Backend Phase

The `Web Programming` course project

Sharif University of Technology - Spring 2024

- **Yasamin Golzar**
  - **Student ID**: 98171064
  - **Email**: yasamingolzar.ce@gmail.com
- **Mehrad Milanloo**
  - **Student ID**: 99105775
  - **Email**: milanloomehrad@gmail.com

What is `National Countries` project?

This project is a simple web application that provides information about countries. The information includes the country's name, capital, population, weather, etc. The project is implemented using the `Spring Boot` framework.

The project has the following features:

1. **List of Countries**: The project provides a list of countries with their names and capitals.
2. **Country Information**: The project provides detailed information about a country. The information includes the country's name, capital, population, weather, etc.
3. **Weather Information**: The project provides weather information for a country. The information includes the temperature, humidity, wind speed, etc.
4. **User Management**: The project provides user management features. The users can register and login. The project uses `JWToken` for authentication. **Admin** users can view the list of users and `activate/deactivate` them.
5. **Caching**: The project uses `Redis` for caching. The weather information is cached for a certain period of time.
6. **Messaging**: The project uses `RabbitMQ` for messaging. The weather service sends a message to the user service when the weather information is updated.

7. **Pagination**: The project provides pagination in every url that returns a list of items. The *page size* and *number of pages* can be specified in the request.

Technologies

The project uses the following technologies:

1. **Spring Boot**: The project is implemented using the `Spring Boot` framework.
2. **JWToken**: The project uses `JWToken` for authentication.
3. **MySQL**: The project uses `MySQL` as the database.
4. **REST API**: The project provides a `REST API` for accessing the information.
5. **Redis**: The project uses `Redis` for caching.
6. **RabbitMQ**: The project uses `RabbitMQ` for messaging. (In the weather service)

How to Run

To run the project, follow these steps:

1. **Maven Build the Project**: If you have installed Maven on your machine then use the below command:

```
mvn clean package
```

Note: Go to the root directory of the project and execute the above command.

2. **Initial `Redis`**: Run the following command to start the `Redis` server:

```
docker run -d -p 6379:6379 redis
```

3. **Initial `RabbitMQ`**: Run the following command to start the `RabbitMQ` server:

```
docker run -d -p 5672:5672 -p 15672:15672 rabbitmq:3-management
```

4. **Run Spring Boot Project**: Use the following command to run the Spring Boot application:

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

Once you run the Spring Boot application, `Hibernate` will create the database tables automatically.

API Documentation

The project provides the following `REST API` endpoints:

Auth APIs

1. **Register User**: The endpoint allows users to register.
   - **URL**: `/users/register`
   - **Method**: `POST`
   - **Request Parameters**:
     - `username`: The username of the user.
     - `password`: The password of the user.
   - **Response**: A message indicating the success or failure of the registration.
2. **Login User**: The endpoint allows users to login.
   - **URL**: `/users/login`
   - **Method**: `POST`
   - **Request Parameters**:
     - `username`: The username of the user.
     - `password`: The password of the user.
   - **Response**: A `JWToken` for the user.

Country APIs

1. **List of Countries**: The endpoint provides a list of countries with their names and capitals.
   - **URL**: `/countries`
   - **Method**: `GET`
   - **Header Parameters**:
     - `Authorization: API {token}`: The token of the user.
   - **Request Parameters**: None (Optional: `page` and `size` for pagination)
   - **Response**: A list of countries with their names and capitals.
2. **Country Information**: The endpoint provides detailed information about a country. The information includes the country's name, capital, population, weather, etc.

- **URL**: `/countries/{countryName}`
- **Method**: `GET`
- **Header Parameters**:
  - `Authorization: API {token}`: The token of the user.
- **Request Parameters**:
  - `countryName`: The name of the country.
- **Response**: Detailed information about the country.

3. **Weather Information**: The endpoint provides weather information for a country. The information includes the temperature, humidity, wind speed, etc.
   - **URL**: `/{countryName}/weather`
   - **Method**: `GET`
   - **Header Parameters**:
     - `Authorization: API {token}`: The token of the user.
   - **Request Parameters**:
     - `countryName`: The name of the country.
   - **Response**: Weather information for the country.

User (Admin) APIs

1. **Get All User**: The endpoint lists all the users for the admin.
   - **URL**: `/users`
   - **Method**: `GET`
   - **Header Parameters**:
     - `Authorization: API {token}`: The token of the user.
   - **Request Parameters**: None (Optional: `page` and `size` for pagination)
   - **Response**: A list of users.
2. **Activate/Deactivate User**: The endpoint allows the admin user to activate/deactivate a user.
   - **URL**: `/users?username={username}&active={active/deactive(true/false)}`
   - **Method**: `PUT`
   - **Header Parameters**:

  - `Authorization: API {token}`: The token of the user.
- **Request Parameters**:
  - `username`: The username of the user.
  - `active`: The status of the user.
- **Response**: A message indicating the success or failure of the operation.

Token APIs

1. **Create Token**: The endpoint allows users to create a token.
   - **URL**: `/user/api-tokens`
   - **Method**: `POST`
   - **Header Parameters**:
     - `Authorization: API {token}`: The token of the user.
   - **Request Parameters**:
     - `name`: The name of the token.
     - `expire_date`: The expiration date of the token.
   - **Response**:
     - `name`: The name of the token.
     - `expirationDate`: The expiration date of the token.
     - `token`: The token.
2. **Get All Tokens**: The endpoint lists all the tokens for the user.
   - **URL**: `/user/api-tokens`
   - **Method**: `GET`
   - **Header Parameters**:
     - `Authorization: API {token}`: The token of the user.
   - **Request Parameters**: None (Optional: `page` and `size` for pagination)
   - **Response**: A list of tokens. (Token values are not shown)
3. **Delete Token**: The endpoint allows users to delete the sent token.
   - **URL**: `/user/api-tokens`
   - **Method**: `DELETE`
   - **Header Parameters**:

    - `Authorization: API {token}`: The token of the user.
- **Request Parameters**: None
- **Response**: A message indicating the success or failure of the operation.

Meow API

1. **Meow**: The endpoint returns a `meow` sound.
    - **URL**: `/users/meow`
    - **Method**: `GET`
    - **Header Parameters**:
        - `Authorization: API {token}`: The token of the user.
    - **Request Parameters**: None
    - **Response**: A `meow` sound.
