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

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №6**

по дисциплине «Интерфейсы прикладного программирования»

**Тема практической работы:**

**Создание и документирование API на основе спецификации OpenAPI**

**Студент группы** ИКБО-01-21 Маров Герман Андреевич

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

(подпись студента)

**Руководитель практической работы**  Зарипов Е.А.

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

(подпись руководителя)

Работа представлена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Москва 2023

**ОГЛАВЛЕНИЕ**

[Цель работы 3](#_Toc115999382)

[Ход работы 3](#_Toc115999383)

[Вывод 5](#_Toc115999384)

# Цель работы

Целью данной практической работы является знакомство студентов с основными подходами к построению API и инструментарием, используемый для ускорения процесса разработки API, в части документирования интерфейса прикладного программирования.

# Ход работы

В данной практической работе необходимо ознакомиться с техникой разработки документации и создания программных интерфейсов на основе спецификации OpenAPI. В качестве задания необходимо выбрать любую прикладную задачу, например связанной с тематикой компьютерных игр, компьютерного интернет магазина и т. д. и создать интерфейс прикладного программирования для данной задачи, используя для этого OpenAPI, ReDOC, prism. Для создания спецификации OpenAPI на языке YAML можно воспользоваться Swagger Editor (https://editor.swagger.io/) Для проверки правильности работы решения необходимо отправить соответствующий запрос, например, в Postman и сверить запрос с ReDoc в правой части html-страницы.

Для автоматического создания документации на основе спецификации OpenAPI, необходимо подключить одну из её реализаций – например Swagger, к текущему проекту. С помощью добавления необходимой зависимости в pom.xml (рисунок 1) сделаем это для Spring Boot-приложения.

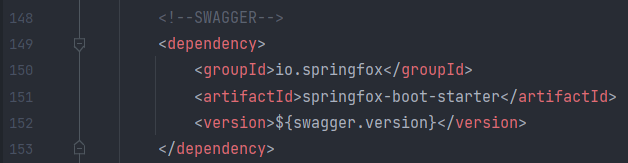


Рисунок 1 – Подключение Swagger’a к проекту

Далее создается конфигурационный файл для Swagger (листинг 1) и указываем базовую директорию, в которой находятся все контроллеры, которые требуют документирования. Весь анализ и конструирование документации в виде веб-страницы Swagger выполнит самостоятельно.

Листинг 1 – Код SwaggerConfig.java

import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.servlet.view.InternalResourceViewResolver;  
import springfox.documentation.builders.PathSelectors;  
import springfox.documentation.builders.RequestHandlerSelectors;  
import springfox.documentation.service.ApiKey;  
import springfox.documentation.service.AuthorizationScope;  
import springfox.documentation.service.SecurityReference;  
import springfox.documentation.spi.DocumentationType;  
import springfox.documentation.spi.service.contexts.SecurityContext;  
import springfox.documentation.spring.web.plugins.Docket;  
  
import java.util.List;  
  
@Configuration  
public class SwaggerConfig {  
  
 @Bean  
 public Docket api() {  
 return new Docket(DocumentationType.SWAGGER\_2)  
 .securityContexts(List.of(securityContext()))  
 .securitySchemes(List.of(apiKey()))  
 .select()  
 .apis(RequestHandlerSelectors.basePackage("com.senla.controller.restcontroller"))  
 .paths(PathSelectors.any())  
 .build();  
 }  
  
 @Bean  
 public InternalResourceViewResolver defaultViewResolver() {  
 return new InternalResourceViewResolver();  
 }  
  
 private ApiKey apiKey() {  
 return new ApiKey("JWT", "Authorization", "header");  
 }  
  
 private SecurityContext securityContext() {  
 return SecurityContext.builder().securityReferences(defaultAuth()).build();  
 }  
  
 private List<SecurityReference> defaultAuth() {  
 AuthorizationScope authorizationScope = new AuthorizationScope("global", "accessEverything");  
 AuthorizationScope[] authorizationScopes = new AuthorizationScope[1];  
 authorizationScopes[0] = authorizationScope;  
 return List.of(new SecurityReference("JWT", authorizationScopes));  
 }  
}

Помимо этого стоит отметить, что выбранное для примера приложение содержит авторизацию по JWT-токену, поэтому чтобы через swagger-ui можно было указать access token и отсылать запросы к защищенным эндпоинтам необходимо провести небольшую дополнительную конфигурацию, которая так же описана в листинге 1.

