**User Service**

**My Implementation**:

Implemented the rest service using Spring Boot.

Written few Unit and Integration tests.

Using TestContainer to run Integration tests out of the box in seperate container with exact production like database (postgreSQL). I’m using TestContainer so that we don’t need to have the postgreSQL database in our local system and we can run tests successfully without effecting to production database.

**Tools & Technologies Used**

Java 8

Spring Boot

Spring AOP

Spring Data JPA

Hibernate

PostgreSQL

Swagger (Used for REST API documentation)

TestContainer (Used to run integration tests in a container )

Junit, Mockito (Used for unit tests)

RestAssured (Used for Integration tests)

Docker

**Source code:**

Source code is available inside : /perseus-assignment/user-service/src

**Steps to run the application:**

You can run the application either by docker compose or maven. Below are the steps for both the ways:

**By using docker compose:**

I have created images for user-service and database and pushed the images into Docker Hub. Linking for these images is present in **docker-compose.yml** file. You just have to run the docker compose.

1. Go to directory: */perseus-assignment/docker-compose*

2. Execute the command:  ***docker-compose up***

Application will start on port **9087 (**[**http://localhost:9087/**](http://localhost:9082/tourbookingapplication/)**userservice**[**/**](http://localhost:9082/tourbookingapplication/)**)**. If you want to configure port, you can change it on docker-compose.yml file

**Note:** I have shared the schema.sql file inside*docker-compose* directory to create database and related schema.If database not get created using the above command (***docker-compose up***) and application throw exception to connect with database then you can either prune the docker volumes or include its volume to not skip the sql script and execute it again.

Below are the helpful command for this:

docker volume ls

docker volume rm **volumeName**

**OR**

docker volume prune

**By using maven:**

You need to have the postgreSQL database to run application using maven:

1. Install postgreSQL DB

2. Execute schema.sql script present inside directory: */perseus-assignment/docker-compose*

*3. Change below datasource properties in application.properties file (/perseus-assignment/user-service/src/main/resources)*

*spring.datasource.url=*jdbc:postgresql://database/userservice

spring.datasource.username=postgres

*spring.datasource.password=postgres*

4. Execute maven command : mvn clean package

5. Run command: java -jar ./target/user-service.jar

**API Documentation**

I have integrated the swagger for detailed API documentation. You can check with these URLs:

[http://localhost:9087/](http://localhost:9087/flaregameswebapplication/swagger-ui.html" \l "/)userservice[/swagger-ui.html#/](http://localhost:9087/flaregameswebapplication/swagger-ui.html" \l "/)

[http://localhost:9087/userservice/v2/api-docs](http://localhost:9087/flaregameswebapplication/v2/api-docs)

I’m also providing short API documentation here for your reference.

**1. GET -** Get a User by user id

Request URL:

[http://localhost:9087/userservice/users/](http://localhost:9087/userservice/users/12){userId}

**2. GET -** Get all users by name

Request URL:

<http://localhost:9087/userservice/users?name>={name}

**3. POST -** Create a user with contact data (Email/Phone)

Request URL:

<http://localhost:9087/userservice/users>

**Request Body Example:**

{

"firstName":"Santosh",

"lastName":"Pandey",

"emails": [

{

"mail":"sp@gmail.com"

},

{

"mail":"sp8@gmail.com"

}

],

"phoneNumbers":[

{

"number":"919878285664646"

},

{

"number":"919877777778282"

}

]

}

**4. DELETE -** Delete a user by id

Request URL:

[http://localhost:9087/userservice/users/](http://localhost:9087/userservice/users/12){userId}

**5. POST -** Add additional contact data for a user

Request URL:

<http://localhost:9087/userservice/users/>[{userId}](http://localhost:9087/userservice/users/6/contactdata)/contactdata

**Request Body Example:**

{

"emails": [

{

"mail":"sp3333@gmail.com"

},

{

"mail":"sp8@gmail.com"

}

],

"phoneNumbers":[

{

"number":"987828588888"

},

{

"number":"9878282999999999"

}

]

}

**6. PUT -** Update contact data for an existing user

Request URL:

[http://localhost:9087/userservice/users/{userId}/contactdata](http://localhost:9087/userservice/users/6/contactdata)

**Request Body Example:**

{

"emails": [

{

"id":15,

"mail":"sp3333@gmail.com"

},

{

"id":11,

"mail":"sp80@gmail.com"

}

],

"phoneNumbers":[

{

"id":11,

"number":"987828588888"

}

]

}