**Национальный исследовательский ядерный университет «МИФИ»**

|  |  |
| --- | --- |
|  | **Институт интеллектуальных и кибернетических систем**  **Кафедра «Компьютерные системы и технологии»** |

Пояснительная записка

к проекту по курсу «Программирование сетевых приложений»

на тему «Разработка веб-сервиса для поиска

и размещения объявлений о продаже автозапчастей»

Студент гр. Б20-523 Рябец Г.А. /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

Руководитель Овчаренко Е.С. / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

Москва, 2023г.

АННОТАЦИЯ

Пояснительная записка 20 с., 9 рис., 6 ист.

Объектом разработки является веб-сервис для поиска и размещения объявлений о продаже автозапчастей. Целью работы является разработка серверной и клиентской части такого веб-сервиса.

В рамках анализа предметной области были рассмотрены существующие веб-сервисы с аналогичным функционалом.

Было произведено ручное тестирование разработанного приложения.

Серверная часть приложения реализована на языке программирования Python с использованием фреймворка FastAPI. В качестве клиентской части выступает SPA (single page application), написанное на языке программирования JavaScript. В качестве СУБД была выбрана PostgreSQL, для хранения пользовательских сессий используется Redis.

Содержание

[Введение 4](#_Toc154441238)

[1 Теоретическая часть 5](#_Toc154441239)

[1.1 Краткий анализ существующих решений 5](#_Toc154441240)

[1.2 Постановка задачи 5](#_Toc154441241)

[1.3 Архитектура веб-сервиса 7](#_Toc154441242)

[1.4 Сущности базы данных 8](#_Toc154441243)

[2 Практическая часть 9](#_Toc154441244)

[2.1 Используемые технологии 9](#_Toc154441245)

[2.2 Схема базы данных 9](#_Toc154441246)

[2.3 ORM 10](#_Toc154441247)

[2.4 API 11](#_Toc154441248)

[2.5 Аутентификация и авторизация пользователей 13](#_Toc154441249)

[2.6 Браузерный клиент 13](#_Toc154441250)

[3 Тестирование 17](#_Toc154441251)

[Заключение 19](#_Toc154441252)

[Список используемых источников 20](#_Toc154441253)

# **ВВЕДЕНИЕ**

Веб-сервис для поиска и размещения объявлений о продаже автозапчастей разрабатывается с целью обеспечить удобство и эффективность взаимодействия между продавцами и покупателями. Сценарии использования разрабатываемого решения охватывают разнообразные пользовательские группы, включая автовладельцев, автосервисы, магазины запчастей и частные лица. Пользователи должны иметь возможность удобного поиска с использованием категорий и фильтров, просмотра подробного описания объявлений, просмотра фотографий автозапчастей и множество других полезных функций.

В процессе работы по курсу «Программирование сетевых приложений»

была поставлена задача разработать такой проект, а именно реализовать браузерный клиент, сервер приложения и взаимодействие с БД.

# **Теоретическая часть**

* 1. **Краткий анализ существующих решений**

В настоящее время существуют такие сервисы по поиску и продаже автозапчастей, как «Авито», «Exist.ru», «Авто.ru», «Автодок», «Emex» и некоторые другие.

При их анализе были выделены следующие их недостатки:

* необходимость для продавцов покупать тарифы, если нужно продавать детали в достаточно больших объемах
* часто завышенные цены из-за финансовых издержек и наличия посредников между продавцом и покупателем
* неинтуитивный и перегруженный интерфейс, усложняющий пользовательские сценарии
* отсутствие б/у деталей
  1. **Постановка задачи**

Веб-сервис должен предоставлять продавцам и покупателям автозапчастей следующий функционал:

1) Поиск объявлений - возможность искать объявления о продаже автозапчастей с использованием следующих фильтров:

1. фильтр автомобиля (марка, модель, поколение)

2. фильтр детали (категория детали, вид детали)

3. другие фильтры (город, диапазон цены, состояние детали)

Поиск происходит на главной странице, результат поиска – список подходящих объявлений с кратким описанием. Подробное описание каждого объявления можно просмотреть на отдельной странице этого объявления либо во всплывающем окне. По умолчанию объявления сортируются по количеству просмотров, пользователь может отсортировать их так же по цене или дате публикации. Если пользователь считает содержание объявления недопустимым либо столкнулся с мошенничеством, он может пожаловаться на объявление. Зарегистрированный пользователь может сохранить выбранный автомобиль, чтобы в дальнейшем быстро искать детали для него.

2) Размещение, редактирование и удаление объявлений – возможность зарегистрированного пользователя размещать новые объявления, редактировать и удалять свои уже созданные. Каждое объявление включает в себя:

1. информацию об автомобиле (марка, модель, поколение)

2. информацию о детали (категория детали, вид детали)

3. адрес, цену, состояние детали

4. произвольное описание (опционально)

5. фотографию детали (опционально)

6. контактный телефон продавца (из его профиля)

7. дату публикации

3) Регистрация пользователя – возможность пользователя зарегистрироваться для последующего размещения объявлений либо использования гаража. При регистрации указываются:

1. имя

2. город

3. контактный телефон

4. email

5. пароль

4) Аутентификация и авторизация пользователя (далее просто авторизация) – возможность зарегистрированного пользователя войти в свой профиль и получить доступ к созданию, редактированию и удалению своих объявлений, редактированию своего профиля, использованию гаража.

