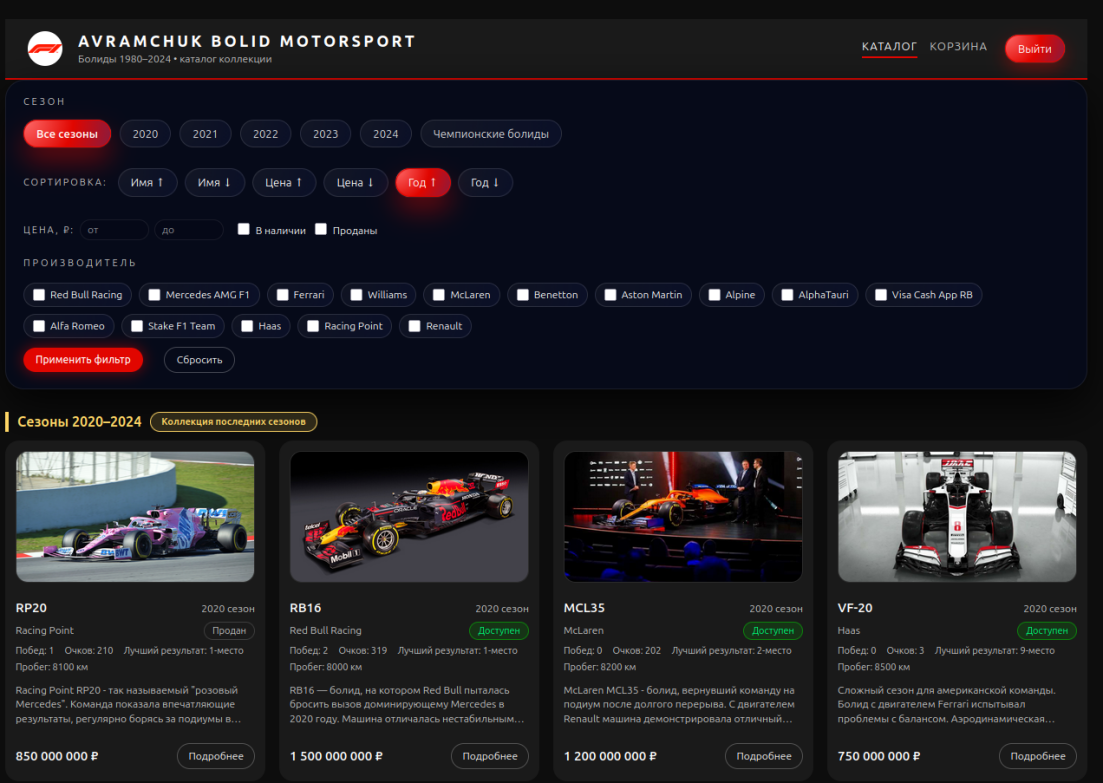


Рисунок 5.4 – Страница каталога болидов

Пользователь выбирает необходимые параметры фильтрации: год, команду, цену или статус наличия. После применения фильтров отображается соответствующий список болидов (Рисунок 5.5).

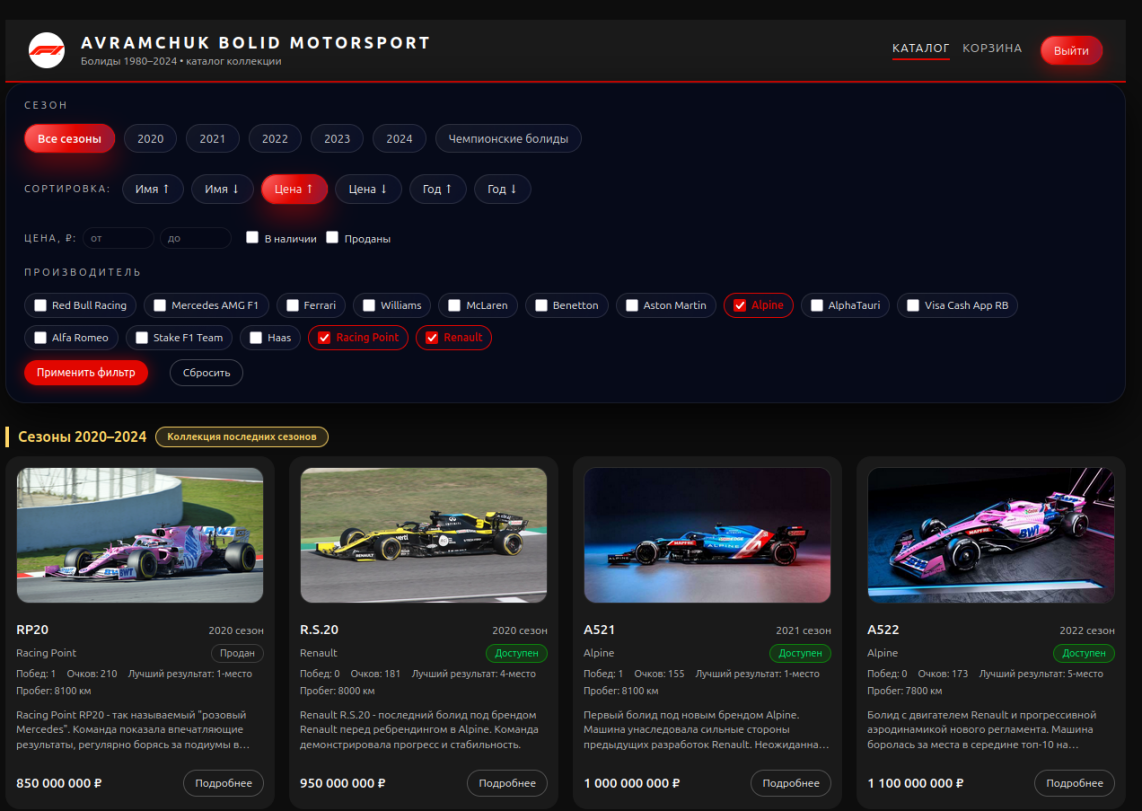


Рисунок 5.5 – Применение фильтров в каталоге

При выборе кнопки «Подробнее» открывается страница карточки болида. Здесь отображаются технические характеристики, стоимость и изображение. Пользователь нажимает кнопку «Добавить в корзину» (Рисунок 5.6).