Проведём тестирование сгенерированной документации для нашего RESTful API (рисунок 1). Так, мы можем просмотреть структуру JSON-ответа на запрос по определенному эндпоинту вместе со статус-кодами этих ответов (рисунок 2).

Перед тем как выполнить запрос, необходимо указать access token, по которому происходит аутентификация (рисунок 3). Затем для примера мы можем выполнить запрос на получение информации об определенной точки проката электросамокатов по id (рисунок 4). После того как Swagger сформирует и выполнит этот запрос, мы можем наблюдать полученный ответ на рисунке 5.

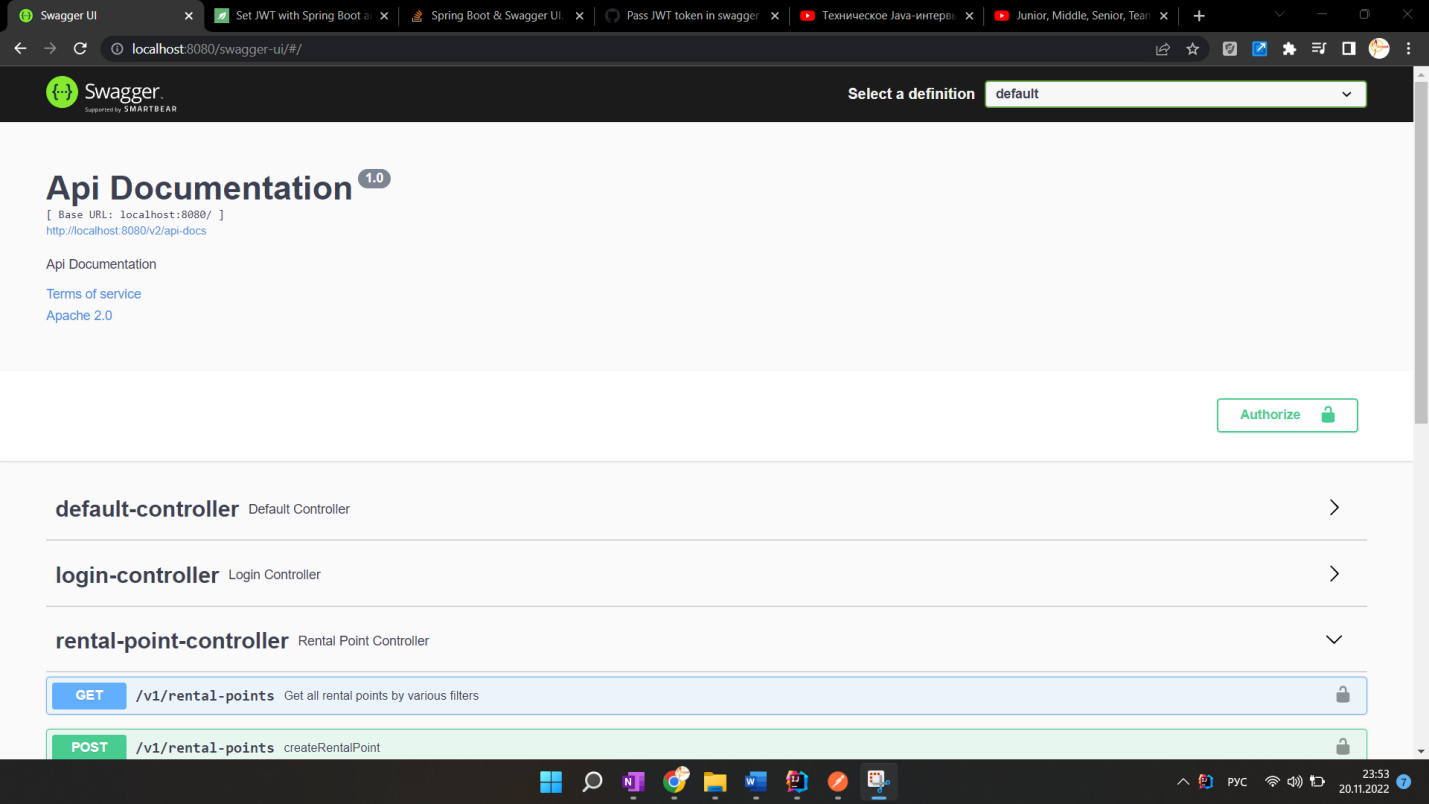


Рисунок 1 – Сгенерированная страница документации нашего API

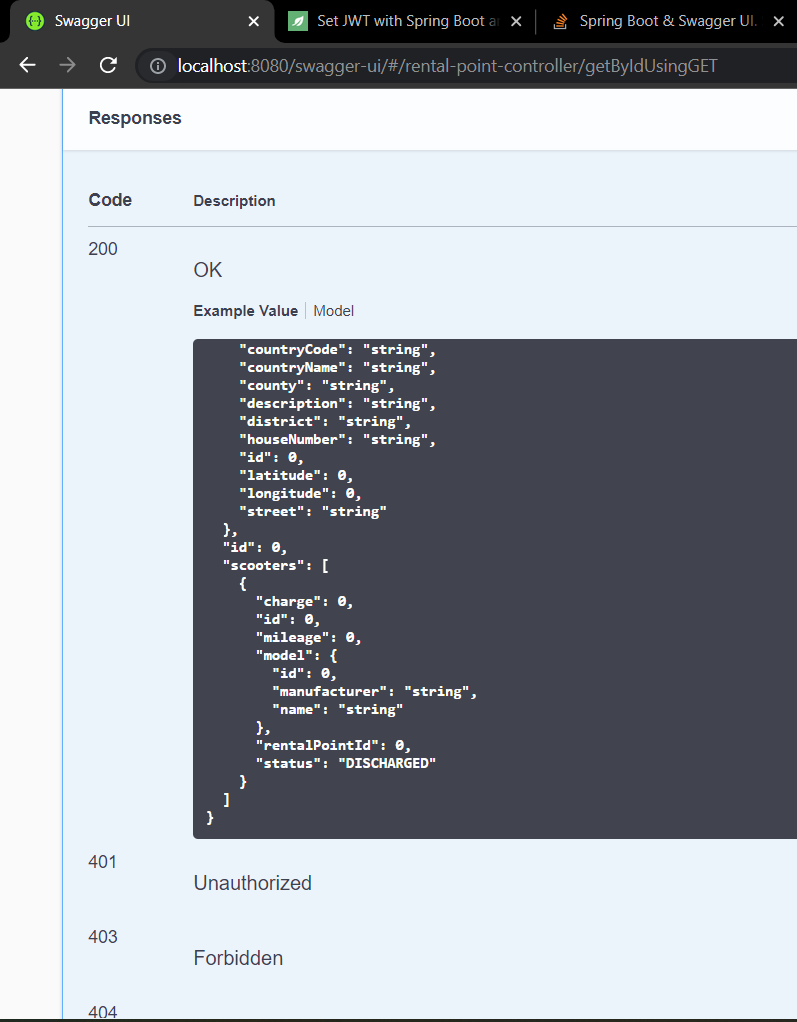


Рисунок 2 – Структура ответа при запросе по эндпоинту и статус-коды возможных ответов