5) Редактирование профиля – возможность зарегистрированного пользователя редактировать информацию в своем профиле.

6) Администрирование – возможность пользователей-администраторов:

1. Создавать профили администраторов

2. Удалять любые профили

3. Удалять любые объявления

4. Редактировать таблицы базы данных (списка возможных автомобилей, деталей и другие)

* 1. **Архитектура веб-сервиса**

Веб-сервис состоит из нескольких компонент: браузерного клиента, сервера приложения, СУБД и key-value хранилища. Была спроектирована архитектура веб-сервиса, представленная на рисунке 1 [1]. Браузерный клиент взаимодействует с сервером по протоколу HTTP. Сервер взаимодействует с СУБД и key-value хранилищем, в котором хранятся пользовательские сессии, используя протоколы баз данных PostgreSQL и Redis.

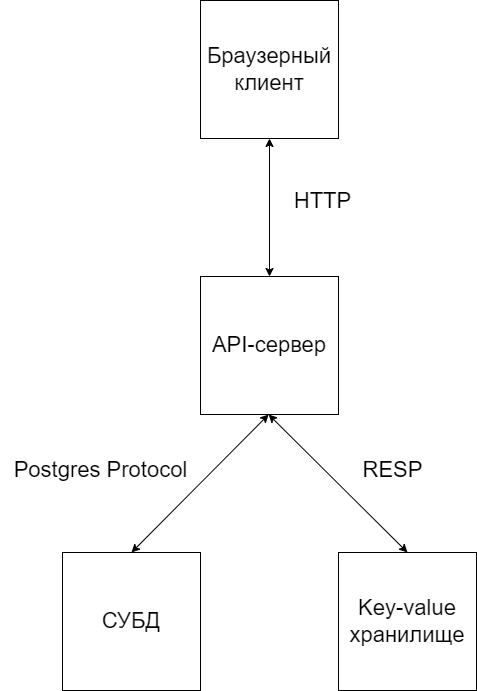


Рисунок 1. Схема взаимодействия между частями приложения

* 1. **Сущности базы данных**

Для хранения в базе данных были выделены сущности, представленные в таблице 1.

Таблица 1. Сущности базы данных

|  |  |
| --- | --- |
| **Сущности** | **Сведения** |
| Профиль | Имя, электронная почта, хеш пароля, телефон, город, флаг, показывающий является ли профиль профилем администратора |
| Марка автомобиля | Название марки, популярность |
| Модель автомобиля | Название модели, популярность, указатель на марку автомобиля |
| Поколение автомобиля | Название поколения, популярность, указатель на модель автомобиля |
| Категория детали | Название категории, описание |
| Тип детали | Название типа, указатель на категорию детали |
| Объявление | Цена, адрес, состояние детали (новая или б/у), описание, время публикации, указатель на профиль создателя, указатель на поколение автомобиля, указатель на тип детали, популярность, количество жалоб |

# **Практическая часть**

**2.1 Используемые технологии**

В качестве языка программирования для разработки серверной части приложения был выбран язык программирования Python, поскольку он позволяет создавать приложения быстро, а также имеет много библиотек, упрощающих разработку. Были использованы следующие библиотеки:

* FastAPI – фреймворк для написания API
* SQLAlchemy – библиотека, предоставляющая ORM
* Pydantic – библиотека для парсинга, валидации и сериализации данных
* Alembic – библиотека для управления миграциями данных

В качестве СУБД был выбран PostgreSQL, так как это свободно распространяемое программное обеспечение, обладающее огромными возможностями [2]. Для хранения пользовательских сессий был выбран Redis, так как он достаточно быстрый и предоставляет весь необходимый в данной задаче функционал.

Для разработки браузерного клиента был выбран язык JavaScript, как практически не имеющий альтернатив, а также библиотека React, как самая популярная библиотека для написания браузерных клиентов [3].

* 1. **Схема базы данных**

Была спроектирована схема базы данных, представленная на рисунке 2. Таблицы car\_brand, car\_model и car\_gen вместе описывают автомобили, причем первичный ключ таблицы car\_gen однозначно задает автомобиль. Таблицы detail\_category и detail\_type – описывают детали, причем первичный ключ таблицы detail\_type однозначно задает деталь. Кроме того, есть таблица car\_in\_garage типа «многие ко многим», в которой хранится информация о всех автомобилях, сохраненных пользователями в свой профиль.



Рисунок 2. Схема базы данных

* 1. **ORM**

Для того чтобы отобразить объекты базы данных в объекты языка программирования Python, была использована ORM, а именно SQLAlchemy. Были описаны классы, дублирующие схему базы данных. Такой подход позволяет не только оперировать объектами базы данных в виде объектов языка Python, но и автоматически генерировать и применять SQL скрипты при миграции схемы базы данных, для чего используется библиотека alembic.

