1. **Utworzenie projektu mikroserwisu UserService w Spring Boot za pomocą Spring Initializr**
   - Ustaw parametry projektu:
     1) **Project**: *Maven Project*
     2) **Language**: *Java*
     3) **Spring Boot Version**: *3.3.6*
     4) **Group**: *com.pbs.edu*
     5) **Artifact**: *UserService*
     6) **Packaging**: *Jar*
     7) **Java**: *17*
   - Dodanie zależności:
     1) Spring Web
     2) Spring Data JPA
     3) MySQL Driver
     4) Lombok
     5) Spring Security
     6) Spring Gateway
   - Pobierz projekt i rozpakuj plik ZIP, a następnie otwórz go w wybranym IDE (np. IntelliJ IDEA).

2. **Konfiguracja mikroserwisu**
   - Przejdź do folderu *src/main/resources* i otwórz plik *application.properties*.
   - Dodaj konfiguracje bazy danych oraz spring.security:

```
spring.application.name=UserService
spring.security.user.name=admin
spring.security.user.password=password
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
```

3. **Implementacja UserService**
   - Utworzenie encji *User*:

```
package com.pbs.edu.userservice.model;

import jakarta.persistence.*;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;
```

```java
import lombok.Data;
import lombok.NoArgsConstructor;

@Entity
@Data
public class User {

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

    @Email
    @NotBlank
    private String email;

    @NotBlank
    private String password;

    private boolean rainNotification;
    private boolean snowNotification;
    private boolean windNotification;
    private boolean highTempNotification;
    private boolean lowTempNotification;

    @NotBlank
    private String country;

    @NotBlank
    private String city;
}
```

- Utworzenie rezpozytorium *UserRepository*:

```java
package com.pbs.edu.userservice.repository;

import com.pbs.edu.userservice.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

}
```

4. Utworzenie *UserService*.

```java
package com.pbs.edu.userservice.service;

import com.pbs.edu.userservice.model.User;
import com.pbs.edu.userservice.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserService {

    @Autowired
```

```
    private UserRepository userRepository;

    private final BCryptPasswordEncoder passwordEncoder = new
BCryptPasswordEncoder();

    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    public User addUser(User user) {
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        return userRepository.save(user);
    }
}
```

5. **Utworzenie *UserController***

```
package com.pbs.edu.userservice.controller;

import com.pbs.edu.userservice.model.User;
import com.pbs.edu.userservice.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/users")
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping
    public List<User> getAllUsers() {
        return userService.getAllUsers();
    }

    @PostMapping
    public User addUser(@RequestBody User user) {
        return userService.addUser(user);
    }
}
```

6. **Utwórz plik Dockerfile oraz zmodyfikuj plik docker-compose.yml, a następnie uruchom.**

- W folderze głównym projektu utwórz plik *Dockerfile*:

```
FROM maven:3.9.9-eclipse-temurin-17 AS builder
WORKDIR /app
COPY pom.xml .
COPY src ./src
RUN mvn clean package -DskipTests
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY --from=builder /app/target/*.jar app.jar
COPY dockerize /usr/local/bin/dockerize
```

```
RUN chmod +x /usr/local/bin/dockerize
EXPOSE 8082
# 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

- Zmodyfikuj plik *docker-compose.yml*:

```
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: ./WeatherDataService
      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

  notification-service:
    build:
      context: ./NotificationService
      dockerfile: Dockerfile
    container_name: notification-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
      SPRING_MAIL_HOST: smtp.mailtrap.io
      SPRING_MAIL_PORT: 587
      SPRING_MAIL_USERNAME: 31aafe68f86106
      SPRING_MAIL_PASSWORD: 230bdd9bdf3ef2
    ports:
      - "8081:8080"
    depends_on:
      - weather-db
```

```
  user-service:
    build:
      context: ./UserService
      dockerfile: Dockerfile
    container_name: user-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:
      - "8082:8082"
    depends_on:
      - weather-db

volumes:
  weather-db-data:
```

- Uruchomienie:

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

7. Uruchom aplikacje Postman

Utwórz nową kolekcję o nazwie *Weather User Test* oraz dodaj zapytanie *Add User* typu POST. Adres to: http://localhost:8082/users. W zakładce *params* dodaj nowy klucz *Content-Type* o wartości *application/json*.

Następnie w zakładce body wybierz raw i ustaw na JSON:

```
{
  "email": "pawplo003@pbs.edu.pl",
  "password": "secret2@",
  "rainNotification": true,
  "snowNotification": false,
  "windNotification": true,
  "highTempNotification": true,
  "lowTempNotification": false,
  "country": "Poland",
  "city": "Bydgoszcz"
}
```

W zakładce Authorization wybierz Auth Type: Basic Auth oraz wpisz Username oraz Password z pliku application.properties.

8. **Kliknij Send i zweryfikuj poprawność zapytania w aplikacji Postman.**

9. **Uruchom terminal i zweryfikuj poprawność danych**

```
docker exec -it weather-db mysql -u weatheruser -p
USE weather;
SHOW TABLES;
SELECT * FROM user;
exit
```