Checkpoint 2

Meeting Room Reservation System

Document version: 2.2.0

Deadline:15/01/2023

Name of Project: Meeting Room Reservation System

Link to the project page: B221_B6B36EAR/let's grow

Solver: Dmytro Rastvorov

Exercise date: WED – 6:15 p.m

Practitioner Name: Miroslav Holeček

Content

- Topic
- A short description of the expected functionality
- Types of users
- Functions for individual types of users
- System limitations
- Object model (UML class diagram)
- Database
- API
- Description of the application and its structure
- Project usage notes
- Docker Compose
- Installing the application
- Links
- Feedback

- **Topic:**

  Meeting Room Reservation System

  - A system that supports the management of meeting rooms and their reservation.

- **A short description of the expected functionality:**

  1. List of main features of the application:

  - The management system consists of a user and administrative part:
    - **User**
    - **Administrator**

  2. Who will the resulting system be intended for:

  - This system is designed for a group of people who plan meetings in conference rooms.

- **Types of users:**

❖ The system is designed for 2 types of users: User and Administrator.

➢ **User - is**the user who interacts with the conference room reservation application.

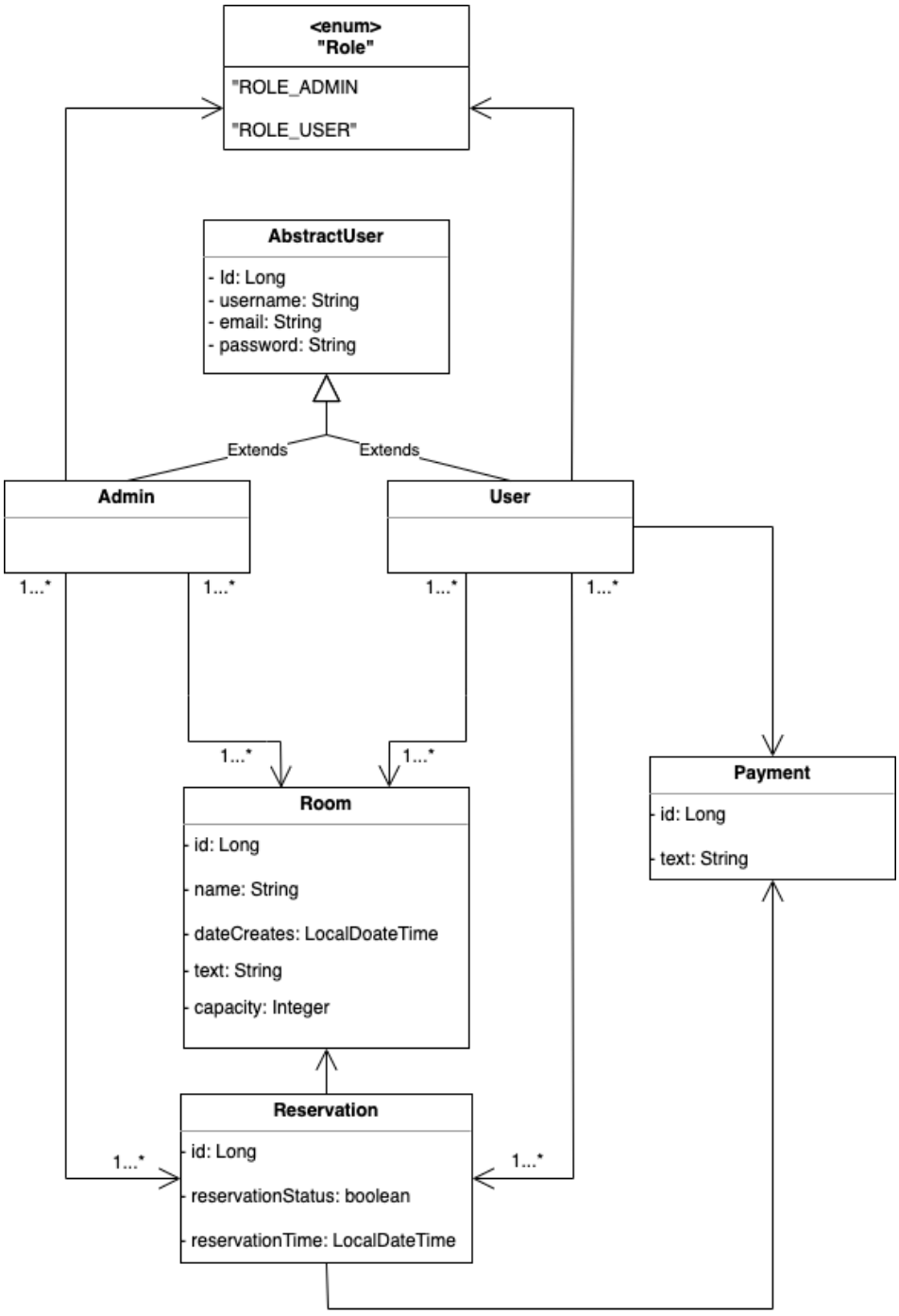
➢ **Administrator – is**a user who manages conference room reservations.