* 1. **API**

Был спроектирован и реализован API, позволяющий клиентскому приложению добавлять, изменять, получать и удалять данные из базы данных, используя заданную бизнес-логику [4]. В таблицах 2, 3, 4, 5, 6 приведены реализованные эндпоинты и их назначение.

Таблица 2. Эндпоинты API для работы с авторизацией

|  |  |  |
| --- | --- | --- |
| Эндпоинт | HTTP-метод | Назначение |
| /auth/register | POST | Регистрация |
| /auth/login | POST | Вход в аккаунт |
| /auth/logout | POST | Выход из аккаунта |

Таблица 3. Эндпоинты API для работы с профилем

|  |  |  |
| --- | --- | --- |
| Эндпоинт | HTTP-метод | Назначение |
| /profile | GET | Получение информации о профиле |
| /profile | PATCH | Редактирование профиля |
| /profile/garage | GET | Получение списка сохраненных автомобилей |
| /profile/garage/cars | POST | Сохранение автомобиля |
| /profile/garage/cars/{car\_gen\_id} | DELETE | Удаление сохраненного автомобиля |
| /profile/products | GET | Получение своих объявлений |
| /profile/products | POST | Создание объявления |
| /profile/products/{product\_id} | DELETE | Удаление объявления |
| /profile/products/{product\_id} | PATCH | Редактирование объявления |

Таблица 4. Эндпоинты API для работы с автомобилями

|  |  |  |
| --- | --- | --- |
| Эндпоинт | HTTP-метод | Назначение |
| /cars/brands | GET | Получение списка марок автомобилей |
| /cars/brands | POST | Создание марки автомобиля |
| /cars/brands/{brand\_id}/models | GET | Получение списка моделей автомобилей |
| /cars/brands/{brand\_id}/models | POST | Создание модели автомобиля |
| /cars/brands/{brand\_id}/models/{model\_id}/gens | GET | Получение списка поколений автомобилей |
| /cars/brands/{brand\_id}/models/{model\_id}/gens | POST | Создание поколения автомобиля |

Таблица 5. Эндпоинты API для работы с деталями

|  |  |  |
| --- | --- | --- |
| Эндпоинт | HTTP-метод | Назначение |
| /details/categories | GET | Получение списка категорий деталей |
| /details/categories | POST | Создание категории детали |
| /details/categories/{category\_id}/types | GET | Получение списка типов деталей |
| /details/categories/{category\_id}/types | POST | Создание типа детали |

Таблица 6. Эндпоинты API для работы c объявлениями

|  |  |  |
| --- | --- | --- |
| Эндпоинт | HTTP-метод | Назначение |
| /products | GET | Поиск объявлений |
| /products/{product\_id}/increase\_score | POST | Увеличение счетчика просмотров объявления |
| /products/{product\_id}/add\_complaint | POST | Увеличение счетчика жалоб на объявление |

* 1. **Аутентификация и авторизация пользователей**

Весь функционал приложения, кроме поиска объявлений, доступен только авторизованным пользователям. Аутентификация реализована с помощью механизма сессий. При вводе почты и пароля, если данные верны, в ответе сервера содержится cookie с сгенерированным идентификатором сессии. Этот идентификатор хранится в key-value хранилище. Аутентификация реализована в слое бизнес-логики: при запросе клиента на эндпоинт, защищенный авторизацией, сервер проверяет, имеет ли данный пользователь право на выполнение данного действия. Пользователь так же может выйти из своего профиля, при этом идентификатор сессии удаляется из key-value хранилища. Если этого не происходит, то он удаляется через заданный промежуток времени.

* 1. **Браузерный клиент**

Для взаимодействия с сервером React-приложение использует библиотеку axios, делая асинхронные запросы к API. На основе предполагаемых пользовательских сценариев был спроектирован и реализован пользовательский интерфейс.

На рисунках 3, 4, 5 и 6 представлены макеты пользовательского интерфейса, а именно окна авторизации, страницы поиска объявлений, окна просмотра детальной информации об объявлении и окна создания объявления.