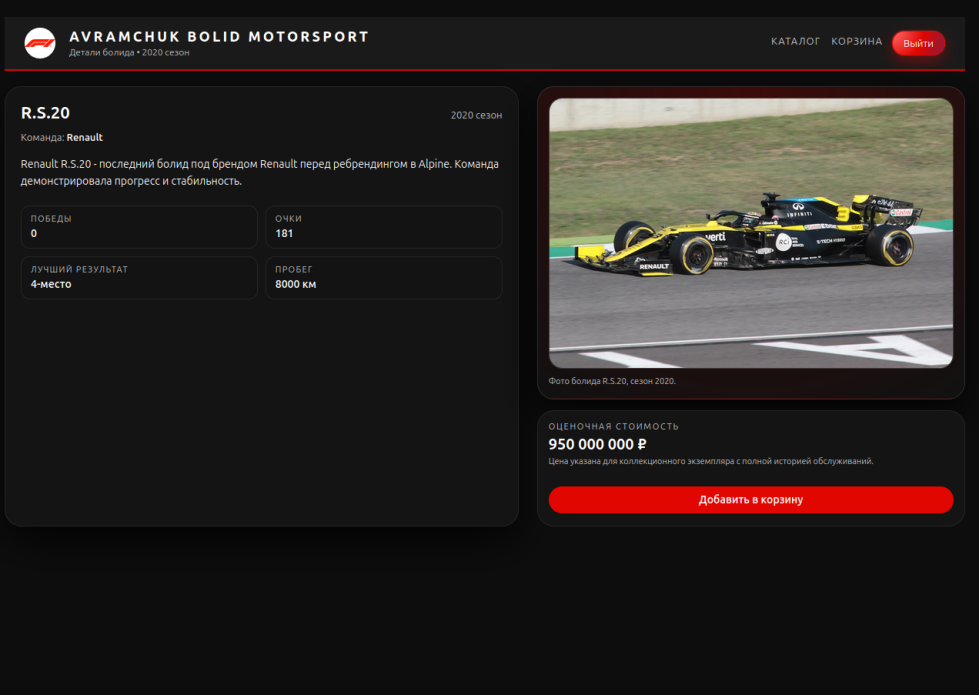


Рисунок 5.6 – Карточка болида

Пользователь возвращается в каталог, выбирает иной болид и аналогично добавляет его в корзину (Рисунок 5.7).

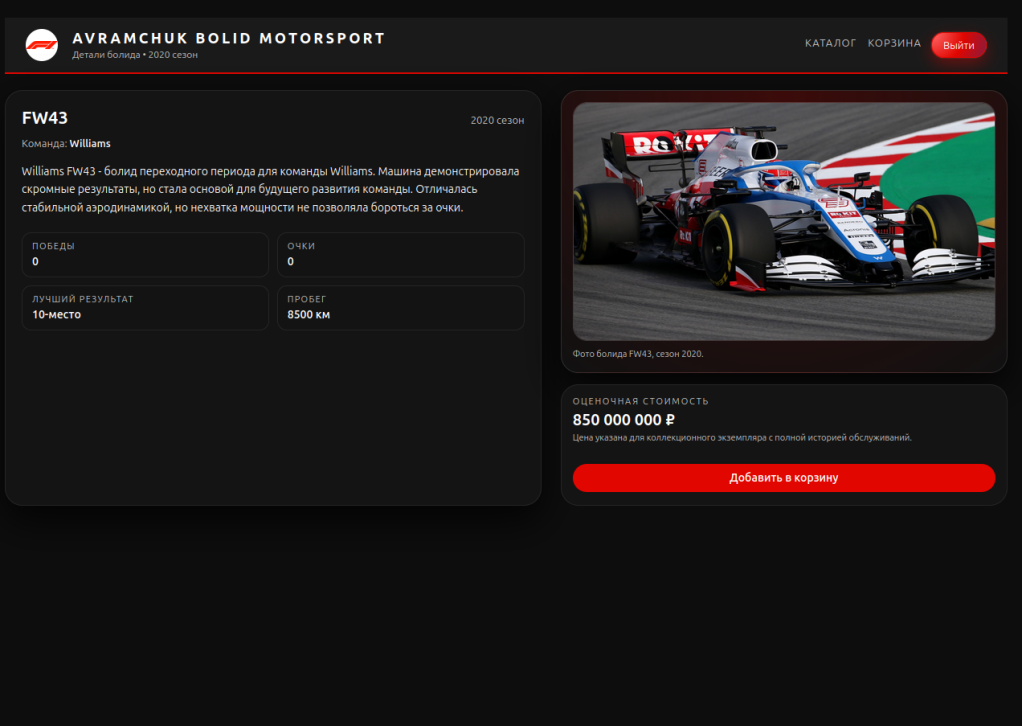


Рисунок 5.7 – Добавление второго болида

Пользователь переходит в раздел «Корзина», где отображается список выбранных болидов, их стоимость и кнопка для удаления позиции (Рисунок 5.8).

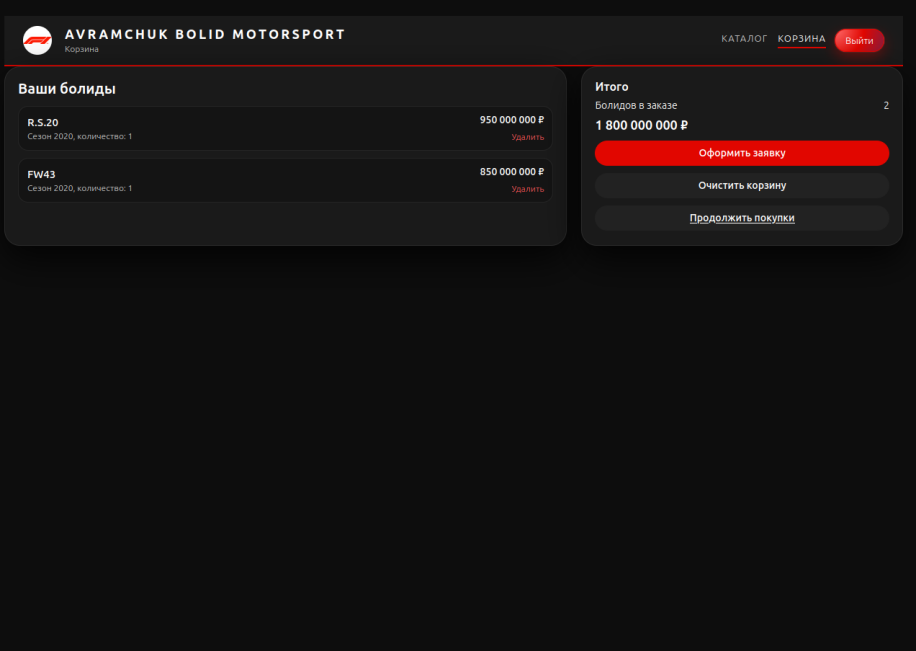


Рисунок 5.8 – Страница корзины

Пользователь выбирает позицию и нажимает кнопку «Удалить». Корзина обновляется, оставляя одну позицию (Рисунок 5.9).

