МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«МИРЭА – Российский технологический университет»**
**РТУ МИРЭА**

**Институт Информационных технологий**
**Кафедра Инструментального и прикладного программного обеспечения**

# КУРСОВАЯ РАБОТА

По дисциплине: Разработка серверных частей интернет-ресурсов
По профилю: Разработка программных продуктов и проектирование информационных систем
Направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Серверная часть веб-сервиса по продаже автомобилей

Студент: Постнов Никита Сергеевич
Группа: ИКБО-24-20
Работа представлена к защите _____ (дата) _____ / _____ /
(подпись и ф.и.о. студента)

Руководитель: старш. преп. Синицын А. В.

Работа допущена к защите _____ (дата) _____ / Синицын А. В. /
(подпись и ф.и.о. руководителя)

Оценка по итогам защиты: (5 – «отл», 4 – «хор», 3 – «удв»)
_____ /
_____ /
(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

**Москва 2022**

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«МИРЭА – Российский технологический университет»**

# РТУ МИРЭА

**Институт информационных технологий (ИТ)**
**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

## ЗАДАНИЕ
## на выполнение курсовой работы

по дисциплине: Разработка серверных частей интернет-ресурсов
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Постнов Никита Сергеевич
Группа: ИКБО-24-20
Срок представления к защите: 05.12.2022
Руководитель: Синицын Анатолий Васильевич, старший преподаватель

**Тема:** Серверная часть веб-сервиса по продаже автомобилей

**Исходные данные:** используемые технологии: Postman, Java, Spring Boot, СУБД PostgreSQL, наличие: межстраничной навигации, внешнего вида страниц, соответствующего современным стандартам веб-разработки, использование паттерна проектирования (MVC). Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18.

**Перечень вопросов, подлежащих разработке, и обязательного графического материала:** 1. Провести анализ предметной области разрабатываемого веб-приложения. 2. Обосновать выбор технологий разработки веб-приложения. 3. Разработать архитектуру веб-приложения на основе выбранного паттерна проектирования. 4. Реализовать слой серверной логики веб-приложения с применением выбранной технологии. 5. Реализовать слой логики базы данных. 6. Создать презентацию по выполненной курсовой работе. 7. Протестировать работу слоя серверной логики с помощью Postman.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО:_____/Р. Г. Болбаков/, «_____»_____2022 г.

Задание на КР выдал:_____/А.В. Синицын/, «_____»_____2022 г.

Задание на КР получил:_____/Н.С. Постнов/, «_____»_____2022 г.

**Содержание**

# ВВЕДЕНИЕ

В современном мире транспорт играет очень важную роль. С его помощью перевозят различные грузы, людей и даже другой транспорт. Для современного обычного человека наиболее привычным транспортом является машина (автомобиль). Машина позволяет быстро перемещаться из одного места в другое, перевозить небольшие грузы и людей. Прогресс сделал современные машины удобными, более экологичными, быстрыми и менее шумными, нежели раньше. Сейчас машина выполняет не только свои прямые функции, но также является предметом гордости, достатка и статуса человека. Данные обстоятельства заставляют множество людей в развитых и развивающихся странах хотеть возыметь свою машину.

Сегодня рынок автомобилей предлагает большое разнообразие брендов и моделей автомобилей. Существуют также и магазины, продающие машины в розницу. Обычно они предлагают каталог с довольно внушительным ассортиментом, выгодные или не очень условия покупки, аренды, тест-драйва, а также услуги по подборке машины.

Для облегчения и оптимизации выбора и заказа машин, магазины создают онлайн сервисы по продаже машин, в основном дублирующие функционал обычных офлайн магазинов. Владельцам же подобных сервисов предоставляется возможность удобного контроля за каталогом и информации о присутствии того или иного автомобиля на складе, его состояния. С их помощью также можно проводить аналитику и управление товаром и персоналом, например менять цены или зарплаты. По сути, для владельцев они предоставляют информационные системы хранения и обработки информации о деятельности предприятия, коим и является магазин по продаже автомобилей.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Предметный анализ

Магазин по продаже автомобилей — это предприятие, занимающееся продажей автомобилей, предоставлением их в аренду или лизинг, предоставлением возможности опробовать автомобиль перед покупкой, так называемых тест драйв. Деятельность осуществляет так называемый диллер, который напрямую у производителя или через посредника закупает готовые автомобили или запчасти к ним.

С целью повышения эффективности осуществления своих функций и обеспечения клиентов более удобным способом взаимодействовать с предприятием удаленно, а также для предоставления управляющим удобного средства мониторинга и администрирования предприятия, его сотрудников и товаров, можно создать веб-сервис, который бы и выполнял вышеописанный функционал.

Данный сервис ориентирован на подбор автомобилей из каталога, с целью ознакомления с их характеристиками, и возможно, для их последующей покупке в филиале розничной сети магазинов, представляемых данным сервисом.

Каждый автомобиль имеет несколько характеристик, среди которых в частности: название бренда, название модели, цвет, возраст, техническое состояние.

Сервис предоставляет доступ к каталогу, в котором находится список имеющихся автомобилей, а также доступ к детальной информации об автомобиле. Предусмотрена возможность заказа автомобиля и отражения сего факта в информационной системе предприятия.

Для мониторинга и управления сервисом, которая предоставляется администраторам или управляющим предприятия в сервисе предусмотрена возможность использования его в многопользовательском режиме,

присутствует административный функционал для управления базой данных магазина.

Взаимодействие с сервисом происходит посредством запросов к серверу, с помощью которых возможно ознакомиться или изменить информацию, хранящуюся в базе данных, доступ к которой, а также бизнес-логику предоставляет и реализует веб-сервер. То есть все то, что реализовывала бы клиентская часть потенциального веб-приложения.

### 1.2 Функциональное назначение

На основе вышеизложенного, можно определить функции (Таблица 1), реализуемые прикладным программным интерфейсом (REST API) сервиса для каждой группы пользователей – обычных пользователей (клиентов) и администраторов.

Хранящаяся информация об объектах, обрабатываемых сервисом должна храниться в электронном хранилище цифровых данных, способного выдерживать высокие нагрузки и множество параллельных соединений.

Таблица 1 – Функции пользователей

| **Пользователь** | **Функция** |
|---|---|
| Клиент | Получить каталог автомобилей |
| | Получить детальную информацию о каждом автомобиле из каталога |
| | Поиск требуемого автомобиля |
| | Получить список автомобилей определенного бренда |
| | Получить изображение автомобиля из каталога |
| | Войти в свой аккаунт |
| | Зарегистрироваться |
| | Выйти из своего аккаунта |
| Администратор | Получить список всех автомобилей |
| | Получить список всех пользователей |
| | Добавить информацию об автомобиле |
| | Добавить информацию о пользователе |
| | Изменить информацию об автомобиле |
| | Изменить информацию о пользователе |
| | Получить информацию о автомобиле |
| | Получить информацию о пользователе |
| | Удалить информацию об автомобиле |
| | Удалить информацию о пользователе |

### 1.3 Диаграмма взаимодействия с веб-приложением

Чтобы лучше понять работу пользователя с приложением составим диаграмму прецедентов (Рисунок 1) для всех групп пользователей.

