МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №7  
по дисциплине «Корпоративные информационные системы»

Тема: «Разработка REST-контроллера»

Вариант №4

Выполнил: Иванов Н. С.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc180325218)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc180325219)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 5](#_Toc180325220)

[КОД ПРОГРАММЫ 6](#_Toc180325221)

[ВЫВОД 10](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

Для данных по паркам, выборка из БД которых реализована в лабораторной работе №6, реализовать контроллер на все операции Добавления, Удаления, Изменения и Выборки данных. Для документации и проверки запросов использовать Swagger.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Разработано приложение для управления данными о парках с использованием Spring Boot, MyBatis и Swagger. Модель Park описывает сущность парка, включая такие поля, как идентификатор, название, местоположение, вместимость и статус открытости. Аннотации Jackson используются для автоматического преобразования данных между JSON и Java-объектами.

Интерфейс ParkMapper реализует взаимодействие с базой данных PostgreSQL, включая методы для добавления, обновления, выборки всех парков и удаления по идентификатору. Запросы описаны с использованием аннотаций MyBatis, а также с помощью XML-файла маппинга, где применяется resultMap для сопоставления данных базы данных с атрибутами модели.

Сервис ParkService обрабатывает бизнес-логику приложения. Он содержит методы для добавления нового парка, обновления существующего, получения списка всех парков и удаления парка по идентификатору. Сервис изолирует взаимодействие с базой данных, предоставляя высокоуровневый API для контроллера.

Контроллер ParkController реализует REST API для обработки HTTP-запросов. Для каждой операции добавлены Swagger-аннотации (@Tag, @Operation), что позволяет легко документировать и тестировать API. Контроллер поддерживает следующие операции:

* **POST** запрос для добавления нового парка.
* **PUT** запрос для обновления информации о парке.
* **GET** запрос для получения списка всех парков.
* **DELETE** запрос для удаления парка по идентификатору.

Конфигурация подключения к базе данных PostgreSQL описана в файле application.properties. Логирование запросов MyBatis включено для удобства отладки. Приложение запускается на порту 8081, предоставляя доступ к функционалу через REST API.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

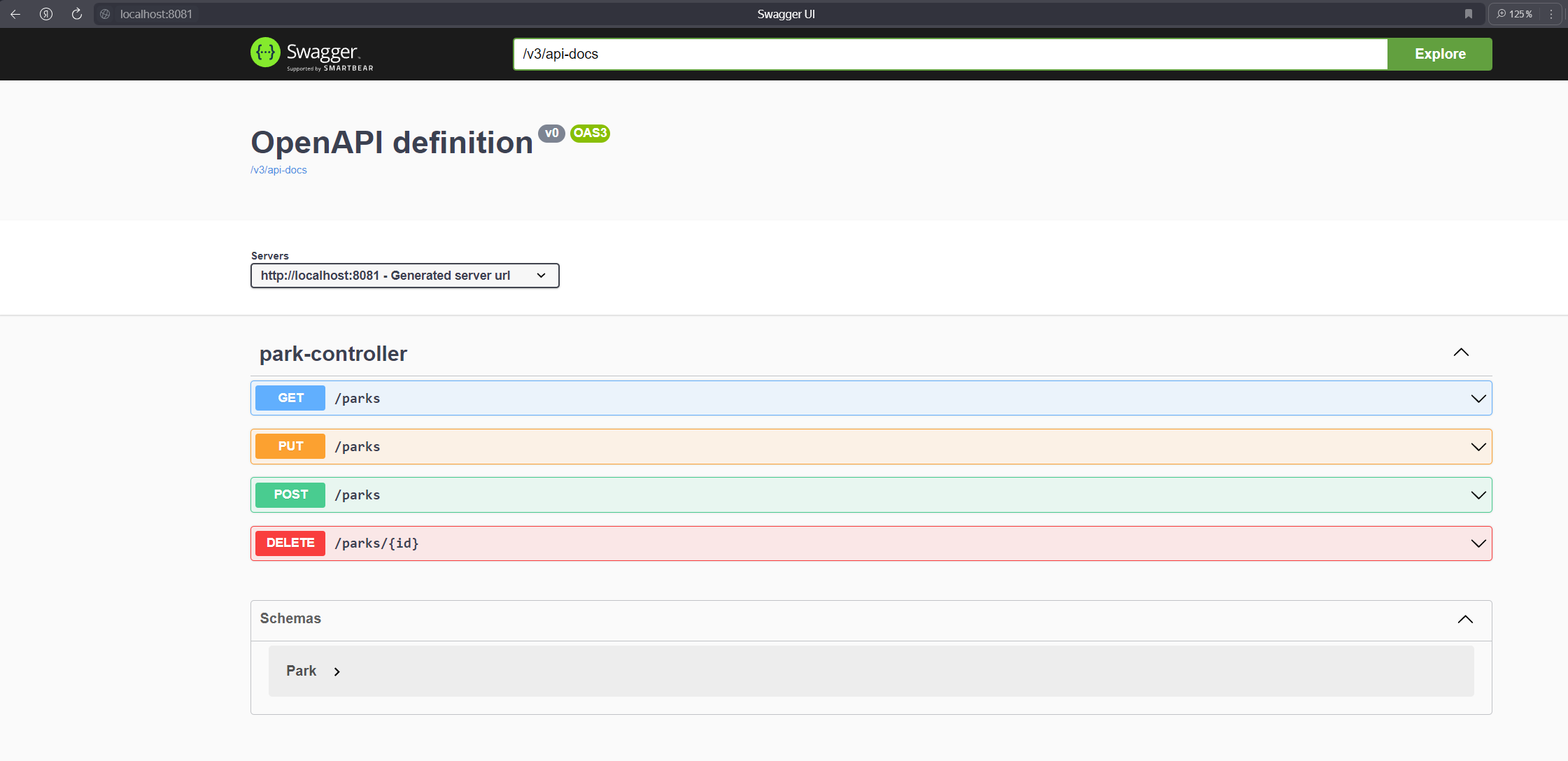


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package com.labsfrom6to8.Model;  
  
import com.fasterxml.jackson.annotation.JsonProperty;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
public class Park {  
  
 private Long id;  
  