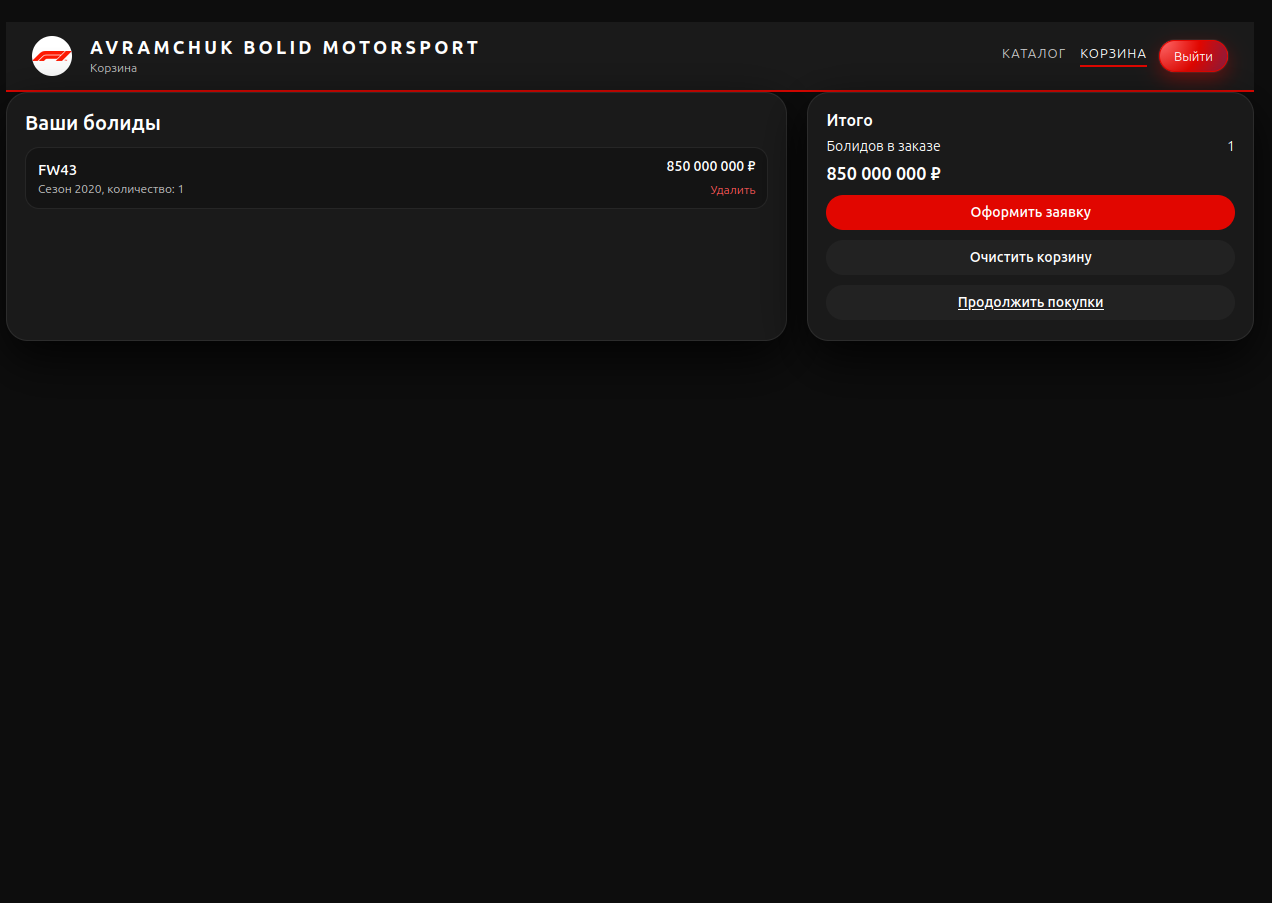


Рисунок 5.9 – Удаление болида из корзины

Пользователь нажимает кнопку «Оформить заявку» и переходит на страницу подтверждения заказа. На экране отображается итоговый список болидов и итоговая стоимость (Рисунок 5.10).

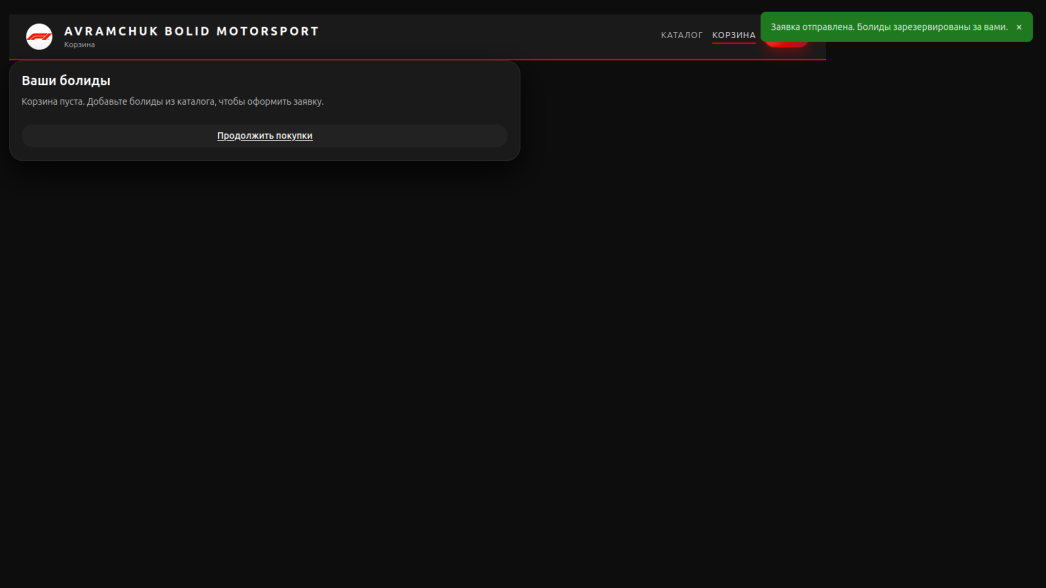
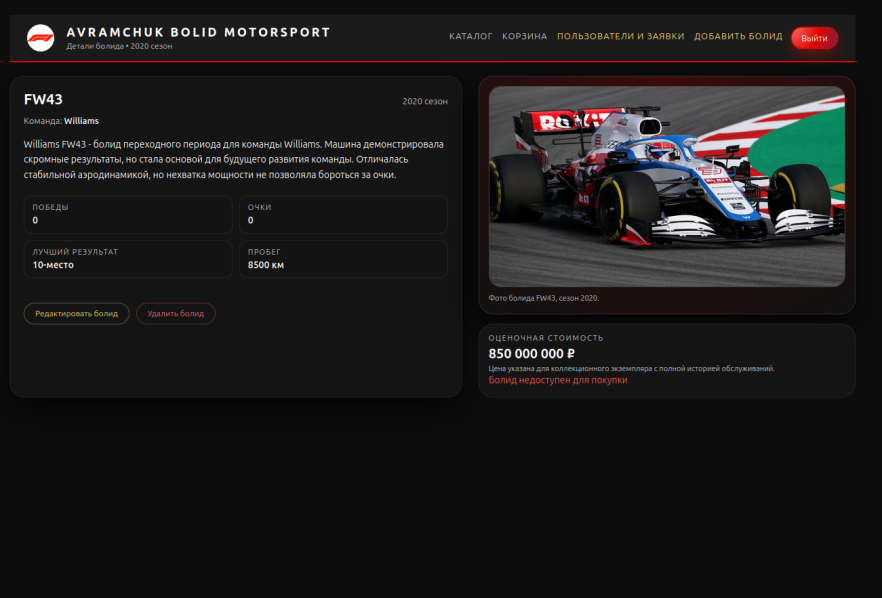


Рисунок 5.10 – Окно оформления заявки

После оформления заявки пользователь открывает страницу болида и видит, что его статус изменён на «Продан» или «Недоступен». Тем самым подтверждается корректное выполнение бизнес-логики (Рисунок 5.11).

  
Рисунок 5.11 – Проверка изменения статуса болида

Администратор авторизуется в системе и получает доступ к расширенному каталогу, содержащему как открытые, так и скрытые болиды (Рисунок 5.12).