Рисунок 3 – Окно авторизации

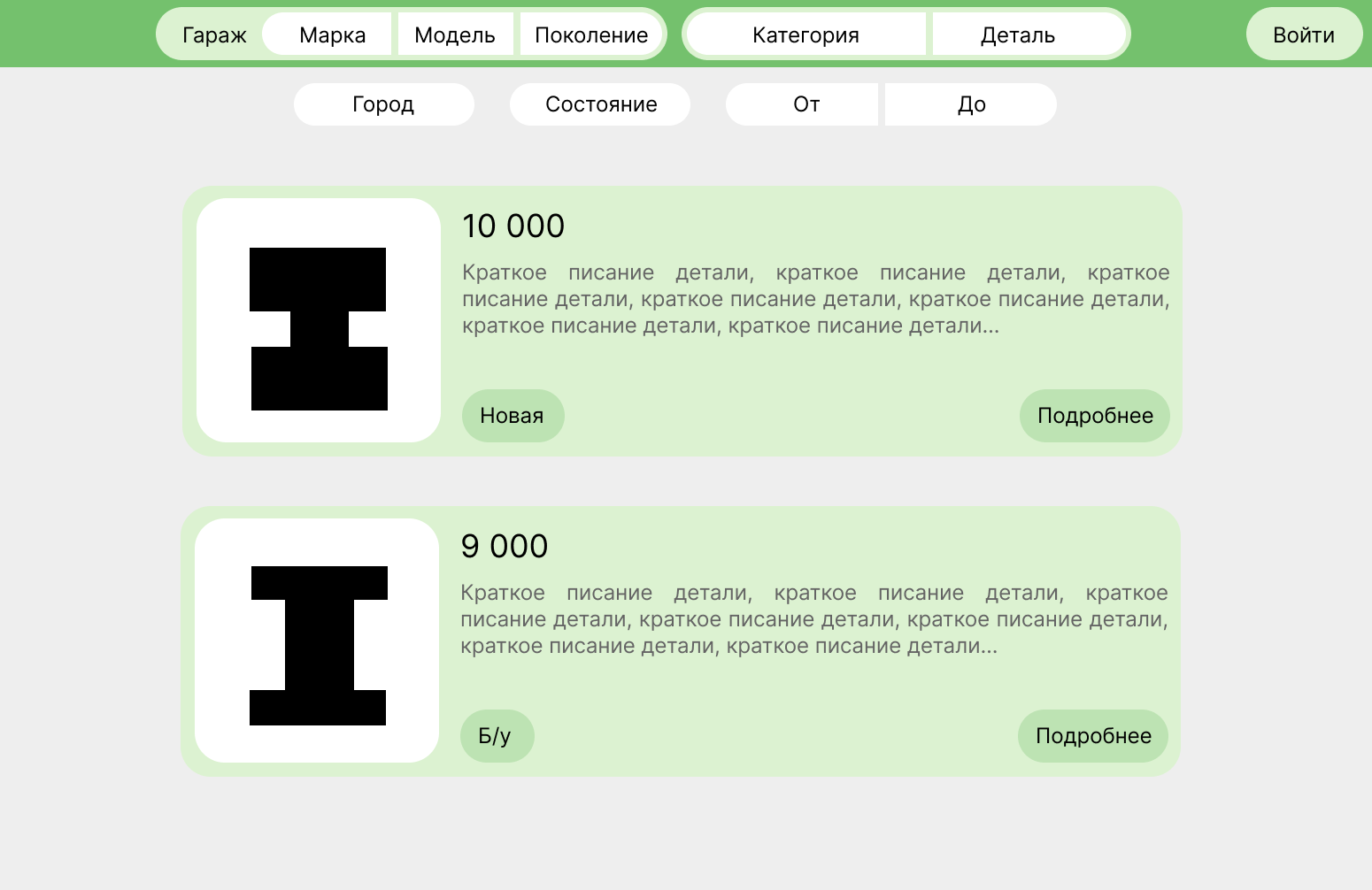


Рисунок 4 – Окно поиска объявлений

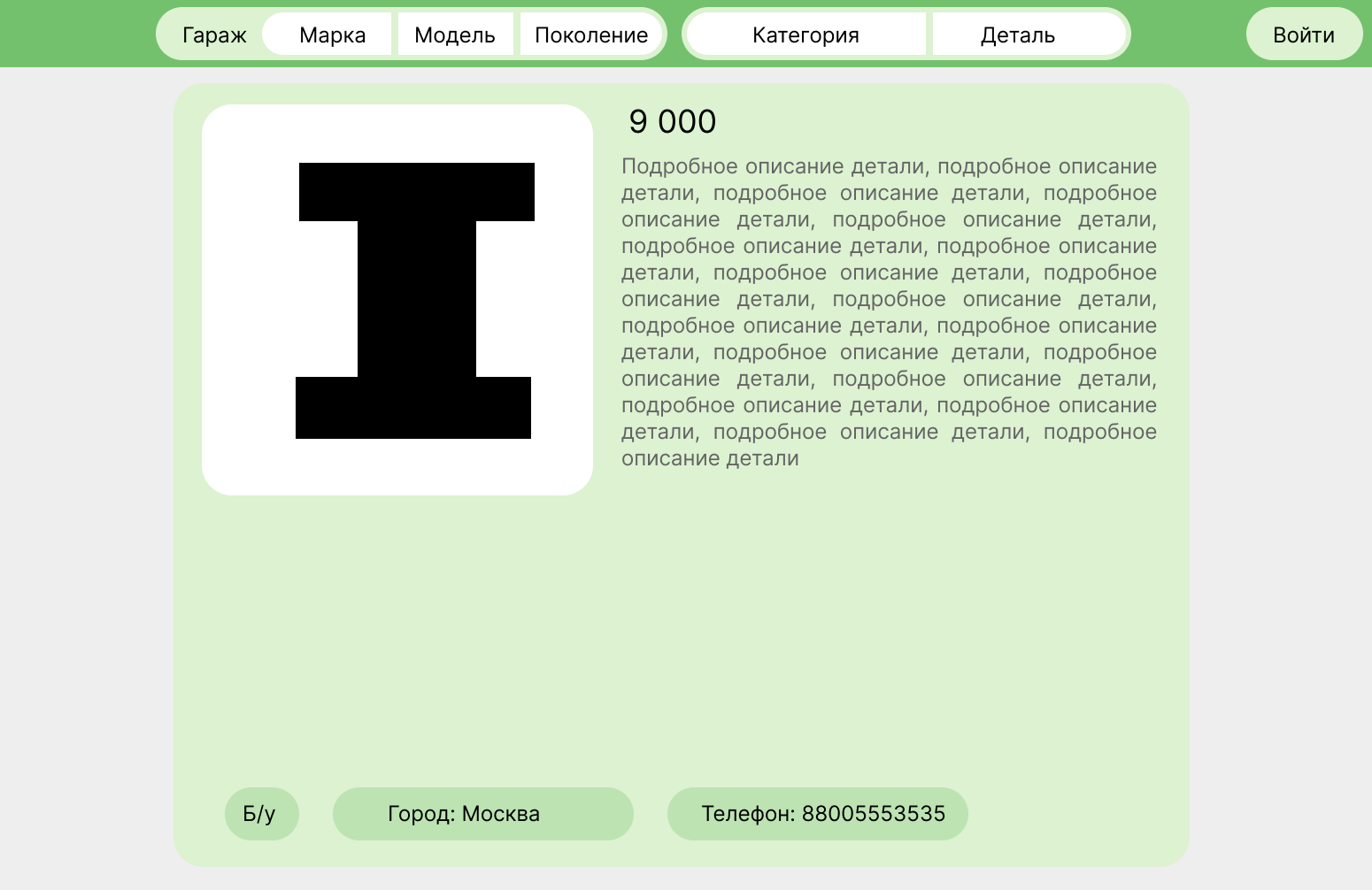


Рисунок 5 – Окно просмотра детальной информации об объявлении

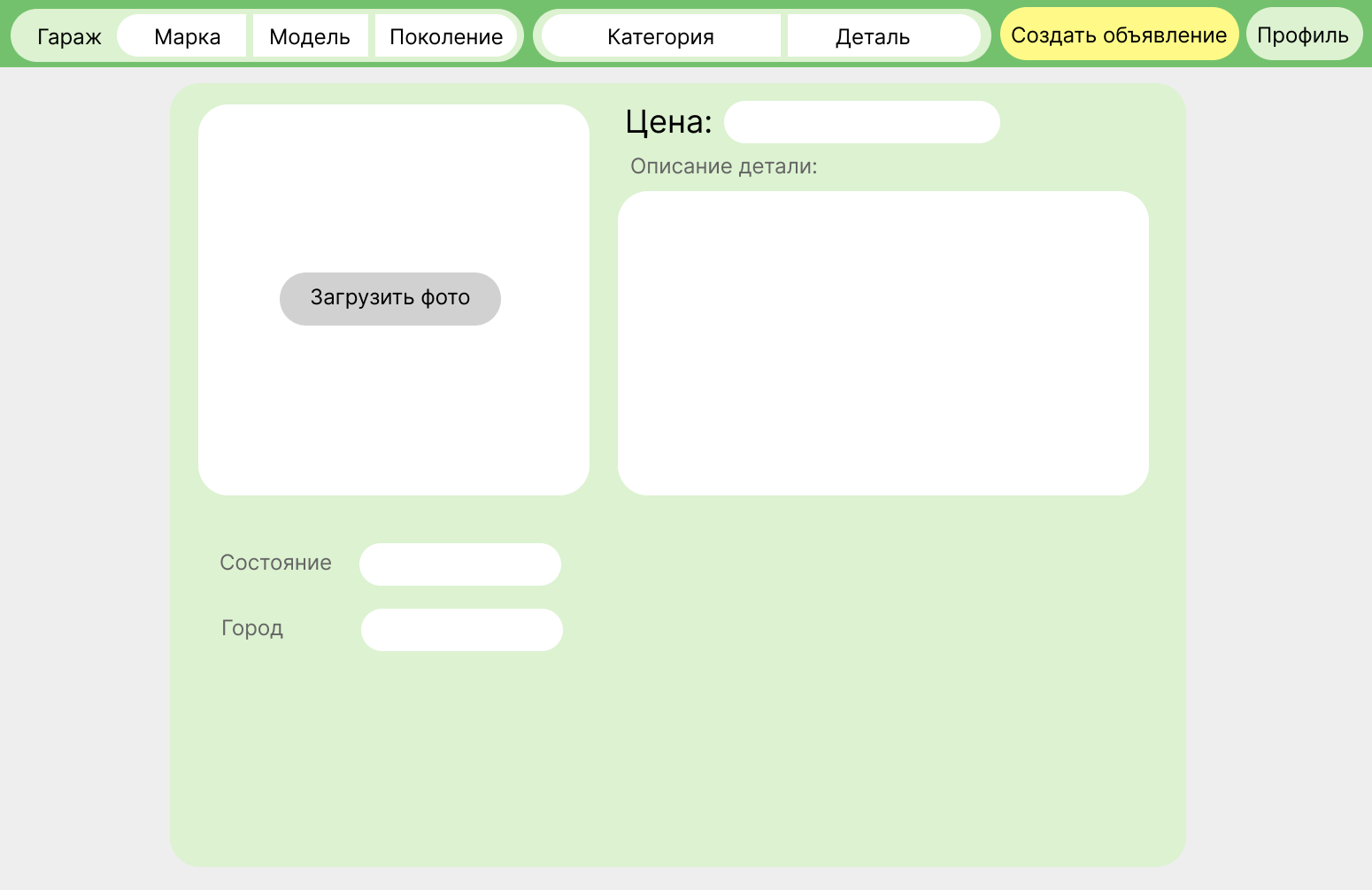


Рисунок 6 – Окно создания нового объявления

Первый основной сценарий использования – поиск объявлений. Он осуществляется посредством последовательного задания автомобиля, детали и фильтров. Автомобиль задается своей маркой, моделью и поколением. Деталь задается категорией и типом. Эти 5 полей поддерживают