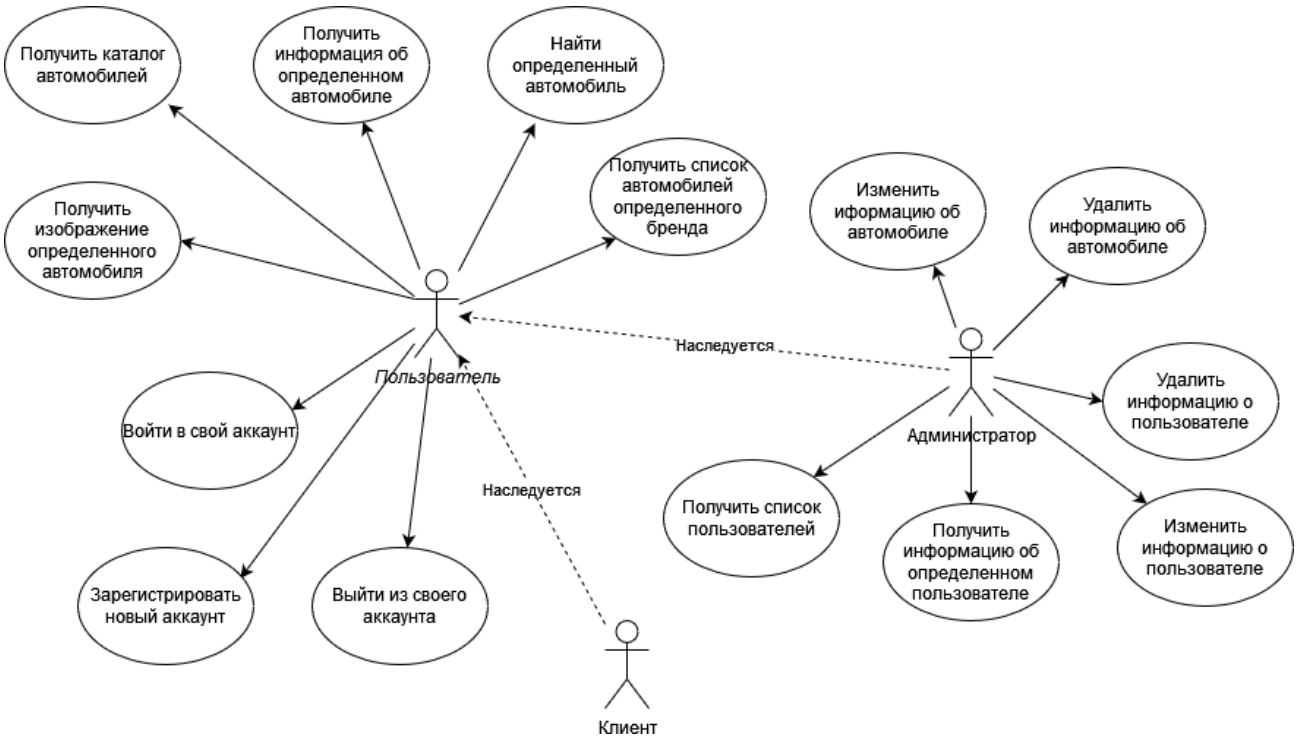


Рисунок 1 – Диаграмма прецедентов

### 1.4 Выбор средств разработки

Разрабатываемый веб-сервис состоит из 2х частей – сервер и база данных.

Серверная часть написана на языке программирования Java [3] программной платформы JVM, так как это один из самых популярных ЯП в принципе и, потому что данный ЯП отлично подходит для создания высоконагруженных профессиональных решений для лидеров в индустрии в сфере веб-технологий. Также используется фреймворк Spring [1], являющийся зарекомендовавшим себя лидером в сфере веб-разработки на платформе JVM.

Также потребуются дополнительные библиотеки для реализации RESP API и архитектуры MVC, библиотеки Hibernate и JPA для объектной реляции данных, библиотека Project Lombok для оптимизации кода. Для сборки проекта используется система сборки Gradle [4] – популярное и мощное решения в данной сфере, доказавшее свои достоинства.

PostgreSQL [2] выбрана в качестве СУБД для базы данных сервиса. Она предоставляет мощь, надежность отказоустойчивость и параллелизм запросов для поддержки и обработки множества одновременных запросов к БД.

Для контейнеризации веб-сервиса используется технология Docker [5] – ведущая технология контейнеризации программного обеспечения.

Разработка будет вестись в интегрированной среде разработки IntelliJ IDEA, являющейся лучшим выбором для разработки под платформу JVM. Для поддержания целостности и удобства разработки используется система контроля версий Git.

Проект расположен в удаленном репозитории на облачной платформе GitHub.com – наиболее популярная подобная платформа.

Тестирование проводится с помощью программы Postman – популярного и мощного инструмента для тестирования REST API серверов.

### 1.5 Техническое задание

Резюмируя, можно выделить следующее:

- Комуницирование с сервером осуществляется с помощью запросов к REST API сервера,
- Взаимодействие пользователей с сервером осуществляется через программу Postman,
- Сервер автоматизирует и оптимизирует работу предприятия и повышает удобство использования услуг предприятия клиентами.

Серверная часть использует:

- язык Java11й версии вместе со своей платформой JVM,
- фреймворк Spring для создания REST-full инфраструктуры сервера, для коммуникации между клиентом и сервером,
- фреймворк Hibernate для создания и управления СУБД и возможности использовать ООП при работе с базой данных,
- PostgreSQL в качестве СУБД для управления базой данных,
- библиотека Project Lombok – для оптимизации разработки,
- система контроля версий Git,
- онлайн платформа для хранения исходных кодов проекта GitHub.com,

- Интегрированная среда разработки IntellyJ IDEA Ultimate для поддержки фреймворков и библиотек, а также для повышения удобства разработки.

# 2. СПЕЦИФИКАЦИЯ ПРИЛОЖЕНИЯ

## 2.1 Описание общей архитектуры приложения

Данное приложение реализовывает архитектуру MVC (Рисунок 2). Данная архитектура представляет из себя 3 слоя.

Первый слой - модель, он, в свою очередь состоит из трех уровней: уровень сервис, который реализует бизнесс-логику всего приложения, уровень репозиторий - предоставляет доступ к данным для уровня сервиса и взаимодействует с СУБД, уровень сущностей - реализует сущности бизнесс-объектов, которые хранятся в БД.

Второй слой - контроллер, не подразделяется на уровни, отвечает за предоставление данных клиентам, взаимодействуя со слоем модели. Контроллер также предоставляет разделение прав пользователей на обычного пользователя и администратора, привелегией которого является управление базой данных. Разделение прав пользователей реализовано через проверку его данных - имени и пароля, которые он указывает при входе, а также его роли, хранящейся в БД и указанной слоем модели при создании нового пользователя, с целью обеспечения безопастности при регистрации нового пользователя, ему всегда присваевается роль обычного пользователя.

Третий слой - представление. В этом приложении слой представления реализуется программой Postman, которая может оправлять и получать запросы к и от сервера.

Само веб-приложение работает внутри отдельного Docker-контейнера для упрощения развертки приложения на целевую систему. Второй контейнер использует база данных и СУБД.