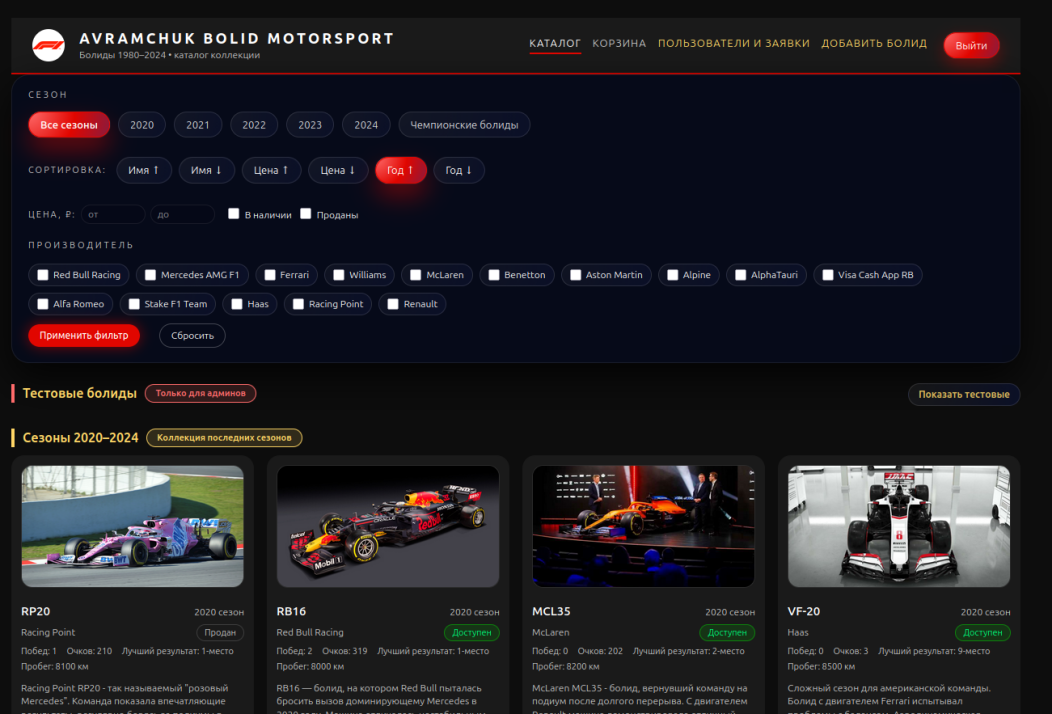


Рисунок 5.12 – Каталог болидов (режим администратора)

Администратор включает отображение скрытых моделей, используемых для внутренней обработки или модерации (Рисунок 5.13).

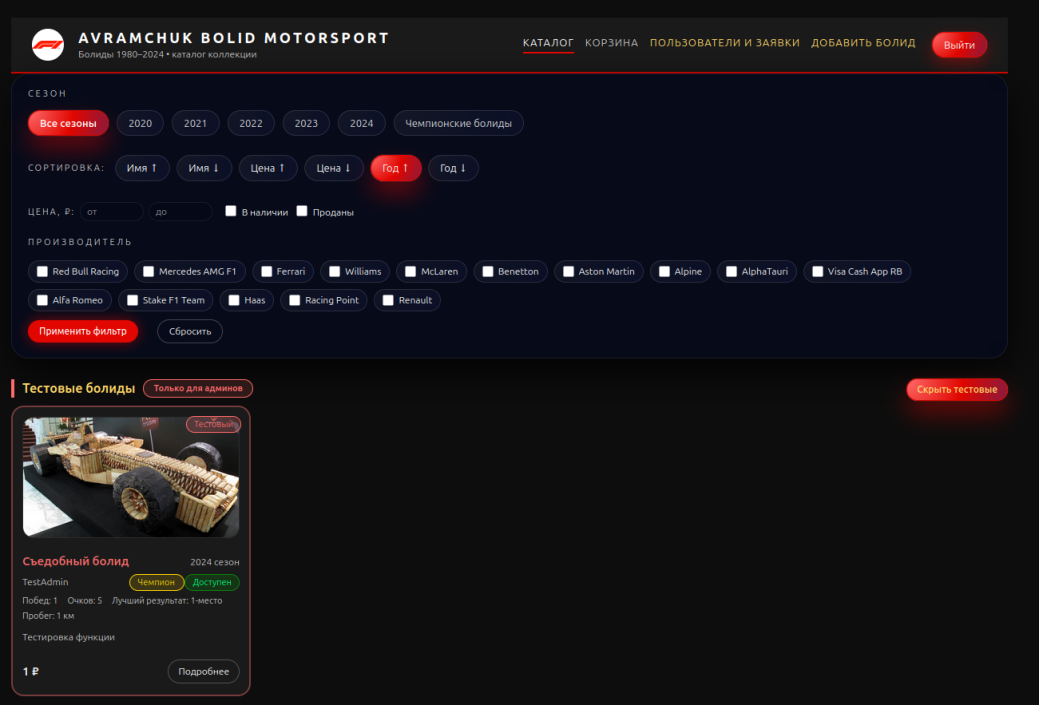
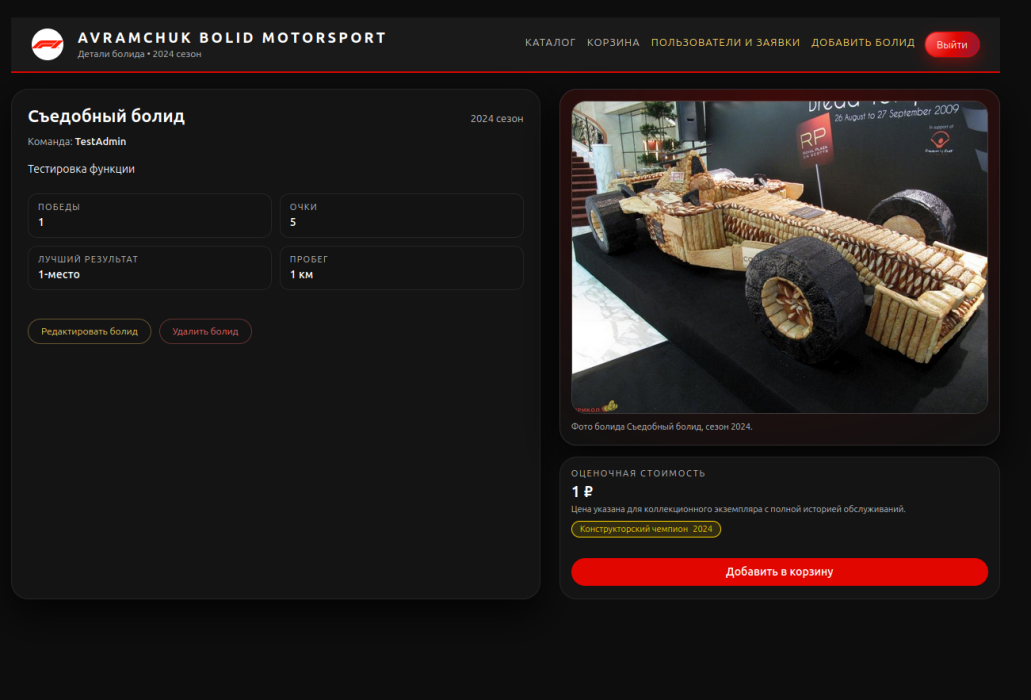