 @JsonProperty("name")  
 private String name;  
  
 @JsonProperty("location")  
 private String location;  
  
 @JsonProperty("capacity")  
 private Integer capacity;  
  
 @JsonProperty("is\_open")  
 private Boolean isOpen;  
}

package com.labsfrom6to8.Mapper;  
  
import com.labsfrom6to8.Model.Park;  
import org.apache.ibatis.annotations.\*;  
  
import java.util.List;  
  
@Mapper  
public interface ParkMapper {  
  
 @Insert("INSERT INTO station\_parks (name, location, capacity, is\_open) VALUES (#{name}, #{location}, #{capacity}, #{isOpen})")  
 void insertPark(Park park);  
  
 @Update("UPDATE station\_parks SET name = #{name}, location = #{location}, capacity = #{capacity}, is\_open = #{isOpen} WHERE id = #{id}")  
 void updatePark(Park park);  
  
 @Select("SELECT \* FROM station\_parks")  
 @Results({  
 @Result(property = "isOpen", column = "is\_open")  
 })  
 List<Park> selectAllParks();  
  
 @Delete("DELETE FROM station\_parks WHERE id = #{id}")  
 void deletePark(Long id);  
}

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper  
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
  
<mapper namespace="com.example.Mapper.ParkMapper">  
  
 <insert id="insertPark" parameterType="com.example.Model.Park">  
 INSERT INTO station\_parks (name, location, capacity, is\_open)  
 VALUES (#{name}, #{location}, #{capacity}, #{isOpen})  
 </insert>  
  
 <update id="updatePark" parameterType="com.example.Model.Park">  
 UPDATE station\_parks  
 SET name = #{name},  
 location = #{location},  
 capacity = #{capacity},  
 is\_open = #{isOpen}  
 WHERE id = #{id}  
 </update>  
  
 <resultMap id="ParkResultMap" type="com.example.Model.Park">  
 <result property="id" column="id"/>  
 <result property="name" column="name"/>  
 <result property="location" column="location"/>  
 <result property="capacity" column="capacity"/>  
 <result property="isOpen" column="is\_open"/>  
 </resultMap>  
  
 <select id="selectAllParks" resultMap="ParkResultMap">  
 SELECT \* FROM station\_parks  
 </select>  
  
 <delete id="deletePark" parameterType="Long">  
 DELETE FROM station\_parks  
 WHERE id = #{id}  
 </delete>  
</mapper>

package com.labsfrom6to8.Service;  
  
import com.labsfrom6to8.Mapper.ParkMapper;  
import com.labsfrom6to8.Model.Park;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class ParkService {  
  
 private final ParkMapper parkMapper;  
  
 public ParkService(ParkMapper parkMapper) {  
 this.parkMapper = parkMapper;  
 }  
  
 public void addPark(Park park) {  
 parkMapper.insertPark(park);  
 }  
  
 public void updatePark(Park park) {  
 parkMapper.updatePark(park);  
 }  
  
 public List<Park> getAllParks() {  
 return parkMapper.selectAllParks();  
 }  
  
 public void deleteParkById(Long id) {  
 parkMapper.deletePark(id);  
 }  
}

package com.labsfrom6to8.Controller;

import com.labsfrom6to8.Model.Park;

import com.labsfrom6to8.Service.ParkService;

import io.swagger.v3.oas.annotations.Operation;

import io.swagger.v3.oas.annotations.tags.Tag;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@RestController

@RequestMapping("/parks")

@Tag(name = "Park API", description = "API для управления парками")

public class ParkController {

private final ParkService parkService;

public ParkController(ParkService parkService) {

this.parkService = parkService;

}

@PostMapping

@Operation(summary = "Добавить новый парк", description = "Добавляет новый парк в систему")

public void addPark(@RequestBody Park park) {

parkService.addPark(park);

}

@PutMapping

@Operation(summary = "Обновить парк", description = "Обновляет информацию о парке")

public void updatePark(@RequestBody Park park) {

parkService.updatePark(park);

}

@GetMapping

@Operation(summary = "Получить список всех парков", description = "Возвращает список всех парков в системе")

public List<Park> getAllParks() {

return parkService.getAllParks();

}

@DeleteMapping("/{id}")

@Operation(summary = "Удалить парк", description = "Удаляет парк по его идентификатору")

public void deletePark(@PathVariable Long id) {

parkService.deleteParkById(id);

}

}

# **ВЫВОД**

Приложение предоставляет удобный и функциональный инструмент для управления данными о парках. REST API делает приложение легко интегрируемым с другими системами. Использование Swagger упрощает тестирование и документирование API, а MyBatis обеспечивает гибкое взаимодействие с базой данных. Приложение выполнено в модульной архитектуре, что упрощает его поддержку и масштабирование.