поиск по первым символам, а также выпадающие списки, динамически подстраивающиеся под уже выбранные значения. Фильтры являются опциональными и включают в себя выбор города, состояния детали и диапазона цены.

После получения списка объявлений можно просмотреть подробную информацию о нужном объявлении, нажав кнопку «подробнее».

Второй важный сценарий использования – это создание объявление. Все созданные объявления можно просмотреть в своем профиле.

Кроме интерфейса пользователя, с помощью библиотеки sqladmin была реализована административная панель. Используя эту панель, администраторы могут редактировать содержимое базы данных, в том числе добавлять новые виды автомобилей и деталей, удалять и редактировать объявления, удалять профили. На рисунке 9 представлен внешний вид административной панели на странице со списком объявлений.

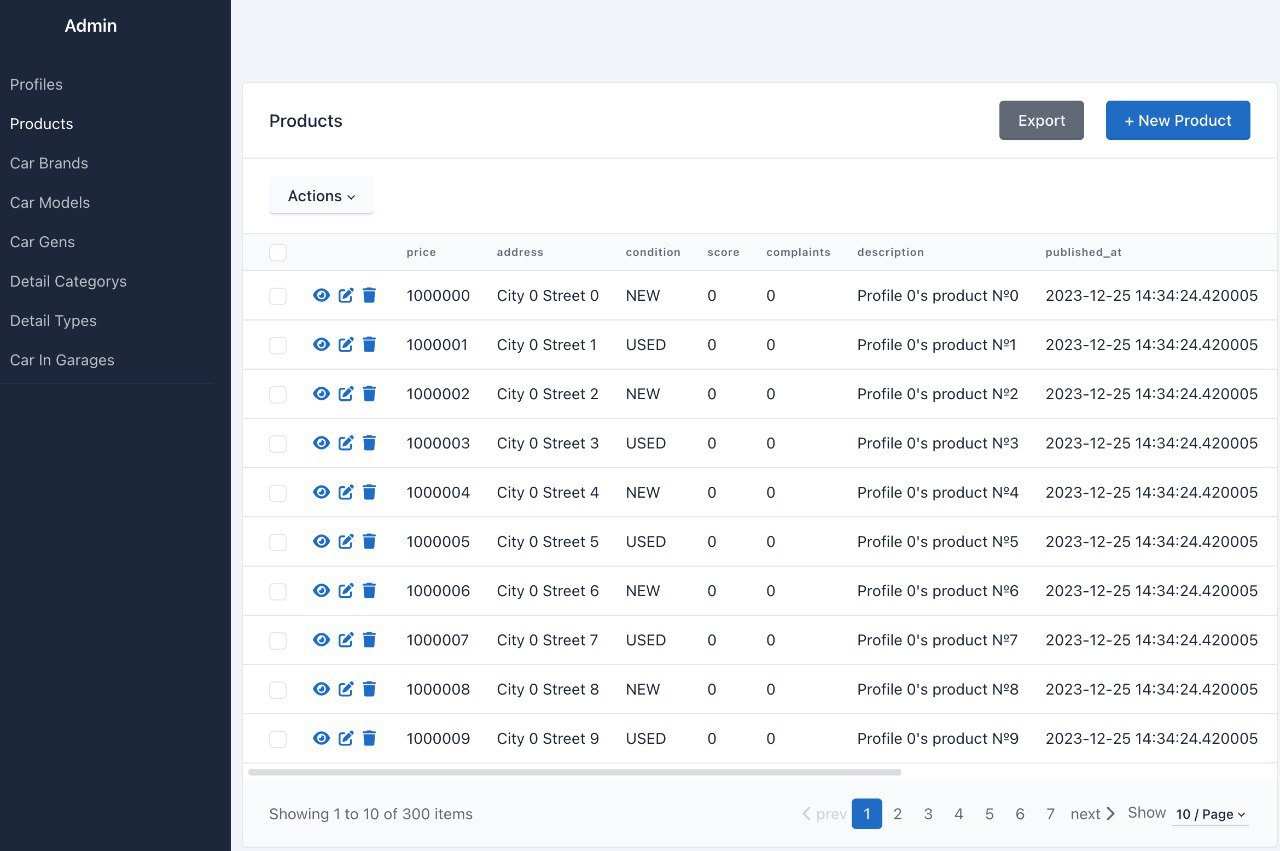


Рисунок 9 – Внешний вид административной панели на странице со списком объявлений

1. **Тестирование**

При разработке серверной части приложения было проведено ручное тестирование функционала с помощью инструментов Postman и Swagger.

