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 №19

Выполнил: Любченкова О. А.

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 железным дорогам, 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 железнодорожных путях с использованием Spring Boot, MyBatis и Swagger. Модель Railway представляет сущность железнодорожного пути, включая такие поля, как идентификатор, название, длина в километрах, страна и статус эксплуатации. Аннотации Jackson обеспечивают преобразование данных между объектами Java и JSON.

Интерфейс RailwayMapper реализует взаимодействие с базой данных PostgreSQL. Методы интерфейса выполняют операции добавления, обновления, выборки всех железнодорожных путей и удаления записи по идентификатору. Используются аннотации MyBatis для описания SQL-запросов. В XML-файле маппинга настроен resultMap, который сопоставляет колонки базы данных с полями модели.

Сервис RailwayService инкапсулирует бизнес-логику приложения. Он предоставляет методы для работы с RailwayMapper, включая добавление, обновление, получение и удаление данных о железнодорожных путях. Сервис позволяет изолировать обработку данных от логики контроллера.

Контроллер RailwayController предоставляет REST API для управления железнодорожными путями. Используя Swagger, API документируется для упрощения тестирования и интеграции. Контроллер поддерживает:

* **POST** запрос для добавления нового железнодорожного пути.
* **PUT** запрос для обновления информации.
* **GET** запрос для получения списка всех путей.
* **DELETE** запрос для удаления записи по идентификатору.

Конфигурация подключения к базе данных PostgreSQL указана в файле application.properties. Включено логирование MyBatis для упрощения отладки. Приложение работает на порту 8081.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

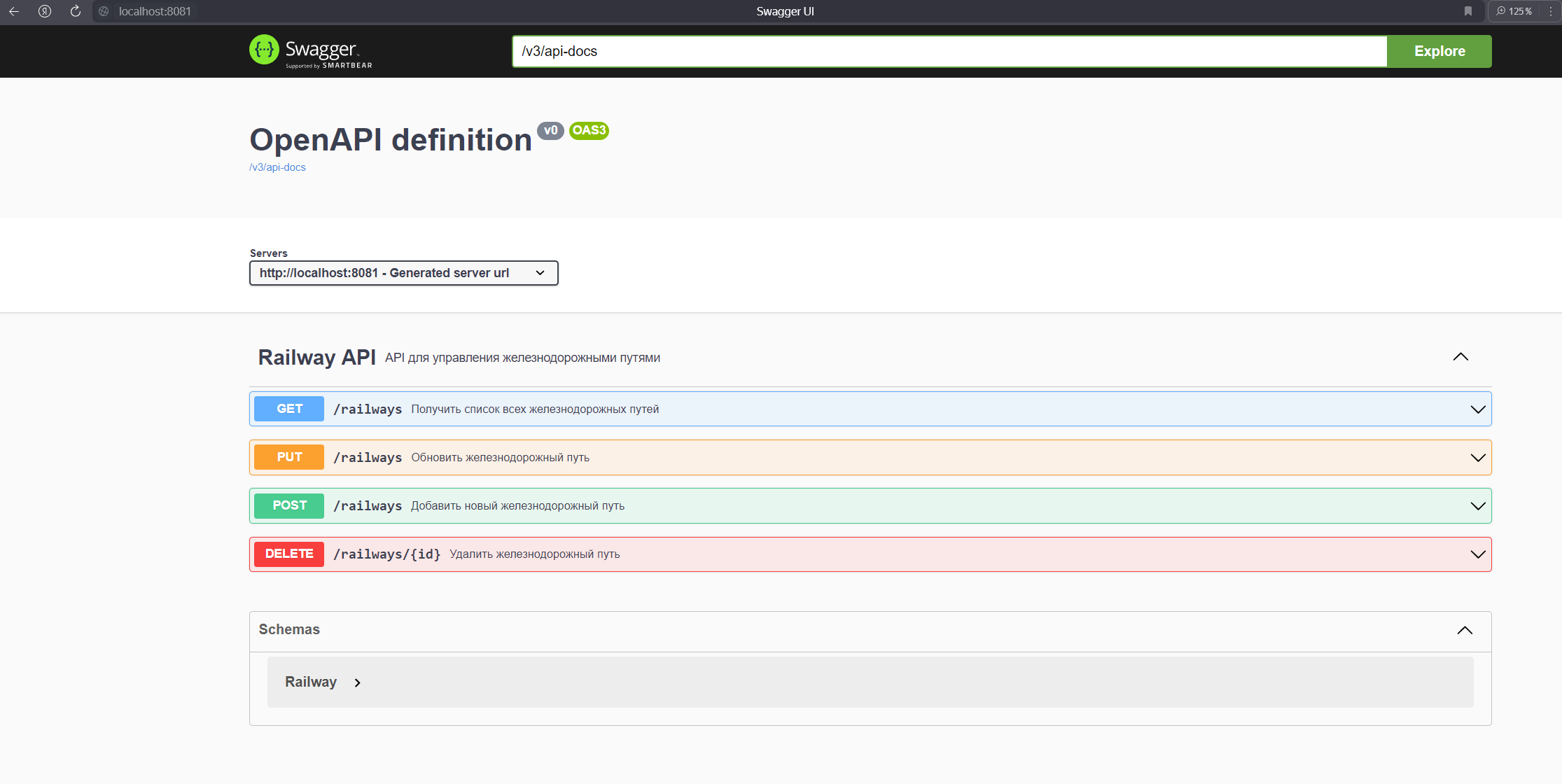


Рисунок 1 – Результат работы программы

# **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 Railway {  
  
 private Long id;  
  