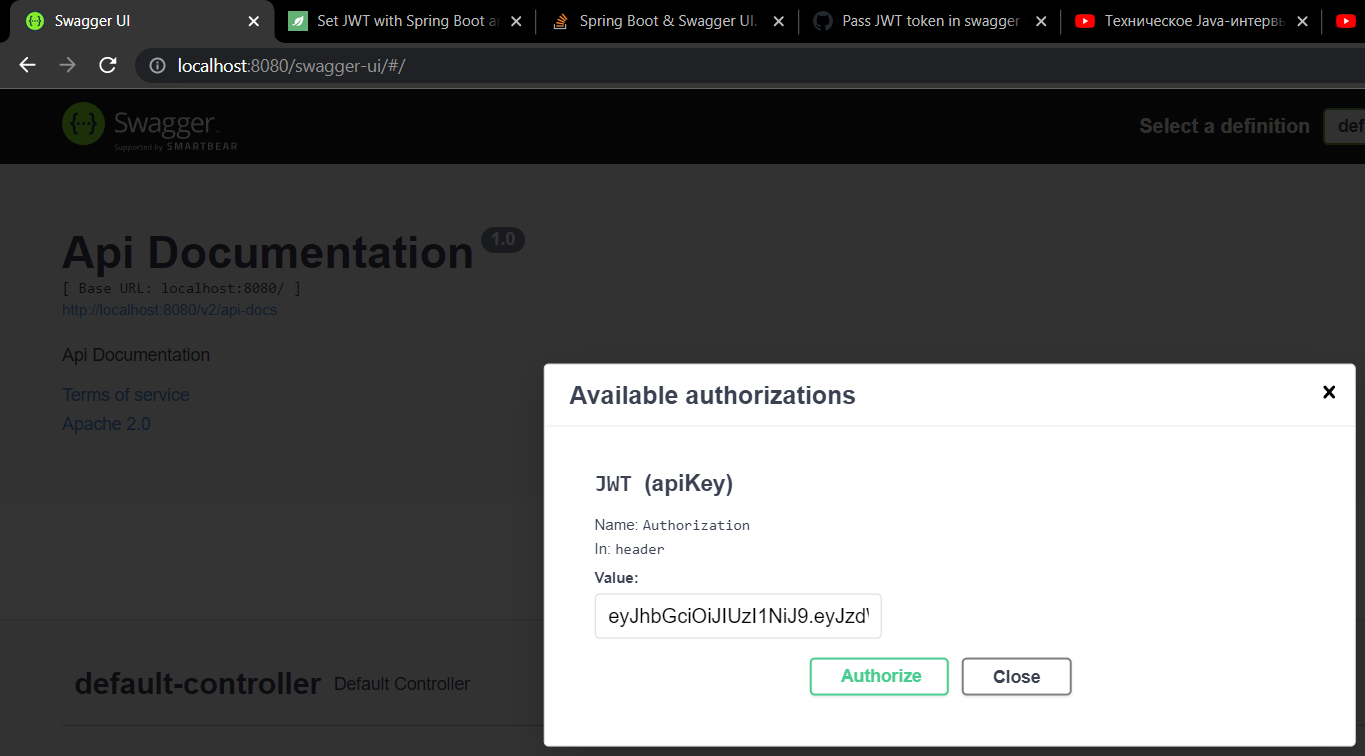


Рисунок 3 – Установка JWT-токена (access token) для авторизации запросов к текущему API