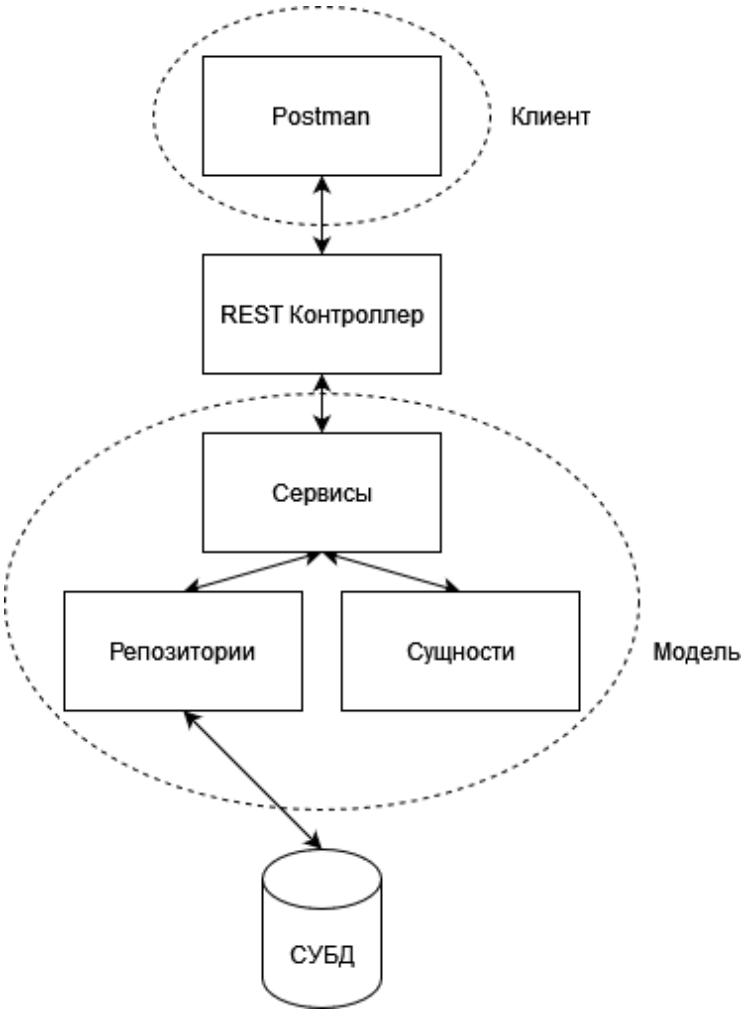


Рисунок 2 – Схема общей архитектуры приложения

### 2.2 Структура данных СУБД

В БД хранятся 5 таблицы (Рисунок 4): машины, пользователи, сессии и 2 таблицы, создаваемые фреймворком. Первые две таблицы соотносятся с сущностями слоя модели.

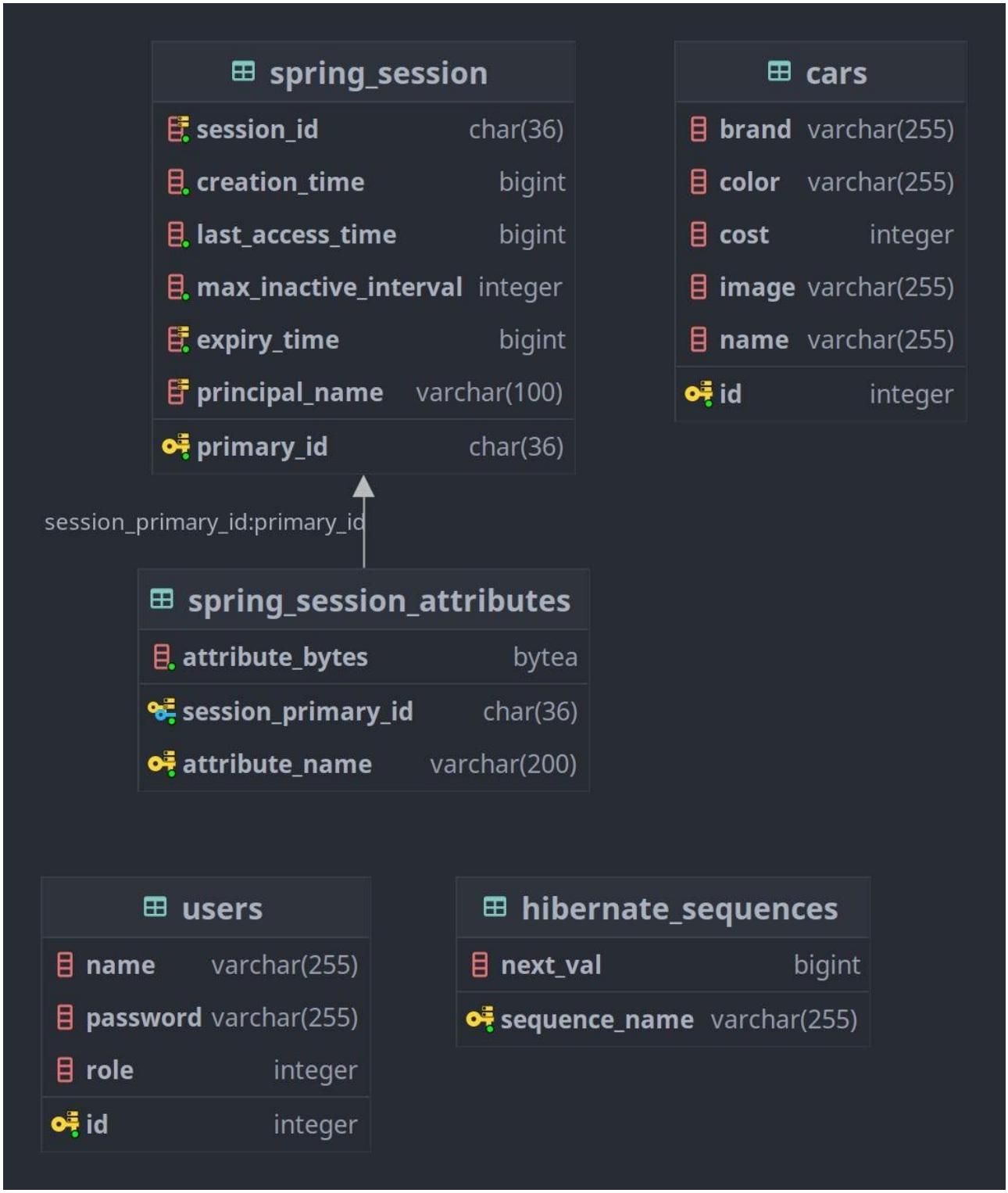


Рисунок 4 - Структура таблиц БД

### 2.3 Протокол взаимодействия с серверной частью

Коммуникация с сервером осуществляется по протоколу HTTP. Он используется при отправке и получении запросов.

Слой контроллера реализует архитектуру REST API, при которой для взаимодействия между клиентом и сервером используются GET, POST, PUT и DELETE запросы (Таблицы 2).

Протокол прикладного уровня HTTP, в свою очередь, использует протокол транспортного уровня TCP и протокол интернет уровня IP.

Таблица 2 – Запрос получения изображения автомобиля

| **Метод** | GET |
|---|---|
| **Адрес** | /static/{imageFileName} |
| **Входные значения** | Название файла изображения автомобиля |
| **Выходные значения** | Статус 200: Файл изображения автомобиля |

Таблица 3 – Запрос входа в аккаунт и создания сессии

| **Метод** | POST |
|---|---|
| **Адрес** | /login |
| **Входные значения** | Form data: name=string&password=string |
| **Выходные значения** | Статус 200: cookie файл, статус 400: ошибка |

Таблица 4 – Запрос выхода из аккаунта и завершения сессии

| **Метод** | POST |
|---|---|
| **Адрес** | /logout |
| **Входные значения** | - |
| **Выходные значения** | Статус 200: cookie файл, статус 400: ошибка |

Таблица 5 – Запрос добавления записи пользователя

| **Метод** | POST |
|---|---|
| **Адрес** | /user |
| **Входные значения** | Json объект сущности пользователя, cooke файл |
| **Выходные значения** | Статус 200: запись добавлена, статус 400: ошибка, статус 403: доступ запрещен |

Таблица 6 – Запрос добавления записи автомобиля

| **Метод** | POST |
|---|---|
| **Адрес** | /car |
| **Входные значения** | Json объект сущности автомобиля, cooke файл |
| **Выходные значения** | Статус 200: запись добавлена, статус 400: ошибка, статус 403: доступ запрещен |

Таблица 7 – Запрос регистрации аккаунта

| **Метод** | POST |
|---|---|
| **Адрес** | /register |
| **Входные значения** | Form data name=string&password=string |
| **Выходные значения** | Статус 200: аккаунт создан, статус 400: ошибка |