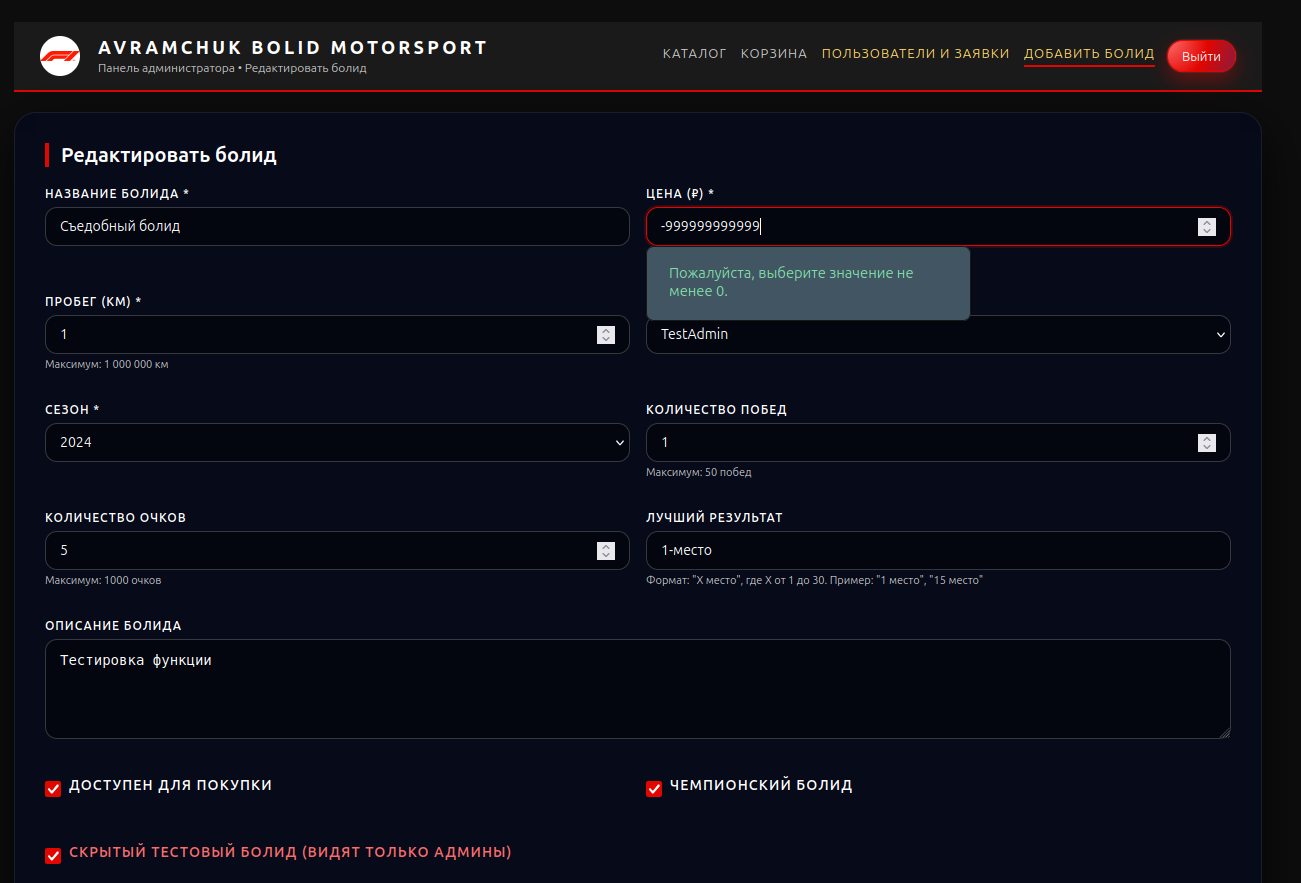
Рисунок 5.13 – Скрытые болиды

На странице карточки появляется функционал редактирования и удаления болида. Администратор открывает карточку и наблюдает две дополнительные кнопки (Рисунок 5.14).



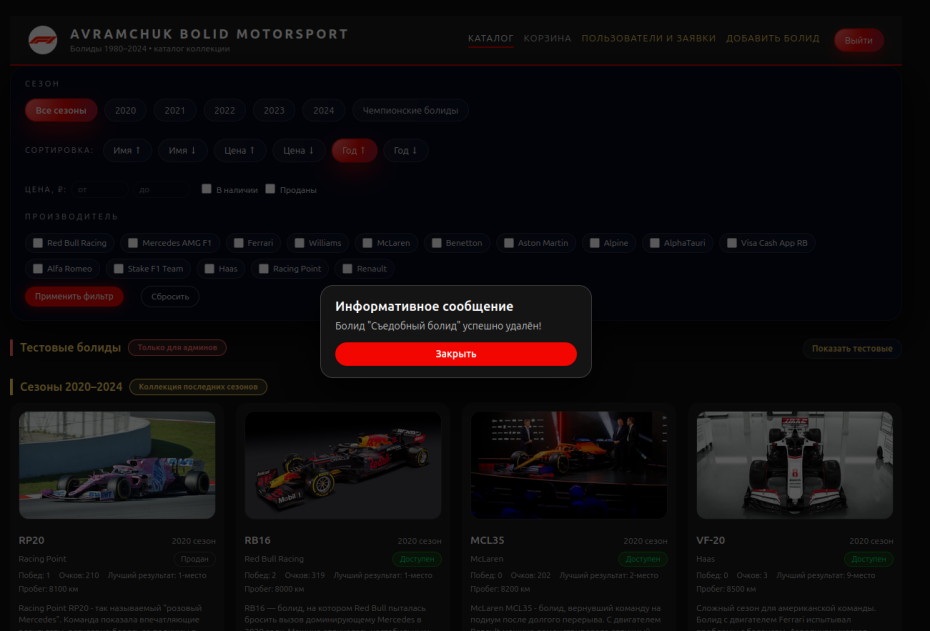
**Рисунок 5.14 – Страница болида с административными функциями**

При нажатии на кнопку «Редактировать» открывается форма изменения характеристик. Администратор может обновить цену, описание, год сезона или изображение (Рисунок 5.15).



**Рисунок 5.15 – Окно редактирования болида**

Администратор нажимает кнопку «Удалить». Система проверяет наличие активных заявок. Если болид не участвует в оформленной заявке, то удаление происходит успешно с всплывающим сообщением (Рисунок 5.16).



**Рисунок 5.16 – Сообщение об успешном удалении болида**

Администратор открывает вкладку «Добавить болид», заполняет все поля и сохраняет модель. Болид появляется в каталоге (Рисунок 5.17).

