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 №12

Выполнил: Бугаева А. Л.

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**

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

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Разработано приложение для управления данными о компьютерах с использованием Spring Boot, MyBatis и Swagger. Модель Computer представляет сущность компьютера с такими атрибутами, как идентификатор, бренд, модель, процессор, оперативная память, объём хранилища и статус доступности. Аннотации Jackson обеспечивают преобразование данных между JSON и Java-объектами.

Интерфейс ComputerMapper содержит методы для выполнения операций с базой данных PostgreSQL, таких как добавление, обновление, выборка всех компьютеров и удаление записи по идентификатору. Для выполнения запросов используется как аннотированный SQL-код, так и XML-файл маппинга, где настроен resultMap для отображения колонок базы данных в свойства модели.

Сервис ComputerService предоставляет бизнес-логику для взаимодействия с ComputerMapper. Реализованы методы для добавления, обновления, получения списка всех компьютеров и удаления записи по идентификатору. Сервис отвечает за обработку данных перед их передачей в базу данных или перед возвращением результата.

Контроллер ComputerController предоставляет REST API для работы с клиентами. С использованием Swagger аннотаций (@Tag и @Operation) реализовано документирование API, что упрощает интеграцию и тестирование. API поддерживает следующие запросы:

POST для добавления нового компьютера.

PUT для обновления информации о компьютере.

GET для получения списка всех компьютеров.

DELETE для удаления записи по идентификатору.

Конфигурация подключения к базе данных PostgreSQL описана в application.properties. Включено логирование запросов MyBatis для упрощения отладки. Приложение запускается на порту 8081.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

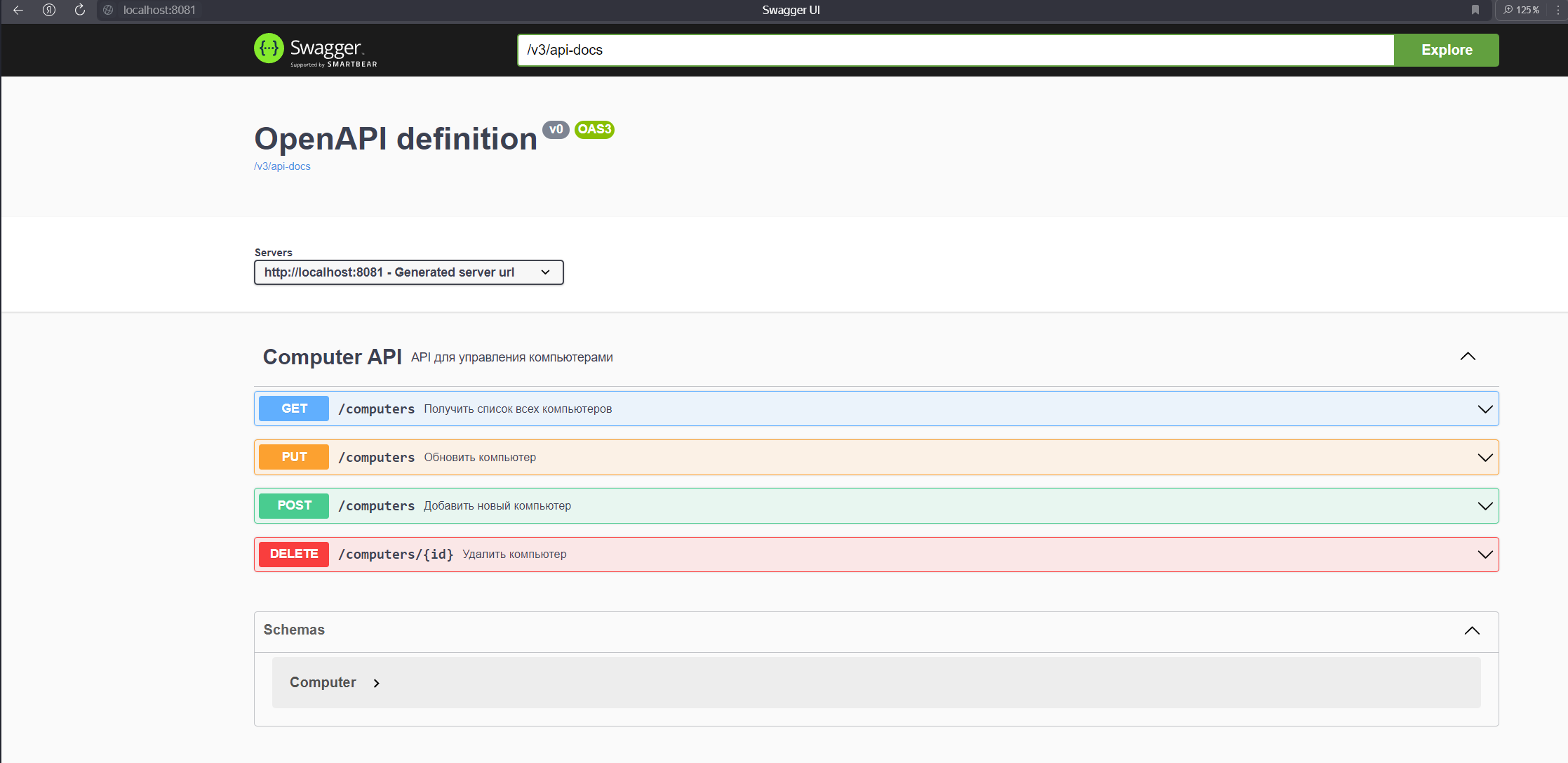


Рисунок 1 – Результат работы программы

# **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 Computer {  
  
 private Long id;  
  