Таблица 8 – Запрос получения записи пользователя

| **Метод** | GET |
|---|---|
| **Адрес** | /user |
| **Входные значения** | Query parameter: id=integer, cooke файл |
| **Выходные значения** | Статус 200: Json объект сущности пользователя, статус 400: ошибка, статус 403: доступ запрещен |

Таблица 9 – Запрос получения записи автомобиля

| **Метод** | GET |
|---|---|
| **Адрес** | /car |
| **Входные значения** | Query parameter: id=integer |
| **Выходные значения** | Статус 200: Json объект сущности автомобиля, статус 400: ошибка |

Таблица 10 – Запрос получения всех записей пользователей

| **Метод** | GET |
|---|---|
| **Адрес** | /all/car |
| **Входные значения** | Cooke файл |
| **Выходные значения** | Статус 200: список Json объектов сущности пользователя, статус 400: ошибка, статус 403: доступ запрещен |

Таблица 11 – Запрос получения всех записей автомобилей

| **Метод** | GET |
|---|---|
| **Адрес** | /all/car |
| **Входные значения** | - |
| **Выходные значения** | Статус 200: список Json объектов сущности автомобиля, статус 400: ошибка |

Таблица 12 – Запрос получения записей автомобилей определенного бренда

| **Метод** | GET |
|---|---|
| **Адрес** | /brand |
| **Входные значения** | Query parameter: brand=string |
| **Выходные значения** | Статус 200: список Json объектов сущности автомобиля, статус 400: ошибка |

Таблица 13 – Запрос получения записи автомобиля по его названию

| **Метод** | GET |
|---|---|
| **Адрес** | /searchCar |
| **Входные значения** | Query parameter: name=string |
| **Выходные значения** | Статус 200: json объект сущности автомобиля, статус 400: ошибка |

Таблица 14 – Запрос изменения записи пользователя

| **Метод** | PUT |
|---|---|
| **Адрес** | /user |
| **Входные значения** | Json объект сущности пользователя, cookie файл |
| **Выходные значения** | Статус 200: запись изменена, статус 400: ошибка, статус 403: доступ запрещен |

Таблица 15 – Запрос изменения записи автомобиля

| **Метод** | PUT |
|---|---|
| **Адрес** | /car |
| **Входные значения** | Json объект сущности автомобиля, cookie файл |
| **Выходные значения** | Статус 200: запись изменена, статус 400: ошибка, статус 403: доступ запрещен |

Таблица 16 – Запрос удаления записи пользователя

| **Метод** | DELETE |
|---|---|
| **Адрес** | /user |
| **Входные значения** | Query parameter: id=integer, cookie файл |
| **Выходные значения** | Статус 200: запись удалена, статус 400: ошибка, статус 403: доступ запрещен |

Таблица 17 – Запрос удаления записи автомобиля

| | |
|---|---|
| **Метод** | DELETE |
| **Адрес** | /car |
| **Входные значения** | Query parameter: id=integer, cookie файл |
| **Выходные значения** | Статус 200: запись удалена, статус 400: ошибка, статус 403: доступ запрещен |

# 3. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

## 3.1 Диаграмма классов приложения

Классы приложения, из которых состоят модули и слои архитектуры, взаимодействуют так, как это показано на диаграмме классов (Рисунок 5) и содержат поля и методы, показанные в диаграмме детализации классов (Рисунок 6).

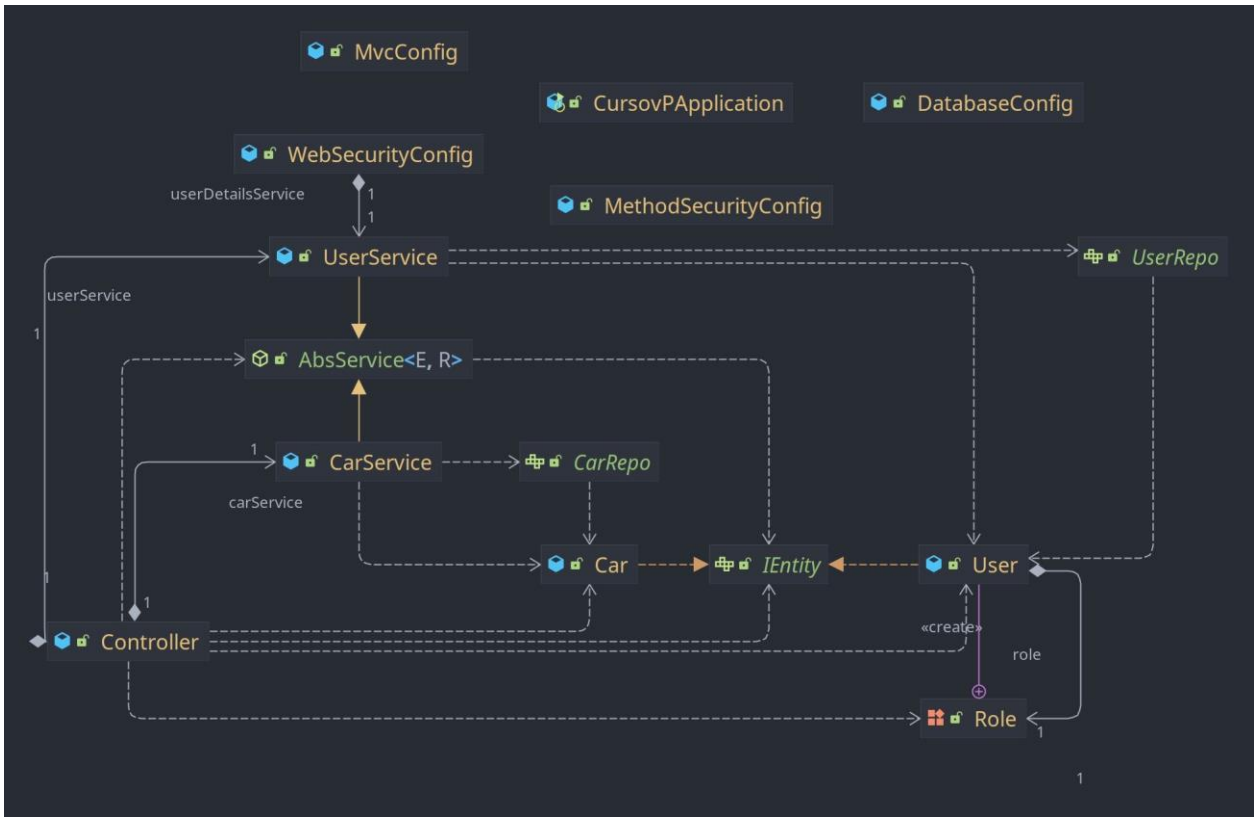


Рисунок 5 – Диаграмма классов


Рисунок 6 – Детализация диаграммы классов

### 3.2 Руководство пользователя

Взаимодействовать с приложением пользователь может с помощью Http запросов. Например, получить список всех автомобилей – каталог, список автомобилей определенного бренда или найти автомобиль по его названию. Пользователь может создать себе аккаунт или войти в него, а после этого,выйти. Если пользователем является администратор он может получить список всех пользователей, получить информацию об определенном пользователе, изменить ее или удалить, также и с автомобилями.

### 3.3 Руководство по сборке и установке

Сперва требуется скачать файлы исходного кода с удаленного репозитория на облачной платформе GitHub.com, установить систему сборки Gradle и систему контейнеризации приложений Docker.

Сборка производится командой gradlew bootJar. Установка производится командой docker-compose up --build.

### 3.4 Методика тестирования

Приложение было полностью протестировано вручную в программе Postman (Рисунки 7-16). Тестировались все запросы, изучалось поведение приложение при выполнении запросов, сравнивалось с ожидаемым и, если оно не удовлетворяло поставленным требованиям, программа дорабатывалась. Так происходило, пока поведение не начинало быть идентичным ожидаемому, затем процесс повторялся для другого запроса.