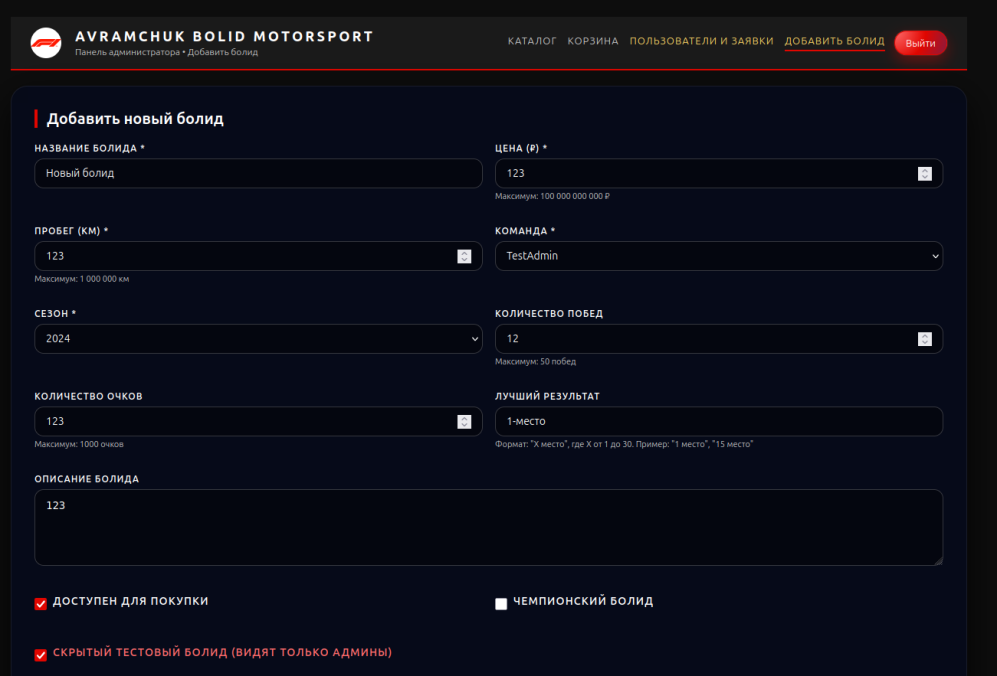


Рисунок 5.17 – Добавление нового болида

**Проверка на странице каталога, что болид успешно добавился (Рисунок 5.18)**

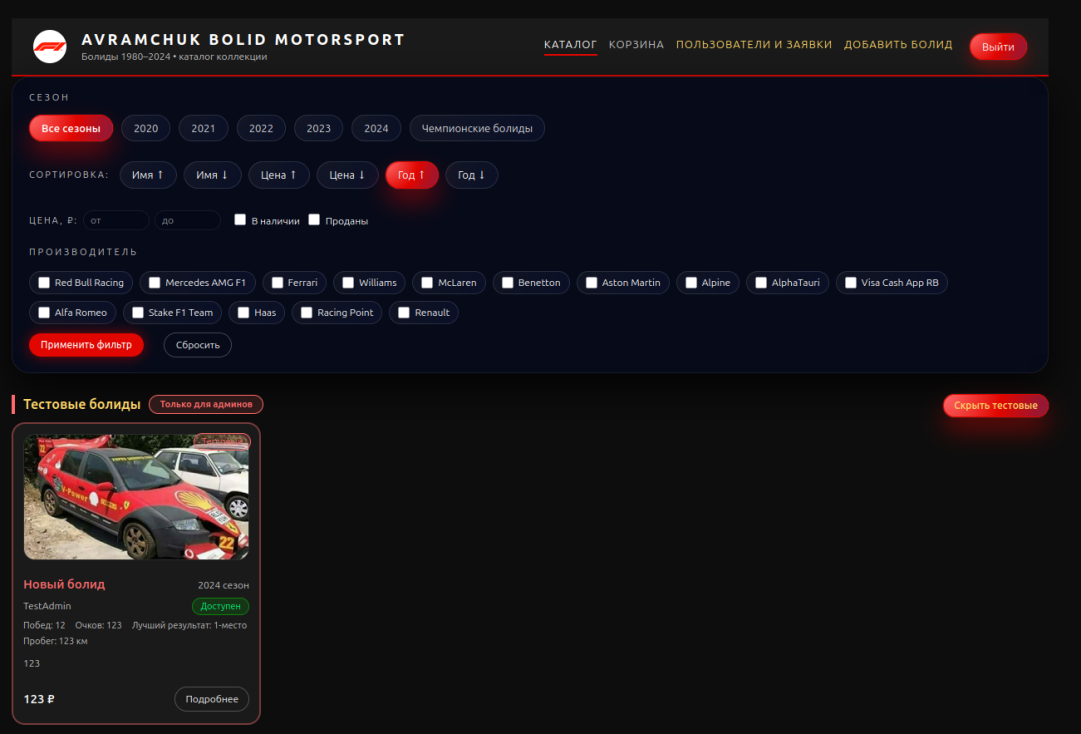
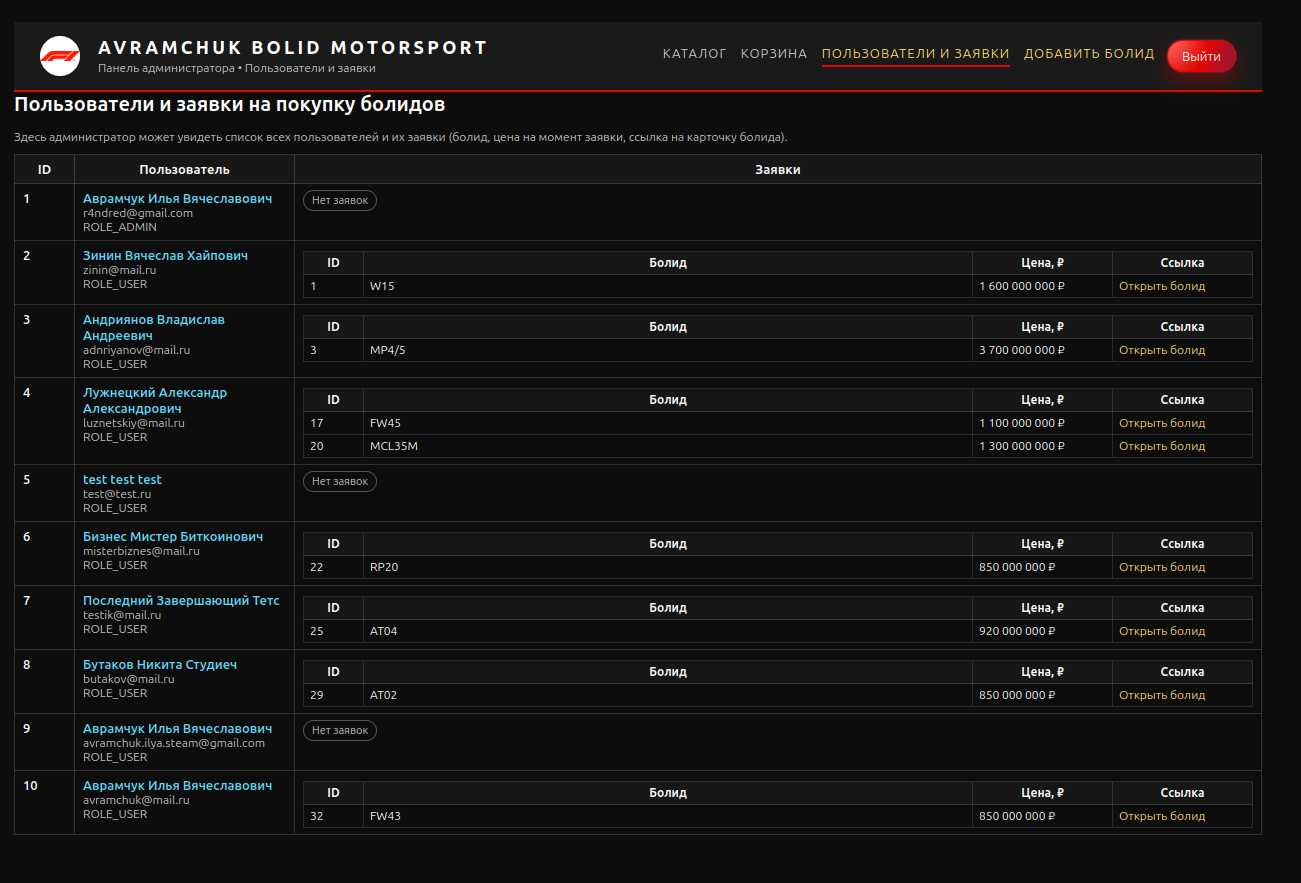


Рисунок 5.18 – Проверка добавления нового болида

Администратор переходит в раздел управления пользователями и заявками. Отображаются все пользователи и их заявки (Рисунок 5.19).



