МИНИСТЕРСТВО ТРАНСПОРТА РФ

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

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №6  
по дисциплине «Корпоративные информационные системы»

Тема: «Работа с библиотекой MyBatis»

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 9](#_Toc180325222)

# **ПОСТАНОВКА ЗАДАЧИ**

Описать класс Парк. В БД Postgres создать таблицу «парки». С помощью MyBatis реализовать операции Добавления, Удаления, Изменения и Выборки данных из таблицы «парки». Продемонстрировать операции работы с БД.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Для управления данными о парках железнодорожных станций было создано приложение с использованием Spring Boot и MyBatis. На уровне модели определён класс Park, который включает свойства идентификатора, имени парка, местоположения, вместимости и статуса открытия. Этот класс аннотирован для работы с JSON, что позволяет эффективно обрабатывать данные на уровне контроллера.

Интерфейс ParkMapper определяет методы для выполнения CRUD-операций с базой данных. Для работы с запросами используется как аннотации, так и XML-файл маппинга, который включает resultMap для отображения данных из базы на модель. Логика бизнес-уровня инкапсулирована в классе ParkService, который взаимодействует с ParkMapper для выполнения операций добавления, обновления, выборки и удаления записей.

Контроллер ParkController предоставляет REST API для управления парками. Он поддерживает POST-запросы для добавления новых парков, PUT-запросы для их обновления, GET-запросы для получения всех данных и DELETE-запросы для удаления парка по идентификатору. Все запросы логируются благодаря настройке MyBatis в файле application.properties.

Подключение к PostgreSQL настроено через параметры в конфигурационном файле, включая URL базы данных, имя пользователя и пароль. Приложение работает на порту 8081, предоставляя удобный доступ к сервисам через REST API.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

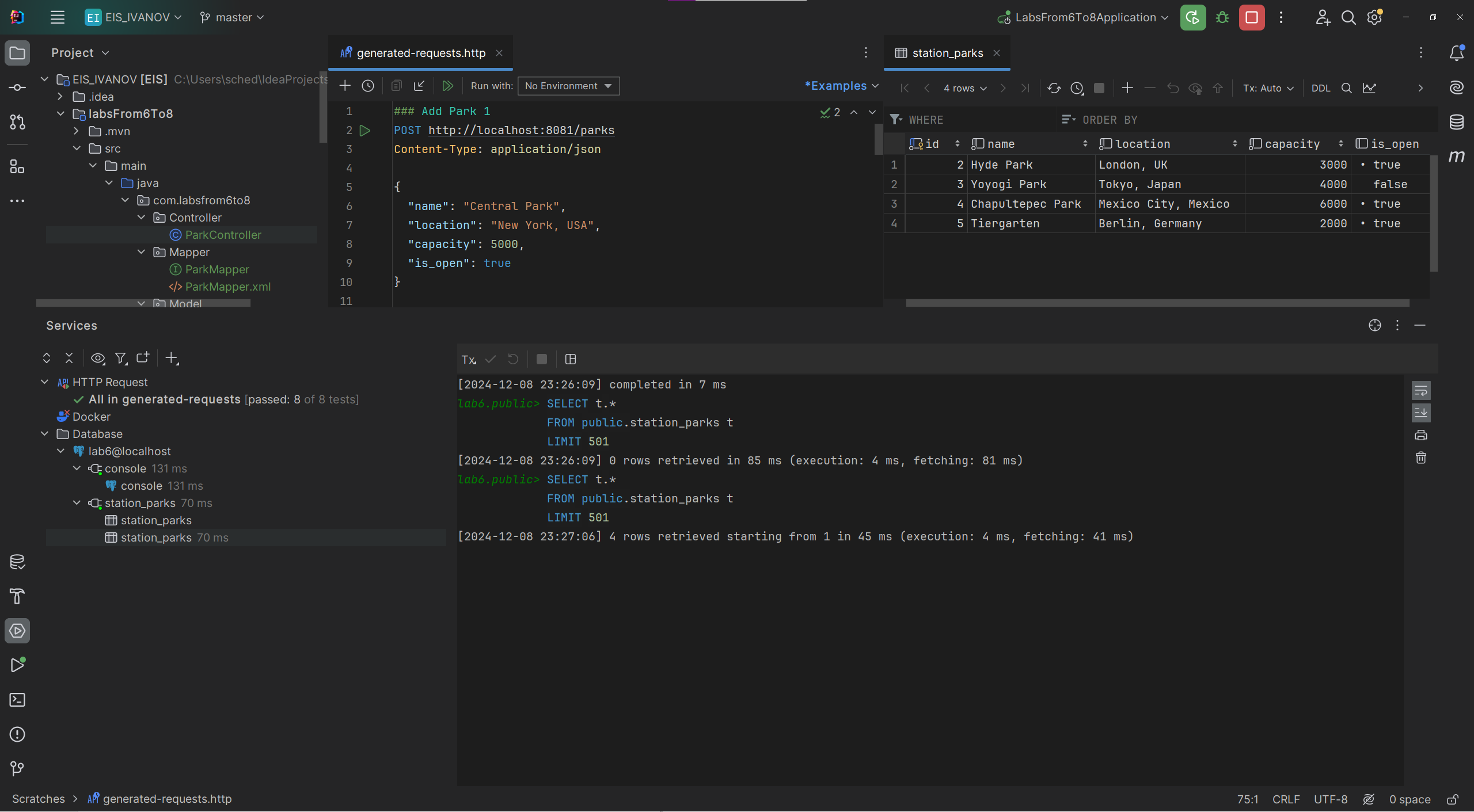


Рисунок 1 – Результат работы программы

# **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 org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/parks")  
public class ParkController {  
  
 private final ParkService parkService;  
  
 public ParkController(ParkService parkService) {  
 this.parkService = parkService;  
 }  
  
 @PostMapping  
 public void addPark(@RequestBody Park park) {  
 parkService.addPark(park);  
 }  
  
 @PutMapping  
 public void updatePark(@RequestBody Park park) {  
 parkService.updatePark(park);  
 }  
  
 @GetMapping  
 public List<Park> getAllParks() {  
 return parkService.getAllParks();  
 }  
  
 @DeleteMapping("/{id}")  
 public void deletePark(@PathVariable Long id) {  
 parkService.deleteParkById(id);  
 }  
}

spring.application.name=labsFrom6To8  
  
spring.datasource.url=jdbc:postgresql://localhost:5432/lab6  
spring.datasource.username=postgres  
spring.datasource.password=123  
spring.datasource.driver-class-name=org.postgresql.Driver  
  
logging.level.org.mybatis=DEBUG  
logging.level.org.springframework.jdbc.core=DEBUG  
logging.level.org.springframework.jdbc.datasource=DEBUG  
logging.level.org.apache.ibatis.logging.stdout.StdOutImpl=TRACE  
mybatis.configuration.log-impl=org.apache.ibatis.logging.stdout.StdOutImpl  
  
spring.mvc.view.prefix=  
spring.mvc.view.suffix=  
  
server.port=8081

# **ВЫВОД**

Созданное приложение предоставляет удобный инструмент для управления парками железнодорожных станций. Использование MyBatis позволяет эффективно работать с базой данных, а REST API обеспечивает гибкость и возможность интеграции с другими сервисами. Решение структурировано, легко расширяемо и поддерживаемо, что делает его подходящим для различных бизнес-сценариев.