Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №1

по дисциплине «Программирование на языках высокого уровня»

«Погода»

Выполнил: Снитко Д.А. Проверил: Скиба И.Г.

гр.250501

Минск 2024

**1. Постановка задачи**

Создать и запустить локально простейший веб/REST сервис, используя любой открытый пример с использованием Java stack: Spring (Spring Boot)/Maven/Gradle/Jersey/Spring MVC. Добавить GET ендпоинт, принимающий входные параметры в качестве queryParams в URL согласно варианту, и возвращающий любой hard-coded результат в виде JSON согласно варианту.

**2. Структура проекта**

В проекте используется послойная архитектура из нескольких пакетов,

которые отвечают за определенные функции.

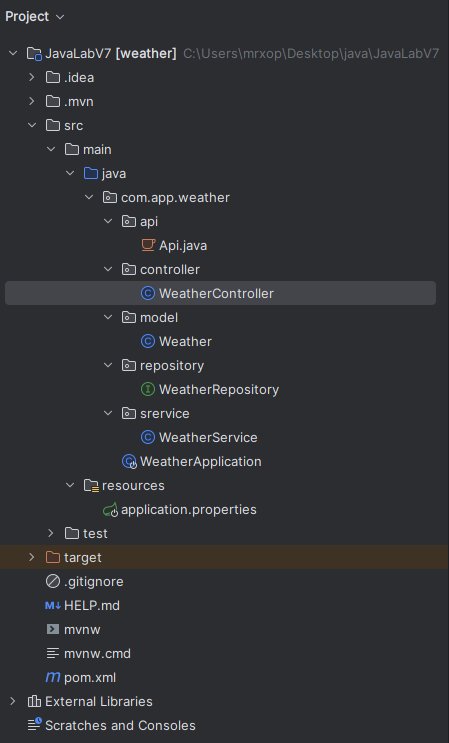


Рисунок 2.1 – Структура проекта

**3. Листинг кода**

Файл WeatherController.java реализующий обработку входящих запросов

package com.app.weather.controller;

import com.app.weather.dto.WeatherDTO;

import com.app.weather.model.Weather;

import com.app.weather.service.WeatherService;

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

import java.util.List;

@RestController

@RequestMapping("/notapi")

public class WeatherController {

private final WeatherService weatherService;

public WeatherController(WeatherService weatherService) {

this.weatherService = weatherService;

}

@GetMapping("/weather")

public WeatherDTO getWeather(@RequestParam(value = "city") String city) {

Weather weather = weatherService.getWeather(city);

return convertToDTO(weather);

}

@PostMapping("/weather")

public WeatherDTO addWeather(@RequestBody WeatherDTO weatherDTO) {

Weather weather = convertToEntity(weatherDTO);

Weather createdWeather = weatherService.createWeather(weather);

return convertToDTO(createdWeather);

}

@GetMapping("/weather/all")

public List<WeatherDTO> getAllWeather() {

List<Weather> allWeather = weatherService.getAllWeather();

return allWeather.stream()

.map(this::convertToDTO)

.toList();

}

private WeatherDTO convertToDTO(Weather weather) {

return new WeatherDTO(weather.getId(), weather.getCity(), weather.getWeatherData());

}

private Weather convertToEntity(WeatherDTO weatherDTO) {

Weather weather = new Weather();

weather.setId(weatherDTO.getId());

weather.setCity(weatherDTO.getCity());

weather.setWeatherData(weatherDTO.getWeatherData());

return weather;

}

}

Файл Weather.java реализующий класс Weather – модель погоды

package com.app.weather.model;

import jakarta.persistence.\*;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import java.sql.Timestamp;

@Entity

@Table(name="weather")

@Getter

@Setter

@AllArgsConstructor

@NoArgsConstructor

public class Weather {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name="date")

private Timestamp date;

@Column(name="city")

private String city;

@Column(name="weather\_data")

private String weatherData;

}

Файл WeatherRepository.java – представляет методы для выполнения запросов к данных

package com.app.weather.repository;

import com.app.weather.model.Weather;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface WeatherRepository extends JpaRepository<Weather, Long> {

Weather findByCity(String city);

Weather findById(long id);