 @JsonProperty("brand")  
 private String brand;  
  
 @JsonProperty("model")  
 private String model;  
  
 @JsonProperty("processor")  
 private String processor;  
  
 @JsonProperty("ram")  
 private Integer ram;  
  
 @JsonProperty("storage")  
 private Integer storage;  
  
 @JsonProperty("is\_available")  
 private Boolean isAvailable;  
}

package com.labsfrom6to8.Mapper;  
  
import com.labsfrom6to8.Model.Computer;  
import org.apache.ibatis.annotations.\*;  
  
import java.util.List;  
  
@Mapper  
public interface ComputerMapper {  
  
 @Insert("INSERT INTO computers (brand, model, processor, ram, storage, is\_available) VALUES (#{brand}, #{model}, #{processor}, #{ram}, #{storage}, #{isAvailable})")  
 void insertComputer(Computer computer);  
  
 @Update("UPDATE computers SET brand = #{brand}, model = #{model}, processor = #{processor}, ram = #{ram}, storage = #{storage}, is\_available = #{isAvailable} WHERE id = #{id}")  
 void updateComputer(Computer computer);  
  
 @Select("SELECT \* FROM computers")  
 @Results({  
 @Result(property = "isAvailable", column = "is\_available")  
 })  
 List<Computer> selectAllComputers();  
  
 @Delete("DELETE FROM computers WHERE id = #{id}")  
 void deleteComputer(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.ComputerMapper">  
  
 <insert id="insertComputer" parameterType="com.example.Model.Computer">  
 INSERT INTO computers (brand, model, processor, ram, storage, is\_available)  
 VALUES (#{brand}, #{model}, #{processor}, #{ram}, #{storage}, #{isAvailable})  
 </insert>  
  
 <update id="updateComputer" parameterType="com.example.Model.Computer">  
 UPDATE computers  
 SET brand = #{brand},  
 model = #{model},  
 processor = #{processor},  
 ram = #{ram},  
 storage = #{storage},  
 is\_available = #{isAvailable}  
 WHERE id = #{id}  
 </update>  
  
 <resultMap id="ComputerResultMap" type="com.example.Model.Computer">  
 <result property="id" column="id"/>  
 <result property="brand" column="brand"/>  
 <result property="model" column="model"/>  
 <result property="processor" column="processor"/>  
 <result property="ram" column="ram"/>  
 <result property="storage" column="storage"/>  
 <result property="isAvailable" column="is\_available"/>  
 </resultMap>  
  
 <select id="selectAllComputers" resultMap="ComputerResultMap">  
 SELECT \* FROM computers  
 </select>  
  
 <delete id="deleteComputer" parameterType="Long">  
 DELETE FROM computers  
 WHERE id = #{id}  
 </delete>  
</mapper>

package com.labsfrom6to8.Service;  
  
import com.labsfrom6to8.Mapper.ComputerMapper;  
import com.labsfrom6to8.Model.Computer;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class ComputerService {  
  
 private final ComputerMapper computerMapper;  
  
 public ComputerService(ComputerMapper computerMapper) {  
 this.computerMapper = computerMapper;  
 }  
  
 public void addComputer(Computer computer) {  
 computerMapper.insertComputer(computer);  
 }  
  
 public void updateComputer(Computer computer) {  
 computerMapper.updateComputer(computer);  
 }  
  
 public List<Computer> getAllComputers() {  
 return computerMapper.selectAllComputers();  
 }  
  
 public void deleteComputerById(Long id) {  
 computerMapper.deleteComputer(id);  
 }  
}

package com.labsfrom6to8.Controller;

import com.labsfrom6to8.Model.Computer;

import com.labsfrom6to8.Service.ComputerService;

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("/computers")

@Tag(name = "Computer API", description = "API для управления компьютерами")

public class ComputerController {

private final ComputerService computerService;

public ComputerController(ComputerService computerService) {

this.computerService = computerService;

}

@PostMapping

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

public void addComputer(@RequestBody Computer computer) {

computerService.addComputer(computer);

}

@PutMapping

@Operation(summary = "Обновить компьютер", description = "Обновляет информацию о компьютере")

public void updateComputer(@RequestBody Computer computer) {

computerService.updateComputer(computer);

}

@GetMapping

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

public List<Computer> getAllComputers() {

return computerService.getAllComputers();

}

@DeleteMapping("/{id}")

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

public void deleteComputer(@PathVariable Long id) {

computerService.deleteComputerById(id);

}

}

# **ВЫВОД**

Приложение представляет собой удобный инструмент для управления данными о компьютерах. REST API позволяет интегрировать приложение с другими системами. Использование Swagger делает API понятным и легко документируемым. Благодаря модульной архитектуре приложение хорошо масштабируется и легко поддерживается.