 @JsonProperty("name")  
 private String name;  
  
 @JsonProperty("length\_km")  
 private Double lengthKm;  
  
 @JsonProperty("country")  
 private String country;  
  
 @JsonProperty("is\_operational")  
 private Boolean isOperational;  
}

package com.labsfrom6to8.Mapper;  
  
import com.labsfrom6to8.Model.Railway;  
import org.apache.ibatis.annotations.\*;  
  
import java.util.List;  
  
@Mapper  
public interface RailwayMapper {  
  
 @Insert("INSERT INTO railways (name, length\_km, country, is\_operational) VALUES (#{name}, #{lengthKm}, #{country}, #{isOperational})")  
 void insertRailway(Railway railway);  
  
 @Update("UPDATE railways SET name = #{name}, length\_km = #{lengthKm}, country = #{country}, is\_operational = #{isOperational} WHERE id = #{id}")  
 void updateRailway(Railway railway);  
  
 @Select("SELECT \* FROM railways")  
 @Results({  
 @Result(property = "lengthKm", column = "length\_km"),  
 @Result(property = "isOperational", column = "is\_operational")  
 })  
 List<Railway> selectAllRailways();  
  
 @Delete("DELETE FROM railways WHERE id = #{id}")  
 void deleteRailway(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.RailwayMapper">  
  
 <insert id="insertRailway" parameterType="com.example.Model.Railway">  
 INSERT INTO railways (name, length\_km, country, is\_operational)  
 VALUES (#{name}, #{lengthKm}, #{country}, #{isOperational})  
 </insert>  
  
 <update id="updateRailway" parameterType="com.example.Model.Railway">  
 UPDATE railways  
 SET name = #{name},  
 length\_km = #{lengthKm},  
 country = #{country},  
 is\_operational = #{isOperational}  
 WHERE id = #{id}  
 </update>  
  
 <resultMap id="RailwayResultMap" type="com.example.Model.Railway">  
 <result property="id" column="id"/>  
 <result property="name" column="name"/>  
 <result property="lengthKm" column="length\_km"/>  
 <result property="country" column="country"/>  
 <result property="isOperational" column="is\_operational"/>  
 </resultMap>  
  
 <select id="selectAllRailways" resultMap="RailwayResultMap">  
 SELECT \* FROM railways  
 </select>  
  
 <delete id="deleteRailway" parameterType="Long">  
 DELETE FROM railways  
 WHERE id = #{id}  
 </delete>  
</mapper>

package com.labsfrom6to8.Service;  
  
import com.labsfrom6to8.Mapper.RailwayMapper;  
import com.labsfrom6to8.Model.Railway;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class RailwayService {  
  
 private final RailwayMapper railwayMapper;  
  
 public RailwayService(RailwayMapper railwayMapper) {  
 this.railwayMapper = railwayMapper;  
 }  
  
 public void addRailway(Railway railway) {  
 railwayMapper.insertRailway(railway);  
 }  
  
 public void updateRailway(Railway railway) {  
 railwayMapper.updateRailway(railway);  
 }  
  
 public List<Railway> getAllRailways() {  
 return railwayMapper.selectAllRailways();  
 }  
  
 public void deleteRailwayById(Long id) {  
 railwayMapper.deleteRailway(id);  
 }  
}

package com.labsfrom6to8.Controller;

import com.labsfrom6to8.Model.Railway;

import com.labsfrom6to8.Service.RailwayService;

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("/railways")

@Tag(name = "Railway API", description = "API для управления железнодорожными путями")

public class RailwayController {

private final RailwayService railwayService;

public RailwayController(RailwayService railwayService) {

this.railwayService = railwayService;

}

@PostMapping

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

public void addRailway(@RequestBody Railway railway) {

railwayService.addRailway(railway);

}

@PutMapping

@Operation(summary = "Обновить железнодорожный путь", description = "Обновляет информацию о железнодорожном пути")

public void updateRailway(@RequestBody Railway railway) {

railwayService.updateRailway(railway);

}

@GetMapping

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

public List<Railway> getAllRailways() {

return railwayService.getAllRailways();

}

@DeleteMapping("/{id}")

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

public void deleteRailway(@PathVariable Long id) {

railwayService.deleteRailwayById(id);

}

}

# **ВЫВОД**

Приложение предоставляет функциональный API для управления данными о железнодорожных путях. Использование Swagger упрощает тестирование и документирование API, а MyBatis позволяет эффективно работать с базой данных. Архитектура приложения модульна, что облегчает поддержку, расширение и интеграцию с другими системами.