- **What functions will it perform?**
  - ❖ The system includes the following features for users:
    - ➢ **User -** has option to reserve a room, cancel the reservation, see the status of the room and the time it will be reserved.
    - ➢ **Administrator –** can add a room, remove a room, change room properties, and has the same options as a User.

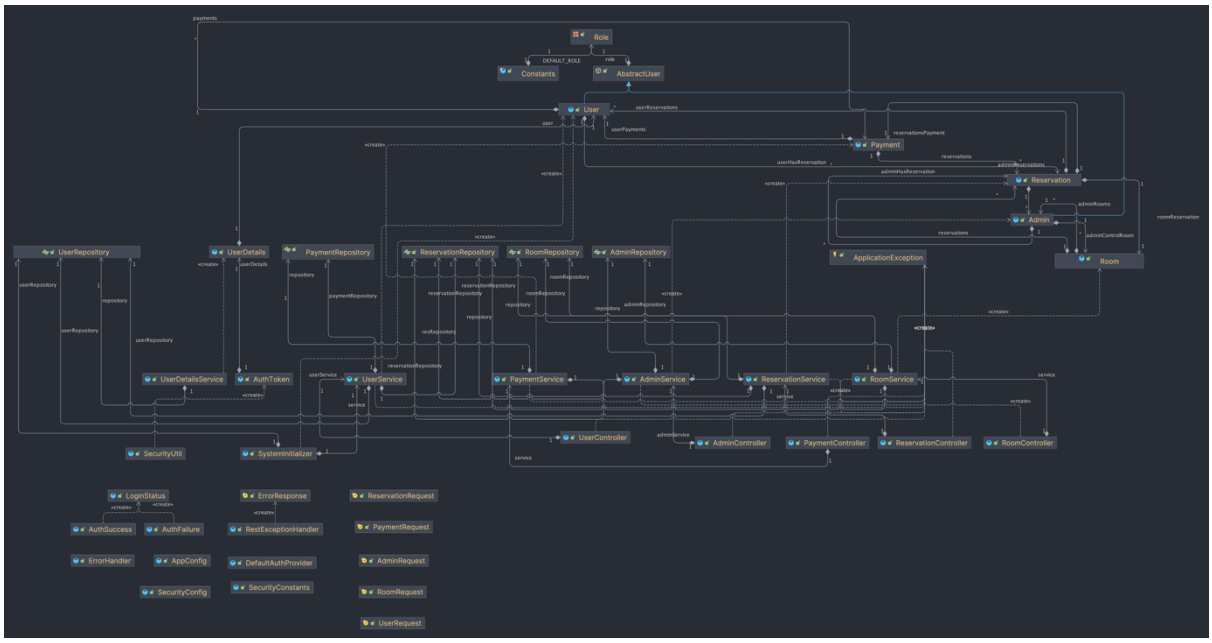
- **System limitations**

  - ❖ **The system has limitations for users.**
    - ➢ The user cannot:
      - ■ Book 1 room 2 times (as well as the administrator)
      - ■ Create and delete a room.
      - ■ Edit room properties.
      - ■ Show users who have booked rooms other than themselves.
  - ❖ **We also have limitations for the application:**
    - ■ We have a time limit according to real time. That is, we cannot make a reservation for next year or month, but we can only at this time in this month.
    - ■ We cannot enter incorrect data. We will get an error.

- **Object model (UML class diagram)**
  - UML diagram (created with draw.io)
    (Demonstrates binding between entities)



- UML diagram created using **IntelliJ IDEA**
  (Demonstrates full coupling between classes)



- ## Database

  The project uses a PostgreSQL database.

- ## API

  The project uses the REST API using Postman.
  In the documentation it is possible to find all the APIs that cover the logic of the project.