**Рисунок 5.19 – Список пользователей и заявок**

Администратор выбирает пользователя и открывает его карточку. Здесь отображаются личные данные, история заказов и доступные административные действия (Рисунок 5.20).

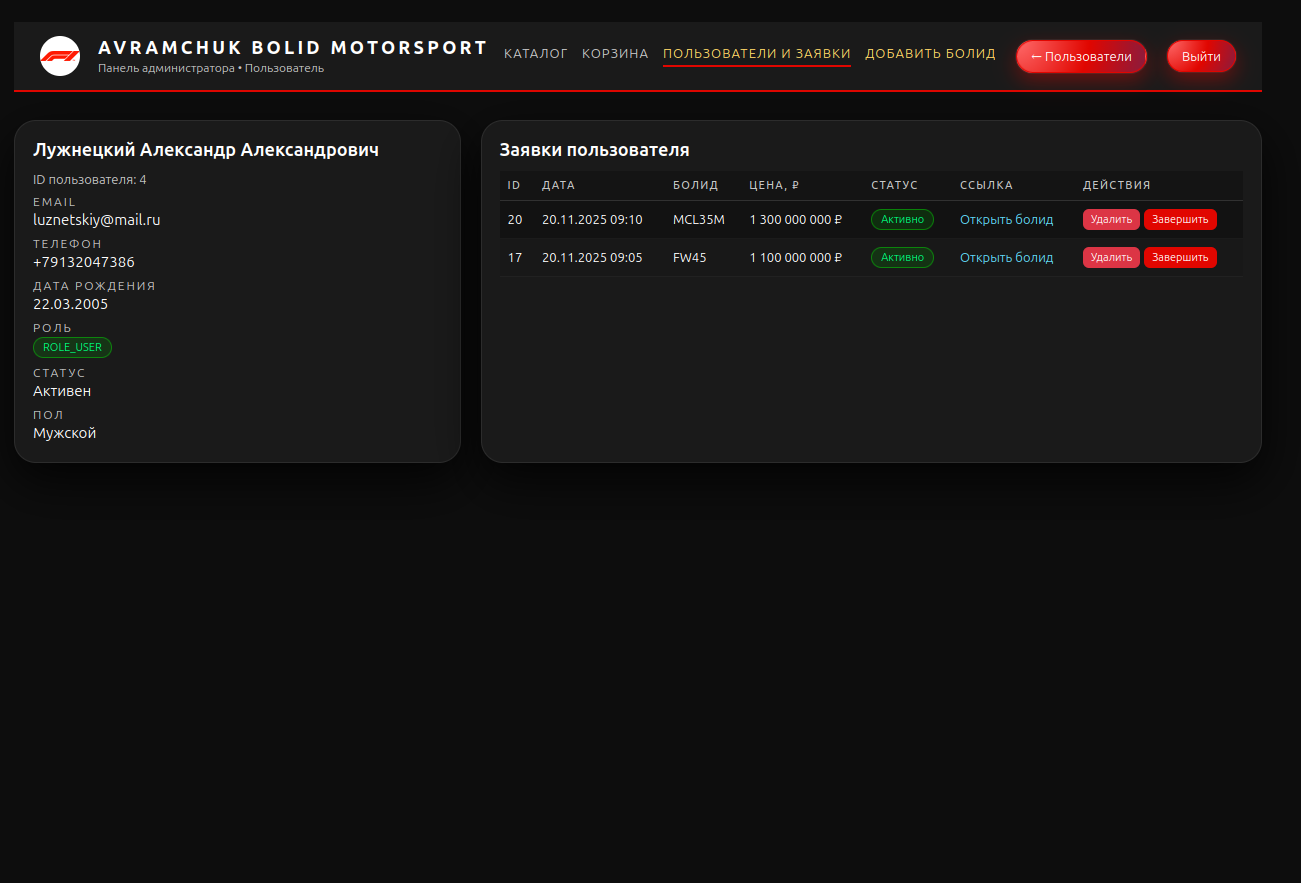


Рисунок 5.20 – Карточка пользователя

Администратор выбирает заявку и удаляет одну из позиций. Если заявка находится в статусе «завершена», удаление недоступно. При успешном удалении отображается уведомление (Рисунок 5.21).

