1. **Uruchomienie MySQL w Dockerze**

W terminalu wykonaj polecenie :

```
docker run --name weather-db \
-e MYSQL_ROOT_PASSWORD=root \
-e MYSQL_DATABASE=weather \
-e MYSQL_USER=weatheruser \
-e MYSQL_PASSWORD=p@ssw0rd \
-p 3306:3306 \
-v weather-db-data:/var/lib/mysql \
--restart unless-stopped \
-d mysql:5.7
```

Po uruchomieniu kontenera połącz się z bazą danych :

```
docker exec -it weather-db mysql -u root -p
```

- Po zalogowaniu jako root, sprawdź czy baza i użytkownik zostały poprawnie skonfigurowane:

W tym celu w terminalu wykonaj polecenia :

```
SHOW DATABASES;
USE weather;
SELECT user, host FROM mysql.user;
```

- Testowanie połączenia spoza kontenera:

Zainstaluj mysql-client :

```
sudo apt install mysql-client
```

Następnie wykonaj polecenia:

```
mysql -h 127.0.0.1 -P 3306 -u weatheruser -p p@ssw0rd
USE weather;
```

- Konfiguracja bazy danych w mikroserwisie

Edytuj plik *application.properties* :

```
spring.datasource.url=jdbc:mysql://weather-db:3306/weather
spring.datasource.username=weatheruser
spring.datasource.password=p@ssw0rd
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialec
t
```

2. **Rozszerzenie mikroserwisu**

- Przekształcenie *WeatherData* w Encję

```
package com.pbs.edu.WeatherDataService.model;
@Data
@NoArgsConstructor
@Entity
public class WeatherData {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;


…

}
```

- Utworzenie rezpozytorium *WeatherDataRepository*:

```
package com.pbs.edu.WeatherDataService.repository;

import com.pbs.edu.WeatherDataService.model.WeatherData;
import org.springframework.data.jpa.repository.JpaRepository;

public interface WeatherDataRepository extends JpaRepository<WeatherData,
Long> {

}
```

3. **Modyfikacja *WeatherService*.**

```
@Service
public class WeatherService {

    @Autowired
    private WeatherDataRepository weatherDataRepository;

    @Value("${weather.api.url}")
    private String apiUrl;

    @Value("${weather.api.key}")
    private String apiKey;

    private final RestTemplate restTemplate = new RestTemplate();

    public WeatherData getWeatherData(String city, String countryCode) {
        String url = String.format("%s?q=%s,%s&APPID=%s&units=metric",
apiUrl, city, countryCode, apiKey);
        WeatherData weatherData = restTemplate.getForObject(url,
WeatherData.class);

        if(weatherData != null) {
            weatherData.setCityName(city);
            weatherData.setCountry(countryCode);
```

```
            weatherDataRepository.save(weatherData);
        }

        return weatherData;
    }

    public Iterable<WeatherData> getWeatherHistory() {
        return weatherDataRepository.findAll();
    }
}
```

4. **Dodanie nowego endpointu w *WeatherController***

```
@RestController
public class WeatherController {
    @Autowired
    private WeatherService weatherService;

    @GetMapping(value = "/weather", produces = "application/json")
    public WeatherData getWeather(@RequestParam String city,
@RequestParam String countryCode) {
        return weatherService.getWeatherData(city, countryCode);
    }

    @GetMapping("/weather/history")
    public Iterable<WeatherData> getWeatherHistory() {
        return weatherService.getWeatherHistory();
    }
}
```

5. **Uruchom oraz przetestuj lokalnie aplikacje.**

- Uruchom aplikację Spring Boot np. używając komendy:

```
mvn spring-boot:run
```

6. **Zmodyfikuj plik Dockerfile oraz utwórz plik docker-compose.yml, a następnie uruchom.**

- W folderze głównym projektu otwórz plik *Dockerfile*:

```
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY target/WeatherDataService-0.0.1-SNAPSHOT.jar app.jar
COPY dockerize /usr/local/bin/dockerize
RUN chmod +x /usr/local/bin/dockerize
EXPOSE 8080
# Komenda startowa: czekaj na dostępność bazy danych na porcie 3306 przed
uruchomieniem aplikacji
ENTRYPOINT ["dockerize", "-wait", "tcp://weather-db:3306", "-timeout",
"60s", "java", "-jar", "app.jar"]
```

Link do dockerize: https://github.com/jwilder/dockerize/releases/download/v0.8.0/dockerize-linux-amd64-v0.8.0.tar.gz

- W folderze głównym projektu utwórz plik *docker-compose.yml* o następującej zawartości:

```yaml
services:
  weather-db:
    image: mysql:5.7
    container_name: weather-db
    restart: unless-stopped
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: weather
      MYSQL_USER: weatheruser
      MYSQL_PASSWORD: p@ssw0rd
    ports:
      - "3306:3306"
    volumes:
      - weather-db-data:/var/lib/mysql

  weather-service:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: weather-service
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://weather-db:3306/weather
      SPRING_DATASOURCE_USERNAME: weatheruser
      SPRING_DATASOURCE_PASSWORD: p@ssw0rd
      SPRING_JPA_HIBERNATE_DDL_AUTO: update
    ports:
      - "8080:8080"
    depends_on:
      - weather-db

volumes:
  weather-db-data:
```

- Uruchomienie:

```
docker-compose up --build
```

7. **Przetestuj działanie mikroserwisu w kontenerze uruchamiając przeglądarkę i wpisując adresy** http://localhost:8080/weather?city=Warsaw&countryCode=PL http://localhost:8080/weather?city=London&countryCode=UK

8. **Przetestuj działanie endopint history wpisując w przeglądarce adres** http://localhost:8080/weather/history