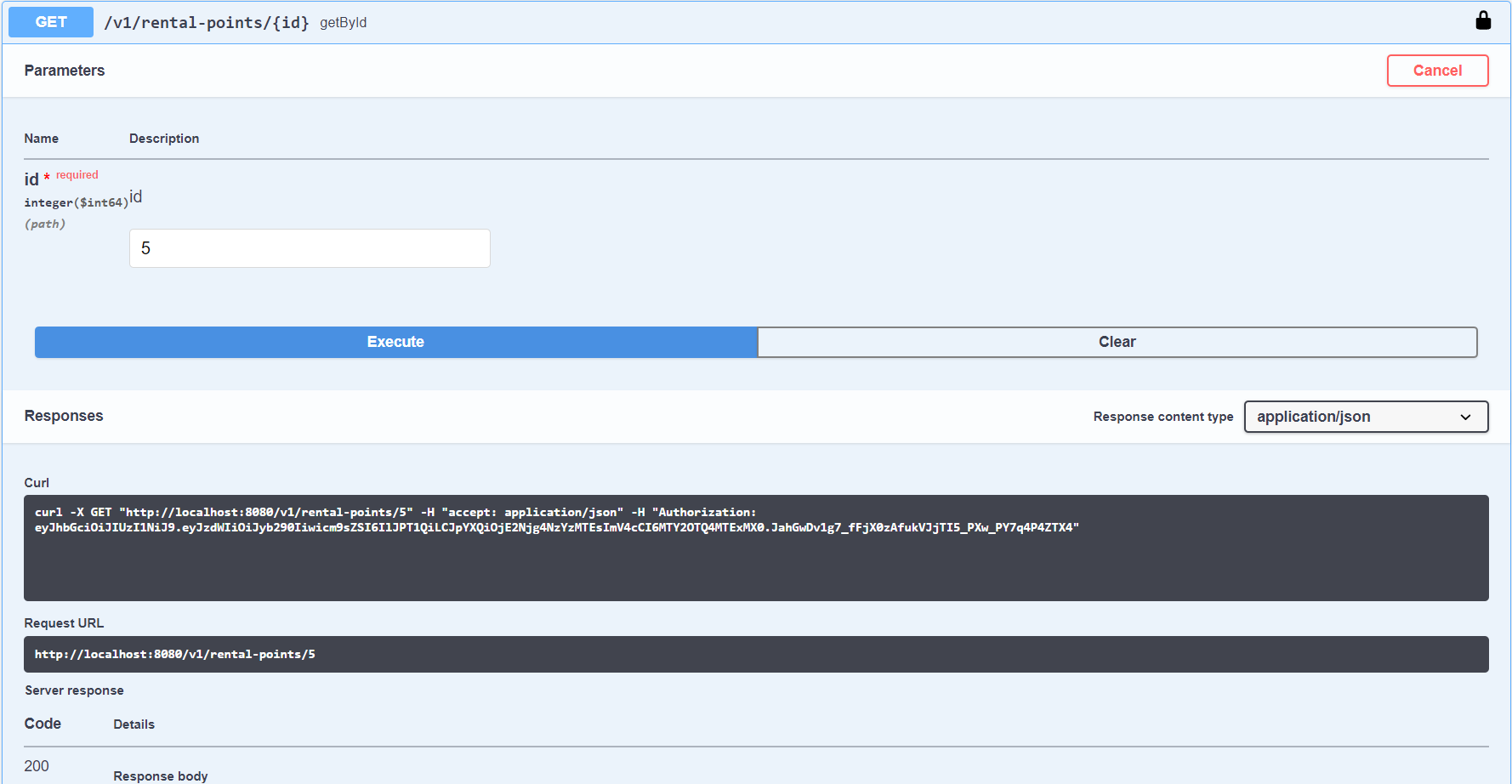


Рисунок 4 – Сгенерированный Swagger’ом GET-запрос

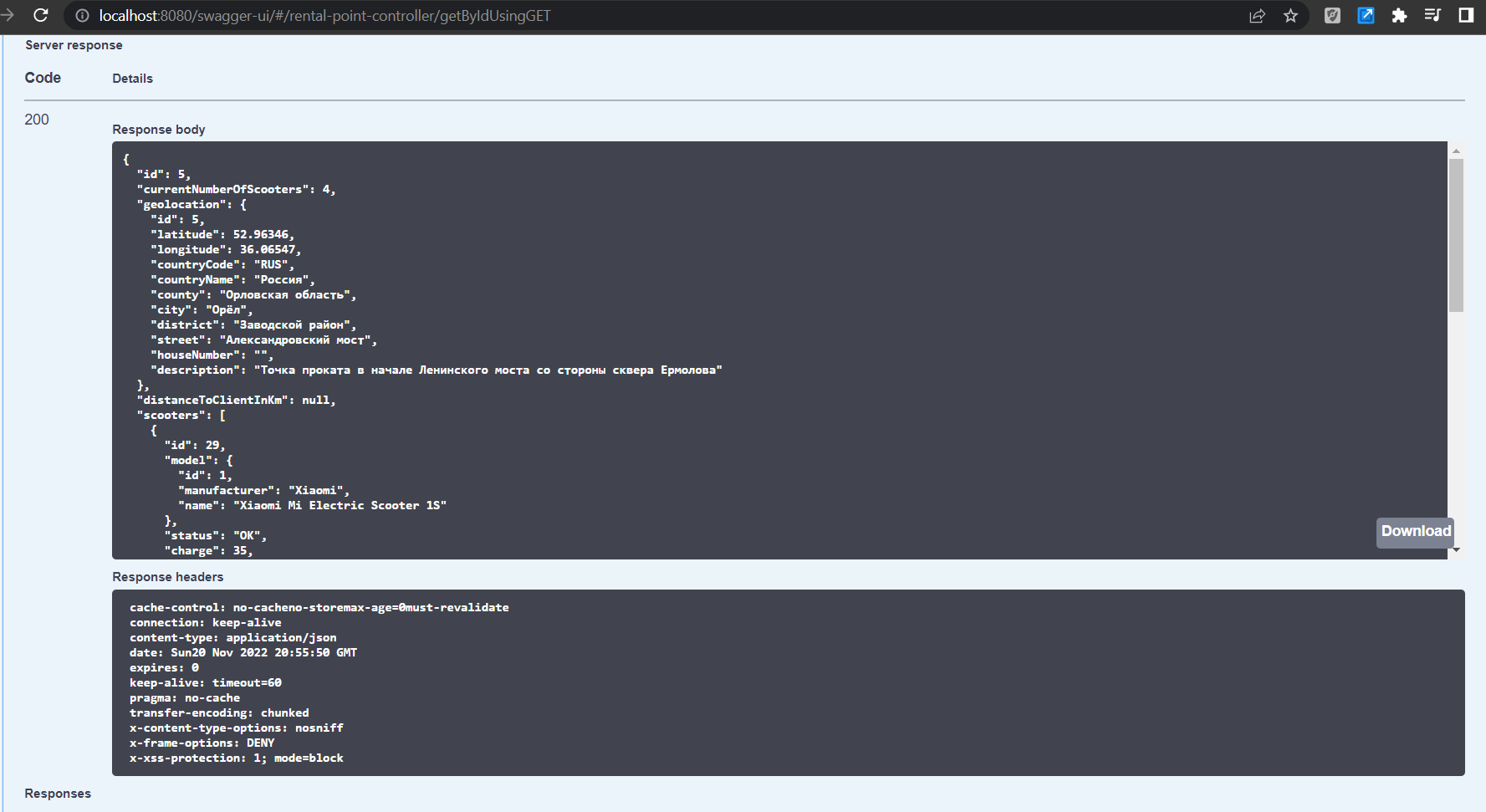


Рисунок 5 – Полученный результат GET-запроса через Swagger

# Вывод

В результате выполнения данной практической работы было проведено знакомство с основными подходами к построению API и инструментарием, используемый для ускорения процесса разработки API, в части документирования интерфейса прикладного программирования с помощью Swagger.