Postman это программа для отправки запросов некоторому API и получения ответов [5]. С помощью неё удобно проводить тестировать эндпоинты, изменяя параметры запросов. Postman так же поддерживает авторизацию. На рисунке 7 представлен пример ручного тестирования эндпоинта при помощи Postman.

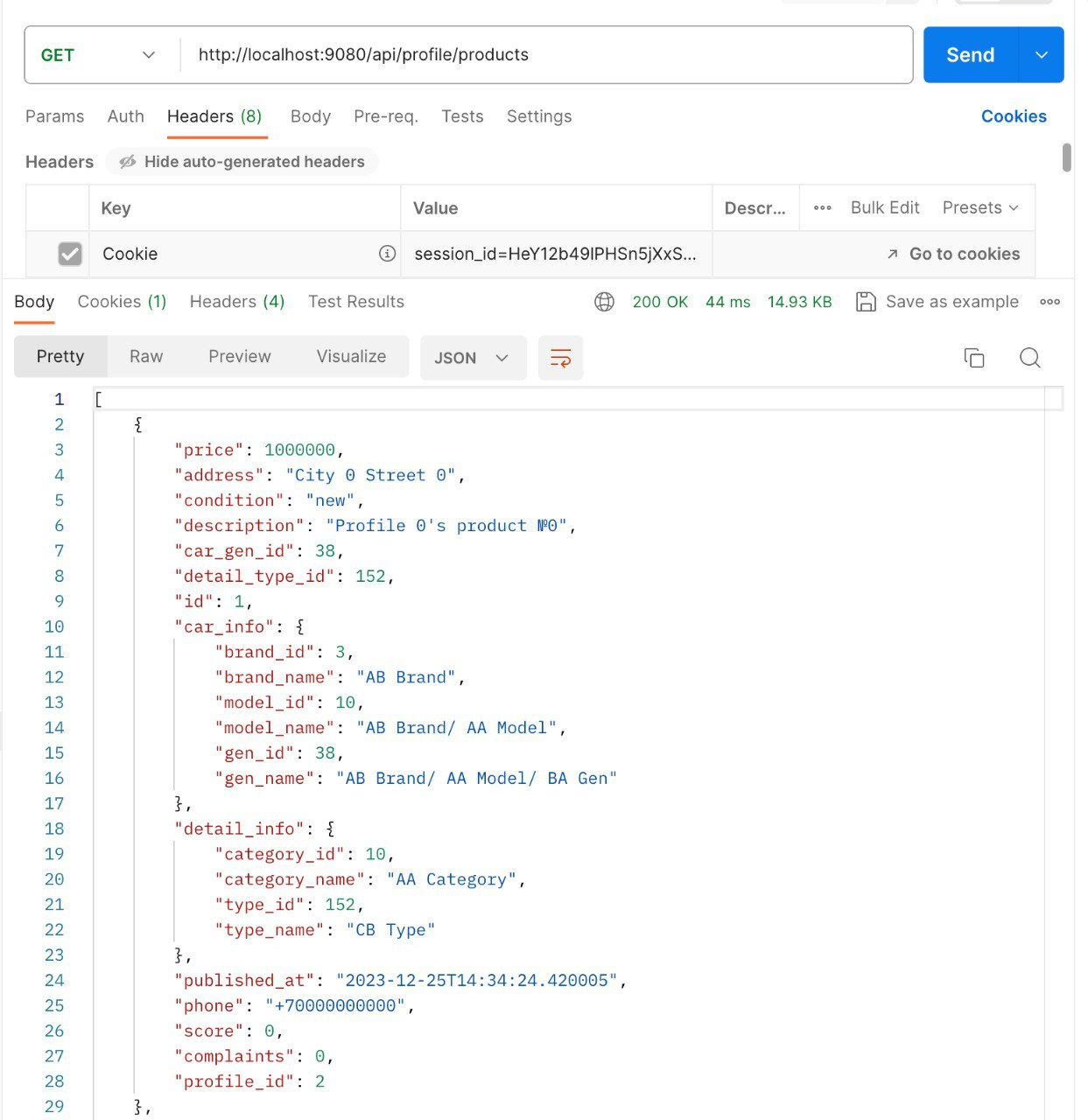


Рисунок 7 – Пример тестирования эндпоинта с помощью Postman

Кроме того, тестирование эндпоинтов иногда проводилось с помощью Swagger, предоставляемый FastAPI автоматически и позволяющий настроить отображаемую информацию [6]. На рисунке 8 представлен пример ручного тестирования эндпоинта с помощью Swagger.