}

}

Файл WeatherRepository.java - реализующий интерфейс для операций хранения, извлечения, обновления, удаления и поиска объектов находящихся в базе данных

package com.app.weather.service;

import com.app.weather.model.Weather;

import com.app.weather.repository.WeatherRepository;

import org.springframework.stereotype.Service;

import java.sql.Timestamp;

import java.time.LocalDateTime;

import java.util.List;

@Service

public class WeatherService {

private final WeatherRepository weatherRepository;

public WeatherService(WeatherRepository weatherRepository) {

this.weatherRepository = weatherRepository;

}

public List<Weather> getAllWeather() {

return weatherRepository.findAll();

}

public void deleteWeatherById(long id) {

weatherRepository.deleteById(id);

}

public Weather createWeather(Weather weather) {

String city = weather.getCity();

Weather existingWeather = weatherRepository.findByCity(city);

if (existingWeather != null) {

// Если запись с таким городом уже существует, обновляем ее

existingWeather.setWeatherData(weather.getWeatherData());

existingWeather.setDate(Timestamp.valueOf(LocalDateTime.now()));

return weatherRepository.save(existingWeather);

} else {

// Если записи с таким городом нет, создаем новую

weather.setDate(Timestamp.valueOf(LocalDateTime.now()));

return weatherRepository.save(weather);

}

}

public Weather getWeather(String city) {

return weatherRepository.findByCity(city);

}

}

Файл WeatherApplication.java реализующий класс main – точка входа в программу

package com.app.weather;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@SpringBootApplication

@EnableJpaAuditing

public class WeatherApplication {

public static void main(String[] args) {

SpringApplication.run(WeatherApplication.class, args);

}

}

Файл application.properties содержащий реализацию подключения базы данных к проекту

spring.datasource.url=jdbc:postgresql://localhost:5432/weather\_db

spring.datasource.username=postgres

spring.datasource.password=1102

spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.hibernate.ddl-auto=update

**4. Результат программы**

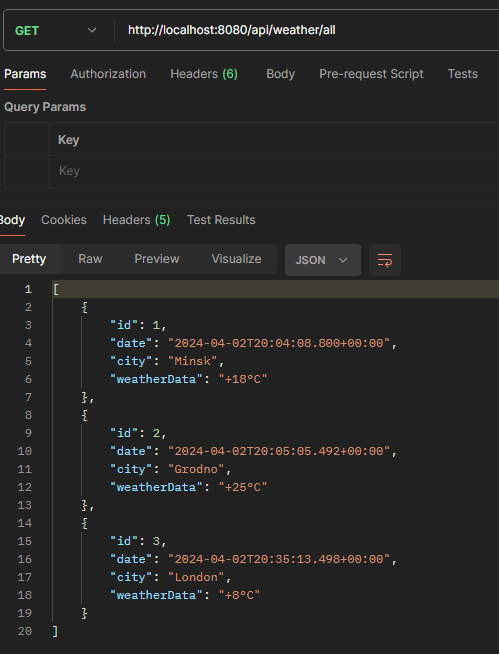


Рисунок 4.1 – Пример получения всех городов через url-запрос

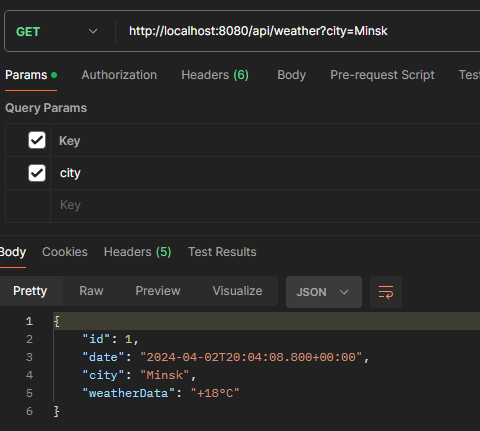


Рисунок 4.2 – Пример получения погоды по заданному городу через url-запрос

**5. Заключение**

В ходе лабораторной работы удалось реализовать простой Rest-сервис с использованием фреймворка Spring. С помощью url-запроса пользователь может получить различную информацию.

В качестве среды разработки использовалась IntelIJ IDEA, для работы с базами данных PostgreSQL16 и pgAdmin 4.

Для запуска и тестирования сервиса использовалось приложение Postman.