  REST API documentation (Postman) can be found here:
  https://documenter.getpostman.com/view/22903223/2s8ZDU4Nzj

- ## Description of the application and its structure

This application allows you to make a room reservation where the user can create a reservation, view its hourly price, and find out.
information about him.

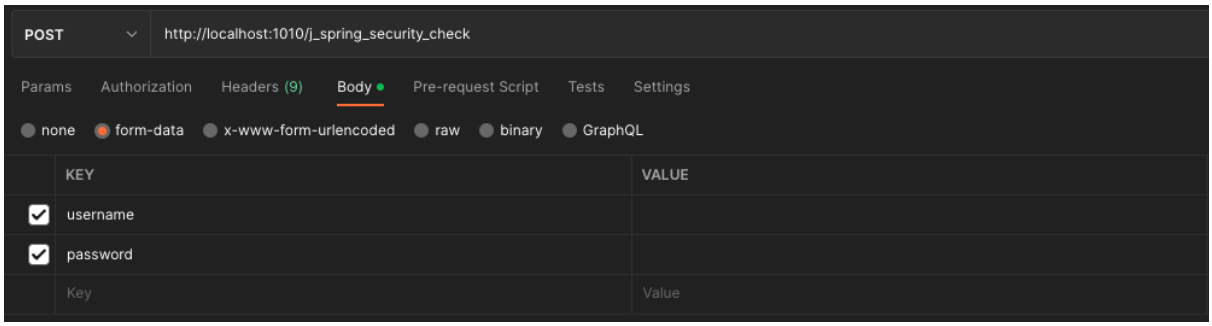
On the other hand, an admin has many more options than a user.

In addition to what the user can do, the admin has full freedom when creating rooms, reservations, users, and other admins and can also change them and
lubricate.

- ## Project usage notes

The project uses Security support so that the user can use the program, you must create your own profile.

When running in the Postman application in the folder **REST API/Authorization/POST** you can confirm the login and below you will see the output (see image below) which says that the login was successful.



- ## Docker Compose (Connecting an application to a database container)

To connect an application to a database container using Docker Compose, you need to ensure that the application configuration is set up correctly to connect to the database container. For this, we need to do the following steps:

Step 1: Configure the application.

• Ensure that the database connection properties in the **application.yaml** file match the database container configuration defined in the **docker-compose.yml** file, such as name, password, database, and port.

Step 2: Start the application and database containers

• Open a terminal or command prompt and navigate to the root directory of the project.

• Run the following command to start the containers:

```
docker-compose up -d
```

• Use the **-d** switch to start the containers in disconnected mode so they will run in the background.

Step 3: Verify the connection

• Once the containers are running, you can verify the connection between the application and the database container.

• Check the application container logs to make sure there are no connection-related errors:

```
docker-compose logs app
```

• If the connection is successful, you will not see any errors related to the database connection.

That's it! You have successfully connected your application to the database container using Docker Compose. You can now use the app with the database connected.

● Installing the application

!! IMPORTANT: *The project works in conjunction with JDK 18. Make sure your version is up to date before installing and running the project.*

1) Copy the SSH project from the page Git.
2) Open a terminal on your computer and type git clone (SSH link)
3) The file will appear in the folder you entered the path to in the terminal.
4) Open the project file in the IDE. (Recommended IntelliJ IDEA)
5) Then you need to configure the database in the file src/main/resources/application.yaml and if we want to use Docker, we need to configure in the docker-compose.yml file.
6) Open Maven, go to Lifecycle, expand it, and click on package (then when you change something, you have to select clean and then package at the beginning). This will generate the project, go through all the necessary operations and be ready to run.
7) Once this process is complete, navigate to the src/main/java/reservation/room/meeting/sem/ear folder. and select the Application.java file.
8) Then click the run arrow and the project will start compiling.
9) Then we launch Postman, we open the Authorization/POST folder, write the configured credentials and you can use the project.

● Links

Link to the project: https://gitlab.fel.cvut.cz/B221_B6B36EAR/rastvdmy

Link to documentation in Postman:

https://documenter.getpostman.com/view/22903223/2s8ZDU4Nzj

- **Feedback**

  For me, this subject was something new and exciting. I learned how Java Spring Boot works, what it is used for and what features it has.

  While writing a semester project, I was able to learn about writing a Spring Boot application, connecting to a database, using Docker and Postman.

  The semester project also taught me to work independently and solve errors that I found difficult.
  I would like to express my special thanks to the lecturers and to the trainer Miroslav Holeček for a pleasant experience in this subject and contribution to our knowledge!

Thank you for your attention!