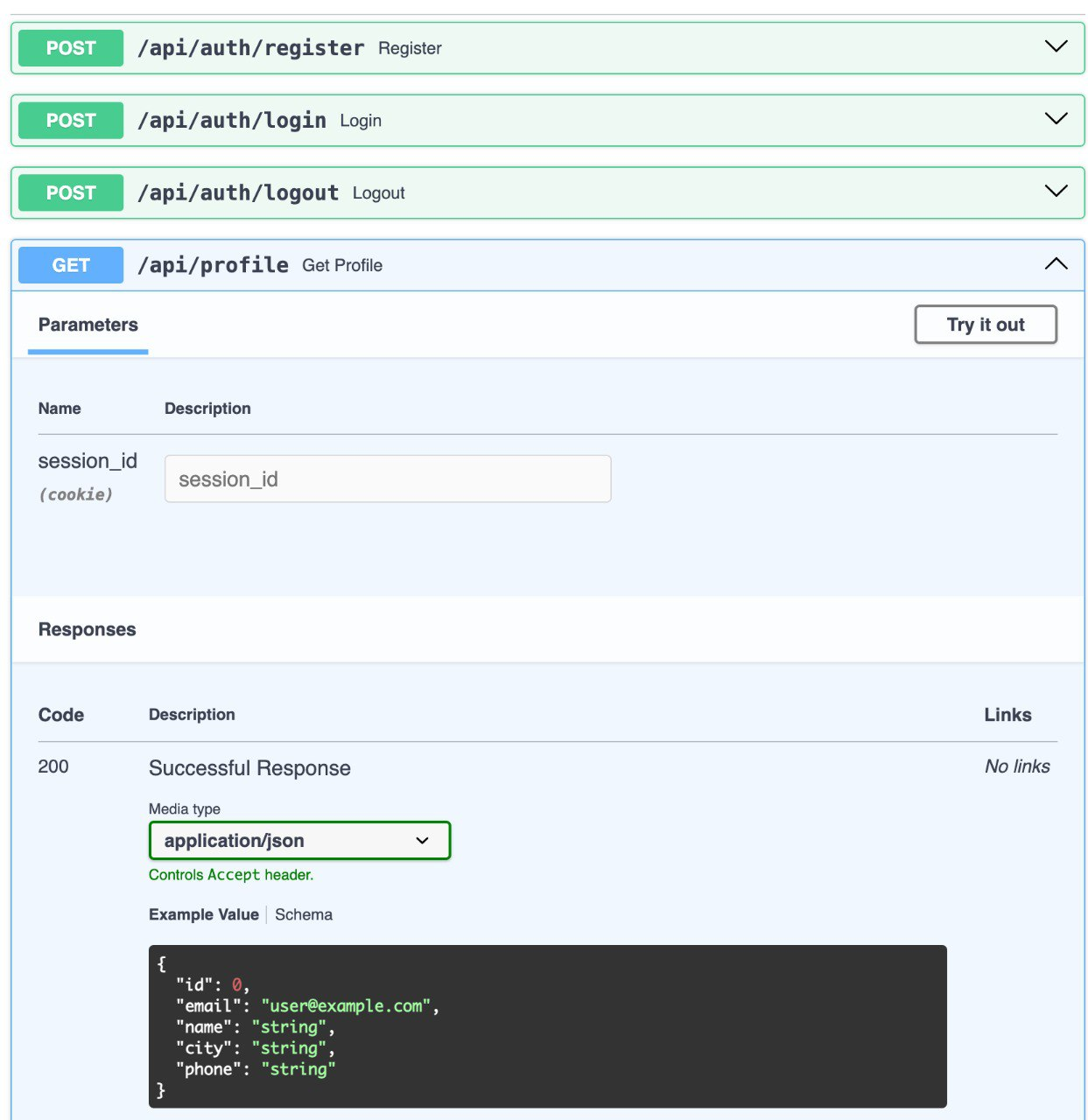


Рисунок 8 – Пример тестирования эндпоинта с помощью Swagger

Было также проведено ручное тестирование пользовательского интерфейса клиентской части приложения, в рамках которого были проверены все реализованные сценарии использования приложения.

# **ЗАКЛЮЧЕНИЕ**

В соответствии с техническим заданием было разработано приложение, которое состоит из браузерного клиента, сервера приложения, базы данных и key-value хранилища.

Была спроектирована схема базы данных, позволяющая реализовать всю необходимую бизнес-логику. Был реализован такой функционал, как поиск объявлений с возможностью задания настроек поиска, создание объявлений и их редактирование, просмотр и редактирование своего профиля, кроме того, была реализована административная панель для управления информацией, хранящейся в базе данных.

Произведено ручное тестирование как API, так и клиентской части приложения.

Дальнейшее развитие веб-сервиса может идти по пути наращивания функционала, улучшения клиентского опыта и обеспечения дополнительной безопасности для пользователей.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения. — СПб.: Питер, 2018 — 352 с.: ил. — (Серия «Библиотека программиста»).
2. The PostgreSQL Global Development Group, PostgreSQL [Электронный ресурс]. 1996-2022 URL: https://www.postgresql.org/ (дата обращения: 20.12.2023)
3. Макфарланд Дэвид. JavaScript. Подробное руководство. - М.: Эксмо, 2015. - 608 с.
4. Lokesh Gupta. What is REST [Электронный ресурс]. 2022 URL: https://restfulapi.net/ (дата обращения 20.12.2023)
5. Postman API [Электронный ресурс] 2022 URL: https://www.postman.com/api-documentation-tool/ (дата обращения 20.12.2023)
6. SmartBear Software. Swagger OpenAPI Specification [Электронный ресурс]. 2022 URL: https://swagger.io/specification/ (дата обращения 20.12.2022)