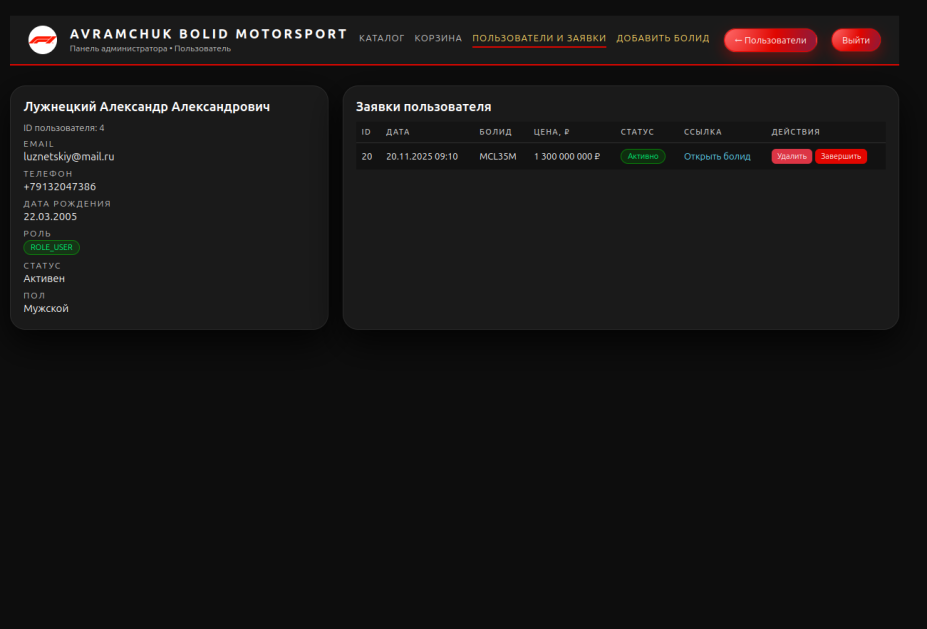
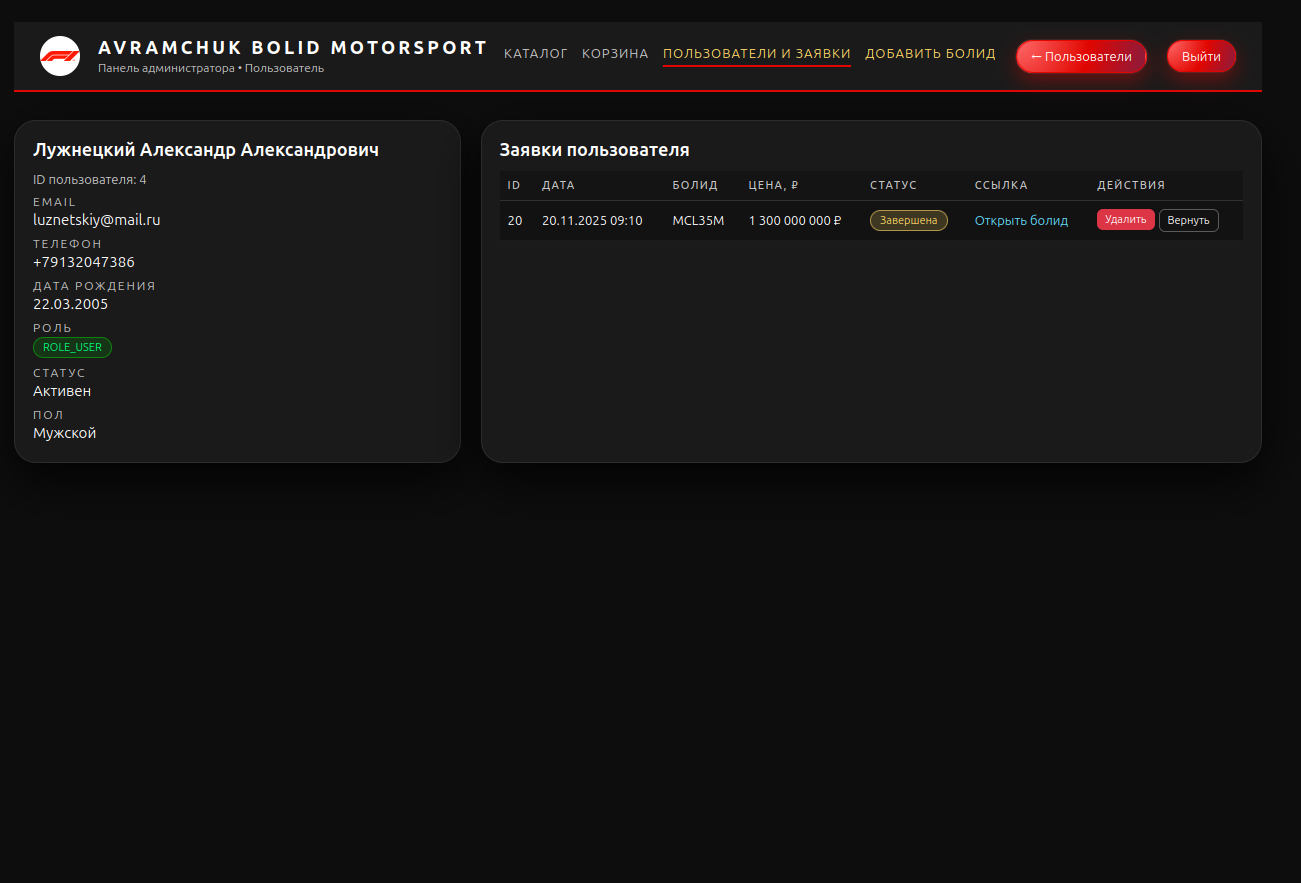


Рисунок 5.21 – удаление болида из заявки

Администратор отмечает заявку как завершённую, после чего болид может быть удалён из системы (Рисунок 5.22).



**Рисунок 5.22 – Завершение заявки**

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана информационная система для управления каталогом болидов Формулы-1, предназначенная для клиентов условного автосалона. Реализовано полноценное веб-приложение, включающее механизмы регистрации и авторизации пользователей, просмотр каталога болидов с фильтрацией, детализированные карточки моделей, функционирующую корзину и систему оформления заявок. Для административного доступа создана отдельная панель управления, обеспечивающая добавление, редактирование и удаление болидов, а также просмотр сведений о пользователях и обработку заявок.

В процессе разработки были изучены и применены современные средства построения серверных приложений на Java с использованием Spring Boot, а также принципов MVC-архитектуры. Взаимодействие с базой данных PostgreSQL реализовано через ORM-фреймворк Hibernate, что позволило обеспечить надёжное хранение данных и строгую структуру сущностей. Были созданы все необходимые связи между таблицами, включая пользователей, болиды, команды, корзину и заявки. Реализованы механизмы валидации данных, проверки доступности моделей, предотвращения удаления болидов, участвующих в активных заявках, а также автоматическая синхронизация корзины с базой данных.

Все цели, поставленные в начале курсовой работы, были достигнуты. Создано удобное и понятное пользовательское интерфейсное решение, позволяющее работать с каталогом болидов в интерактивном режиме. Проведено тестирование интерфейса, логики работы корзины, корректности обработки запросов и работы административной части, результаты подтвердили корректность функционирования системы.

Разработанное веб-приложение обладает потенциалом для дальнейшего развития. К возможным направлениям относятся: расширение каталога характеристик, добавление системы комментариев, внедрение механизма подтверждения оплаты, журналирование действий администраторов, а также доработка системы ролей и разграничения прав доступа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Макконнелл, С. Совершенный код. Мастер класс / пер. с англ. Издательство «Русская редакция», 2010. — 896 с.
2. Коннолли, Т., Бегг, К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание / пер. с англ. — М.: Издательский дом «Вильяме», 2003. — 1440 с.
3. Docker Docs [Электронный ресурс]. — Режим доступа: <https://docs.docker.com/manuals/>, свободный. Загл. с экрана (12.10.2025).
4. PostgreSQL Docs [Электронный ресурс]. — Режим доступа: <https://www.postgresql.org/docs/>, свободный. Загл. с экрана (12.10.2025).
5. Полуэктова, Н. Р. Разработка веб-приложений: учебник для среднего профессионального образования / Н. Р. Полуэктова. — 2-е изд. — Москва: Издательство Юрайт, 2025. — 204 с.
6. f1authentics.com [Электронный ресурс]. — Режим доступа: https://www.f1authentics.com/, свободный. Загл. с экрана (15.10.2025).
7. Spring Framework Documentation [Электронный ресурс]. — Режим доступа: https://spring.io/projects/spring-boot, свободный. Загл. с экрана 13.10.2025).
8. Hibernate ORM User Guide [Электронный ресурс]. — Режим доступа: https://hibernate.org/orm/documentation/, свободный. Загл. с экрана (13.10.2025).
9. Java Persistence API (JPA) Specification [Электронный ресурс]. — Режим доступа: https://jakarta.ee/specifications/persistence/, свободный. Загл. с экрана (13.10.2025).
10. Thymeleaf Documentation — серверные HTML-шаблоны [Электронный ресурс]. — Режим доступа: https://www.thymeleaf.org/, свободный. Загл. с экрана (14.10.2025).

ПРИЛОЖЕНИЕ

Ссылка на GitHub с проектом:

https://github.com/Randred/CourseWork