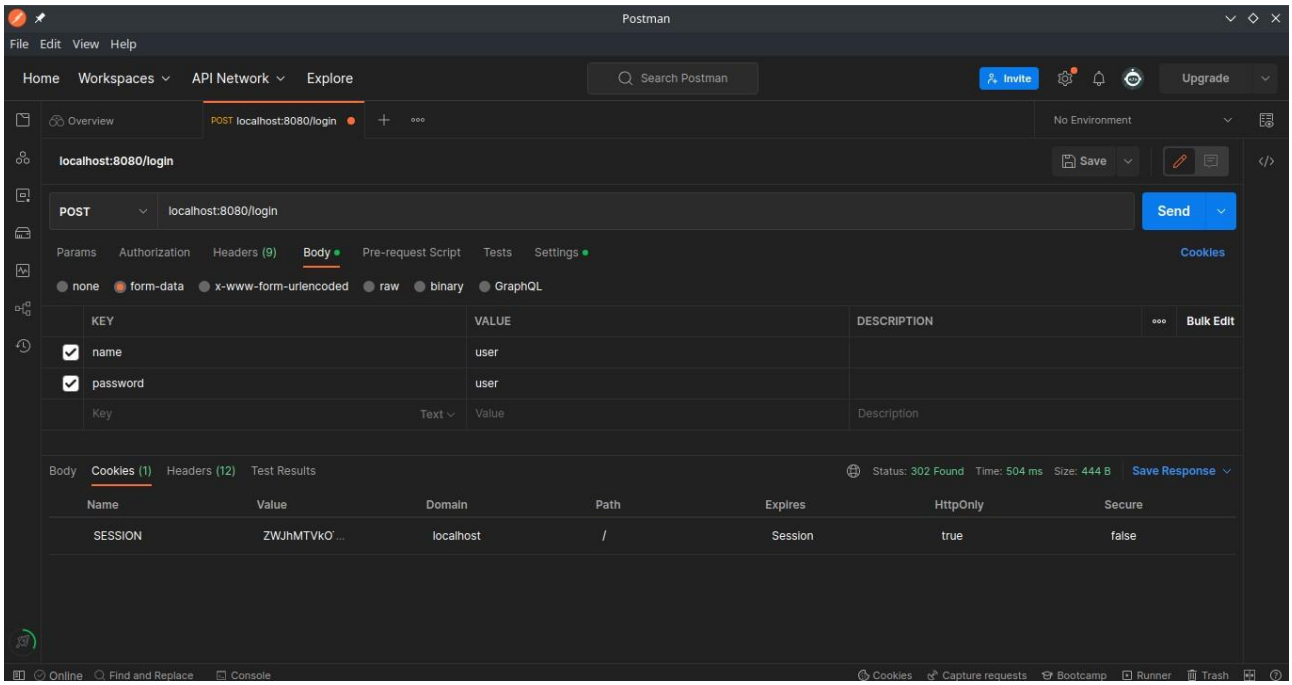

Рисунок 7 – Пример тестирования запроса

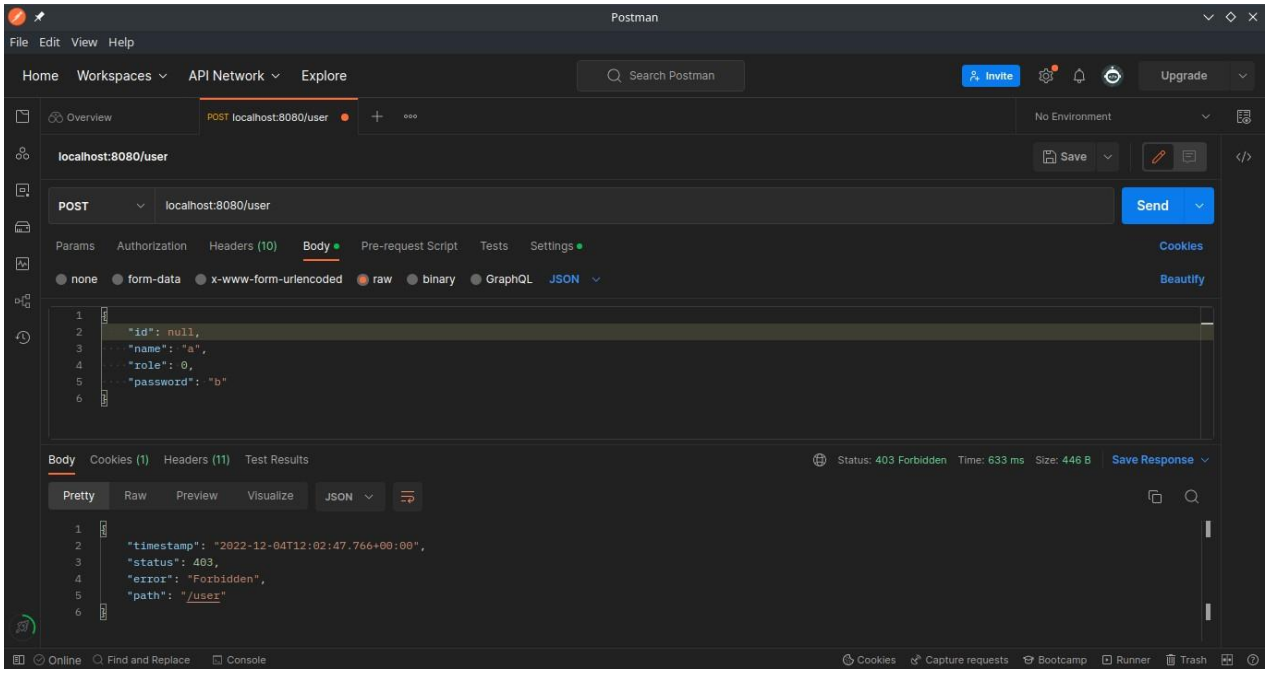


Рисунок 8 – Пример тестирования запроса

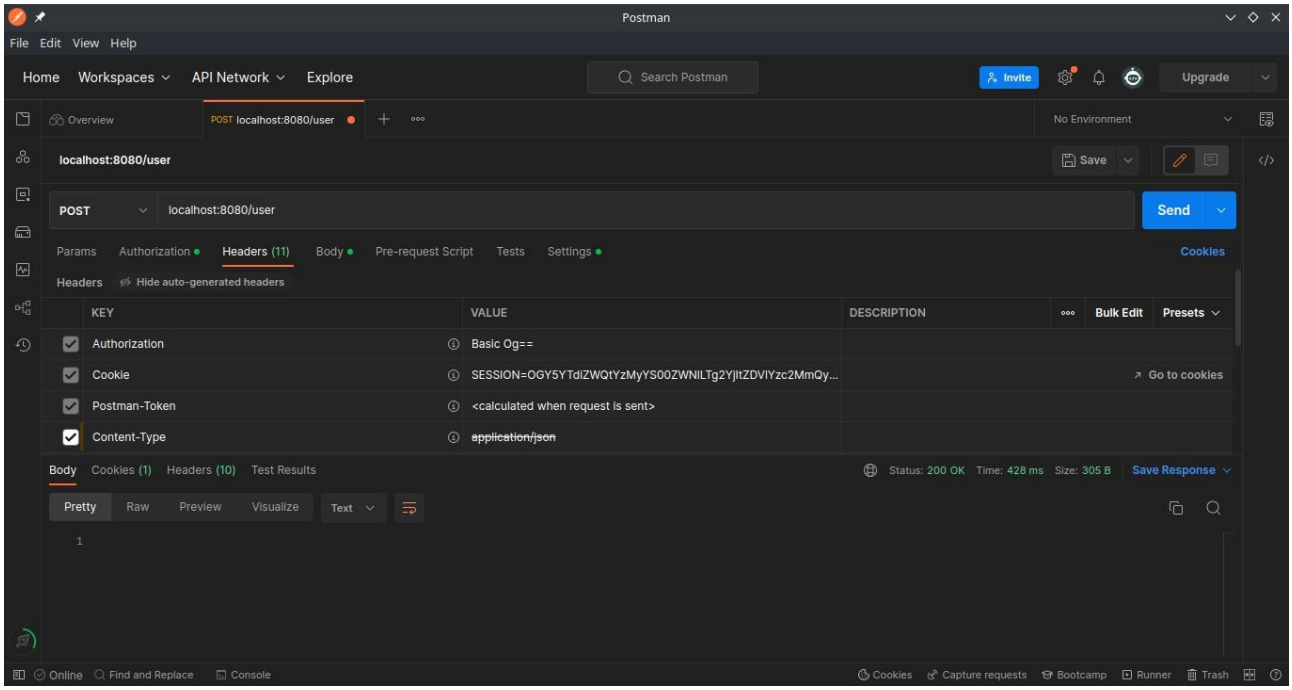


Рисунок 9 – Пример тестирования запроса

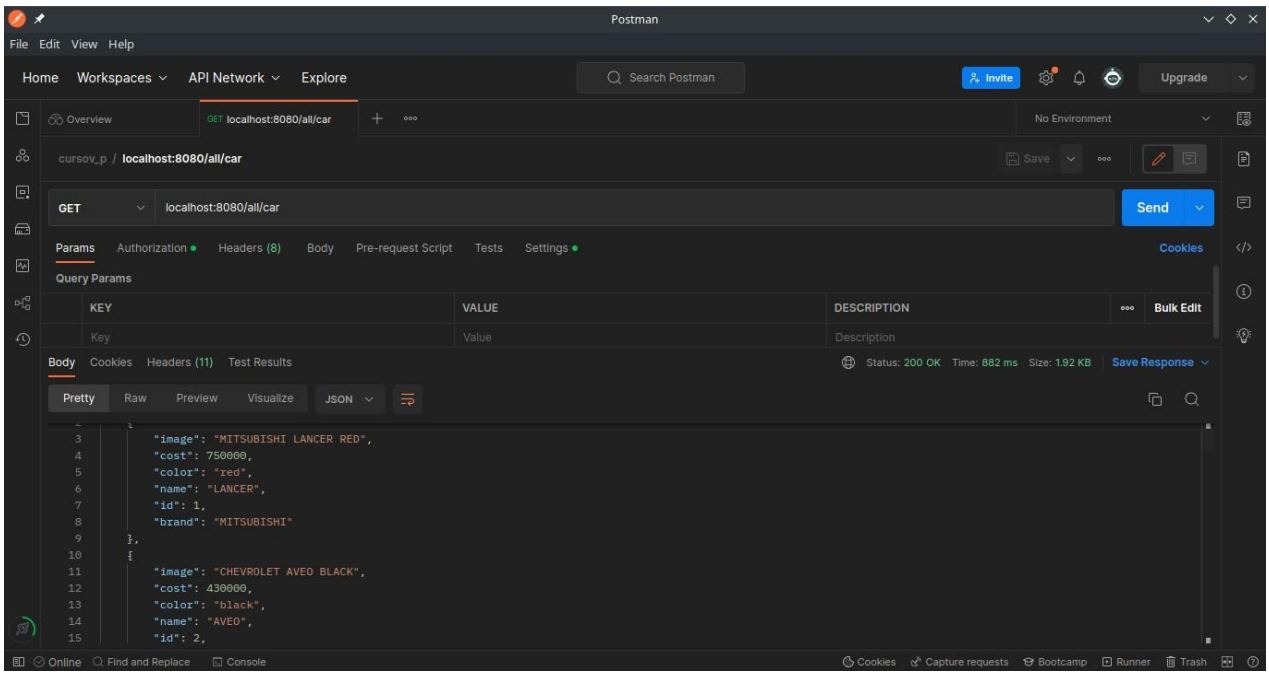


Рисунок 10 – Пример тестирования запроса

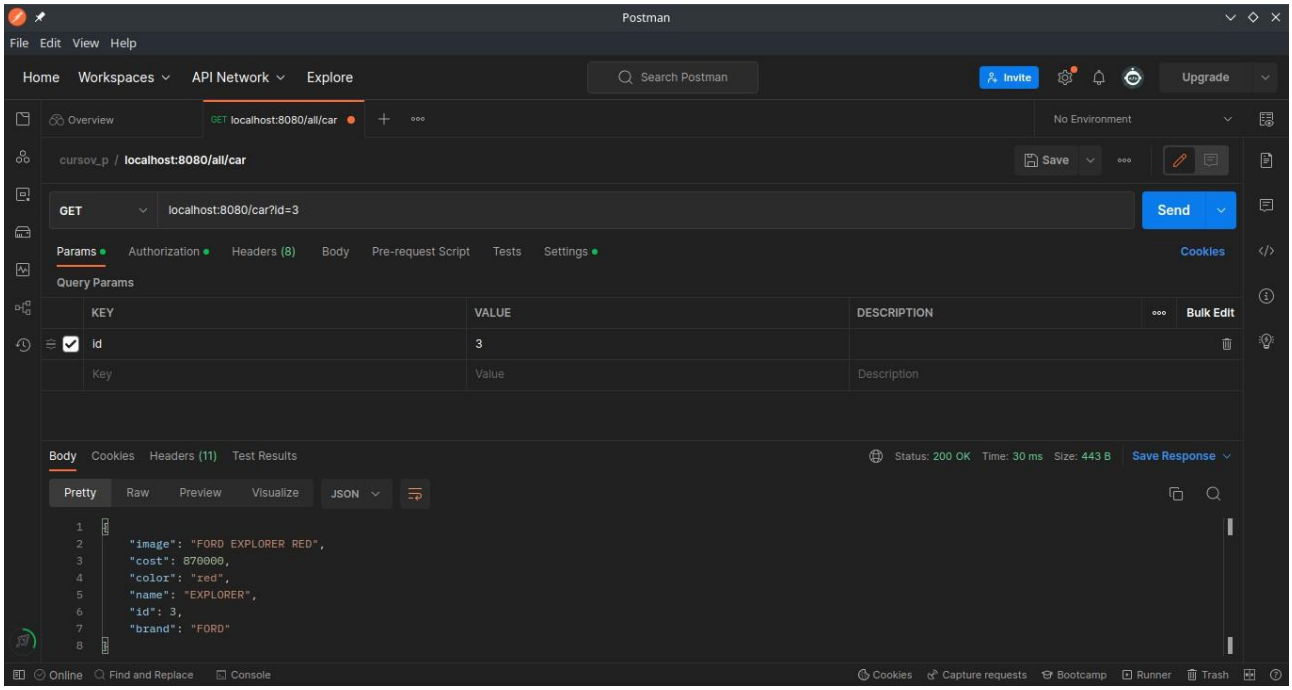


Рисунок 11 – Пример тестирования запроса

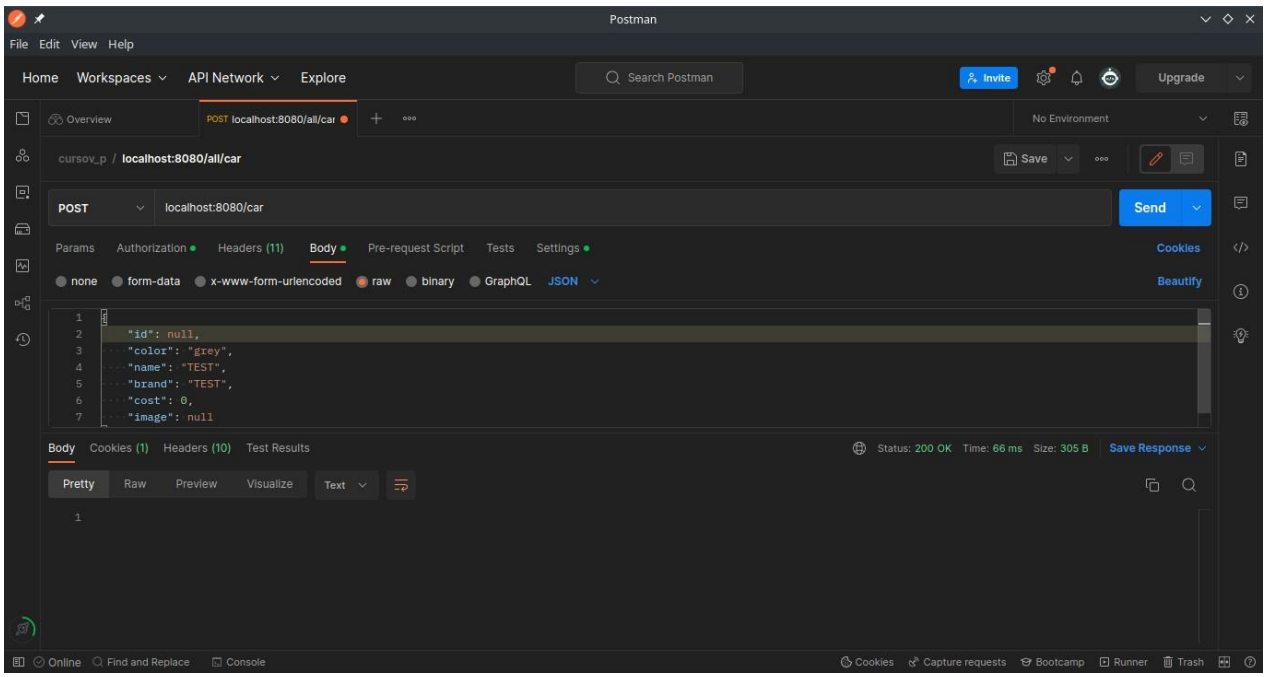


Рисунок 12 – Пример тестирования запроса

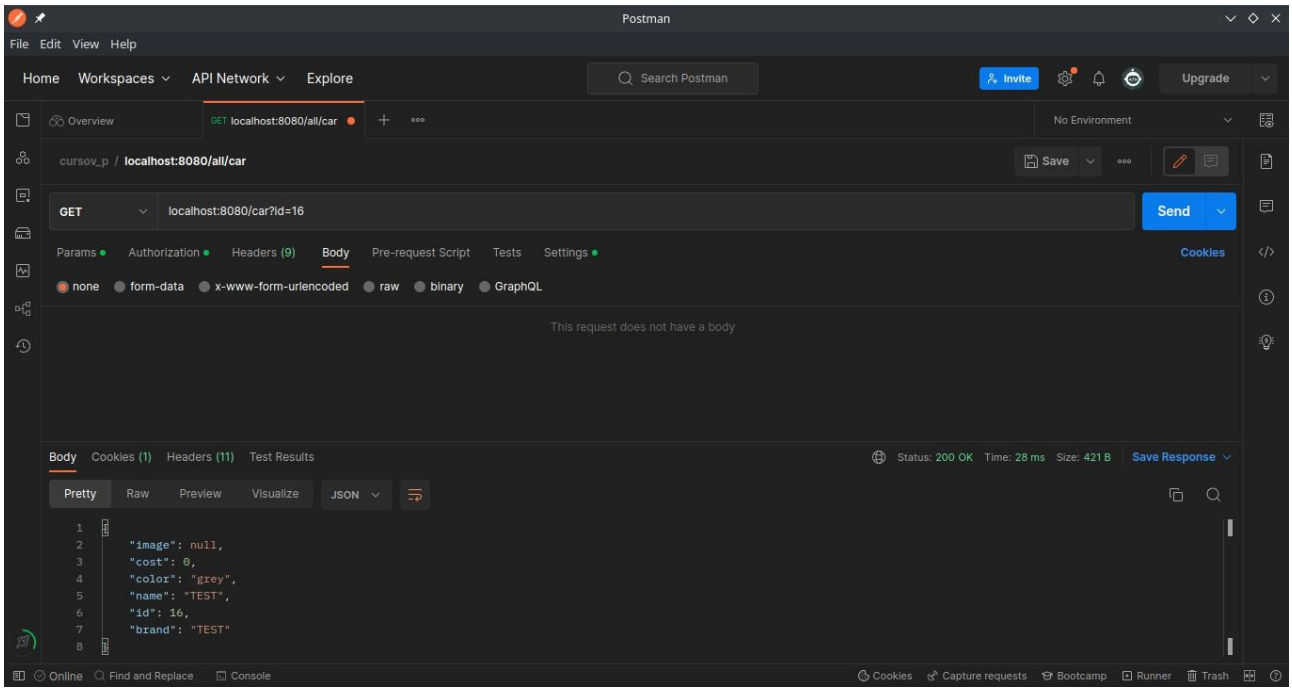


Рисунок 13 – Пример тестирования запроса

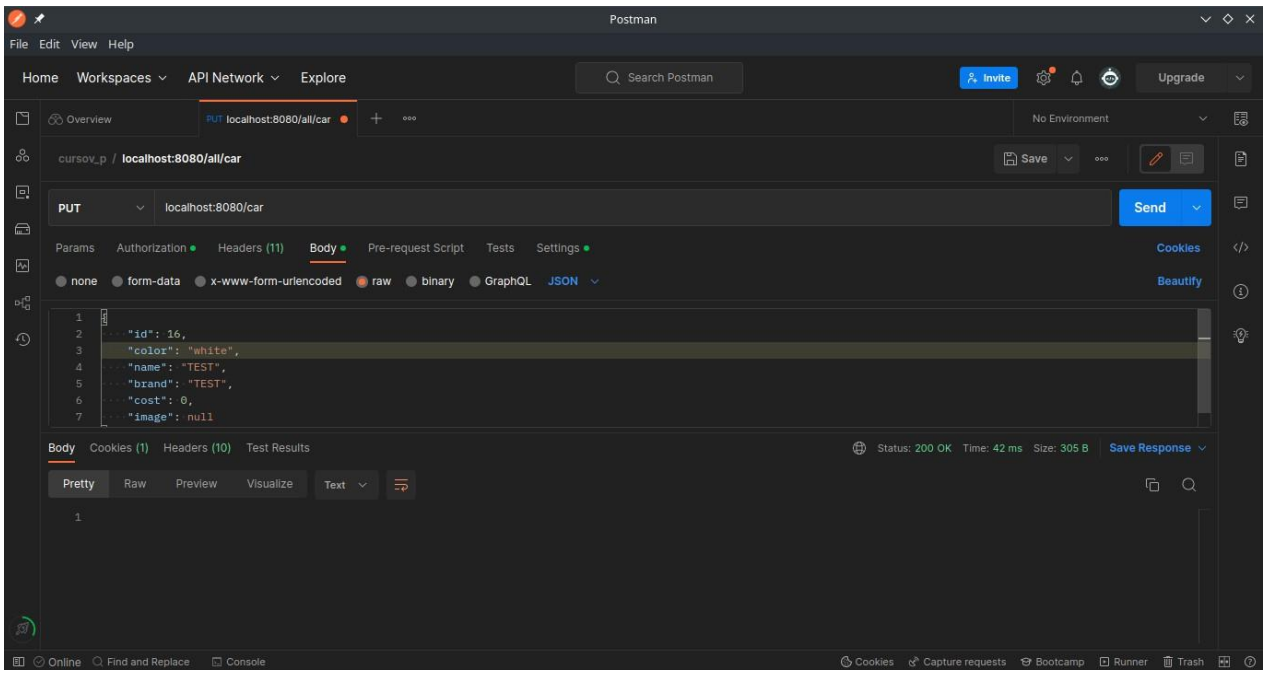


Рисунок 14 – Пример тестирования запроса

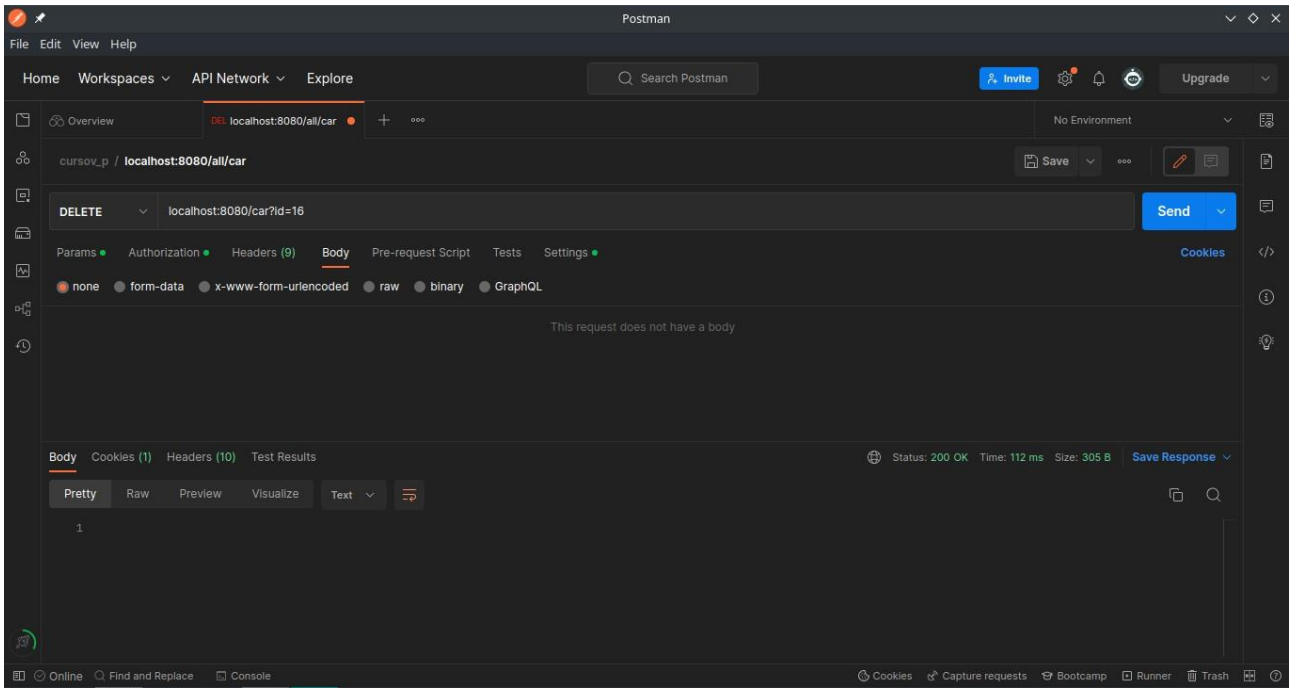


Рисунок 15 – Пример тестирования запроса

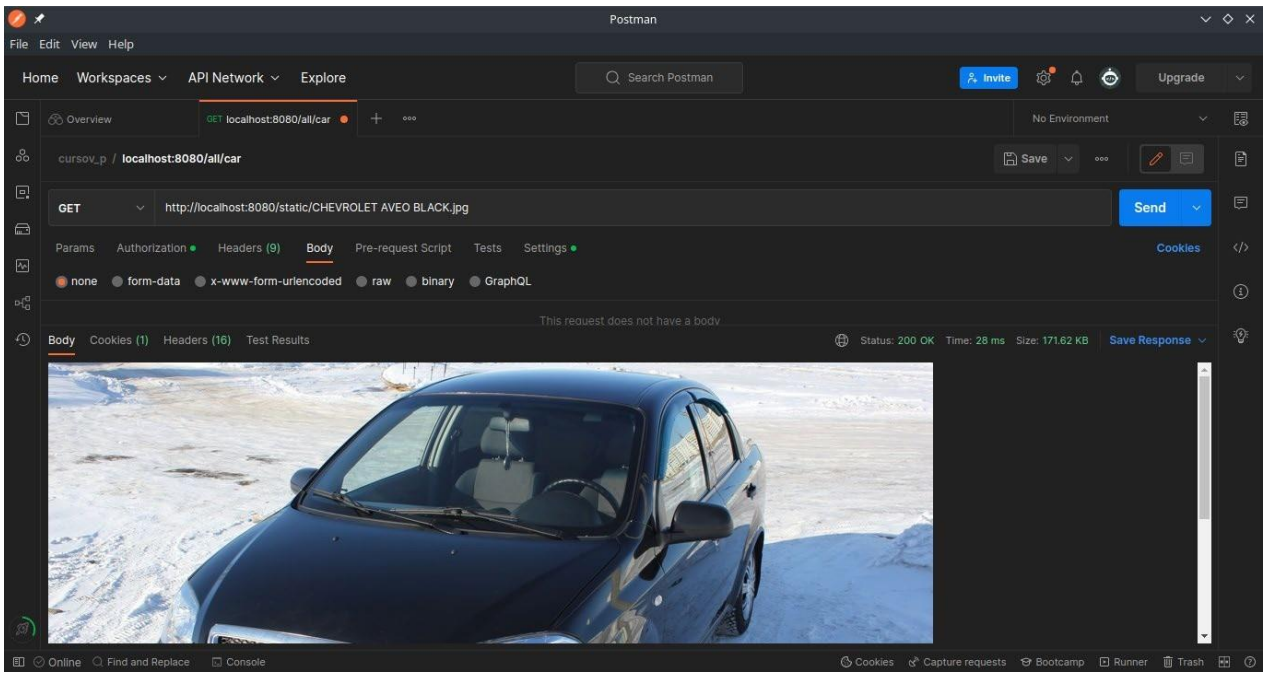


Рисунок 16 – Пример тестирования запроса

# ЗАКЛЮЧЕНИЕ

По итогам выполнения данной работы были получены навыки проектирования и разработки серверных частей интернет-ресурсов и баз данных.

Разработка велась с учетом спроектированной архитектуры на основе паттерна MVC. Использовалась архитектура REST для создания прикладного интерфейса для взаимодействия с сервером.

Созданный программный продукт отвечает всем поставленным техническим заданием целям. Он был полностью протестирован.

Исходные коды данного продукта расположены в удаленном репозитории на облачной платформе GitHub.com и доступны по ссылке https://github.com/NikitaPostnov101/RSCHIR/tree/main/Course%20work.

## СПИСОК ИНФОРМАЦИОНЫХ ИСТОЧНИКОВ


1. Документация по фреймворку Spring [Электронный ресурс] – URL: https://spring.io/ (Дата последнего обращения: 04.12.2022);
2. Документация по СУБД PostgreSQL [Электронный ресурс] – URL: https://www.postgresql.org/ (Дата последнего обращения: 04.12.2022);
3. Документация по библиотекам языка Java [Электронный ресурс] – URL: https://docs.oracle.com/javase/8/docs/api/ (Дата последнего обращения: 04.12.2022);
4. Документация по системе сборке проектов для JVM Gradle [Электронный ресурс] – URL: https://gradle.org/ (Дата последнего обращения: 04.12.2022);
5. Документация по системе контейнеризации приложений Docker [Электронный ресурс] – URL: https://www.docker.com/ (Дата последнего обращения: 04.12.2022);