Distributed Systems

Assignment 1:
Request-Reply Communication

Alin Tcaci
Group: 30441

1.Overview

The project requirement is to develop an Energy Management System that consists of a frontend and two microservices design for managing the users and their associated devices.

2.Microservices

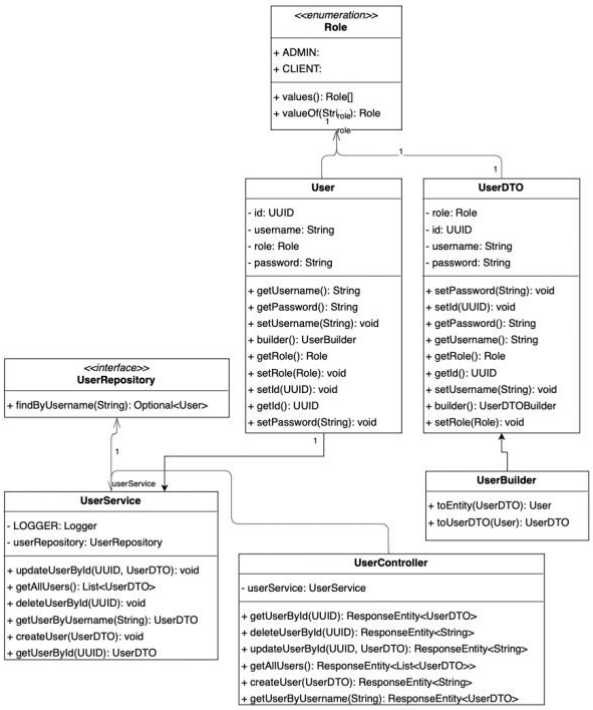
For each service, we need to include a REST API controller that can handle basic CRUD operations for specific entities, while also making sure data stays consistent. These services are built with the Java Spring Boot framework and are structured using a layered approach, splitting everything into distinct layers like entity, service, repository, and controller.

2.1.User Microservice

The user controller handles the user management, where each operation works independently and interacts directly with the user database. The corresponding endpoints are:

- /getAllUsers - Fetches all users from the database.
- /getUserById/{id} - Retrieves a specific user by their unique ID.
- /createUser - Adds a new user to the database.
- /updateUserById/{id} - Updates an existing user's details based on their ID.
- /deleteUserById/{id} - Deletes a user from the database by their ID.
- /getUserByUsername/{username} - Finds a user based on their username.

Each endpoint operates individually, meaning it directly performs actions on the user database without depending on other endpoints.

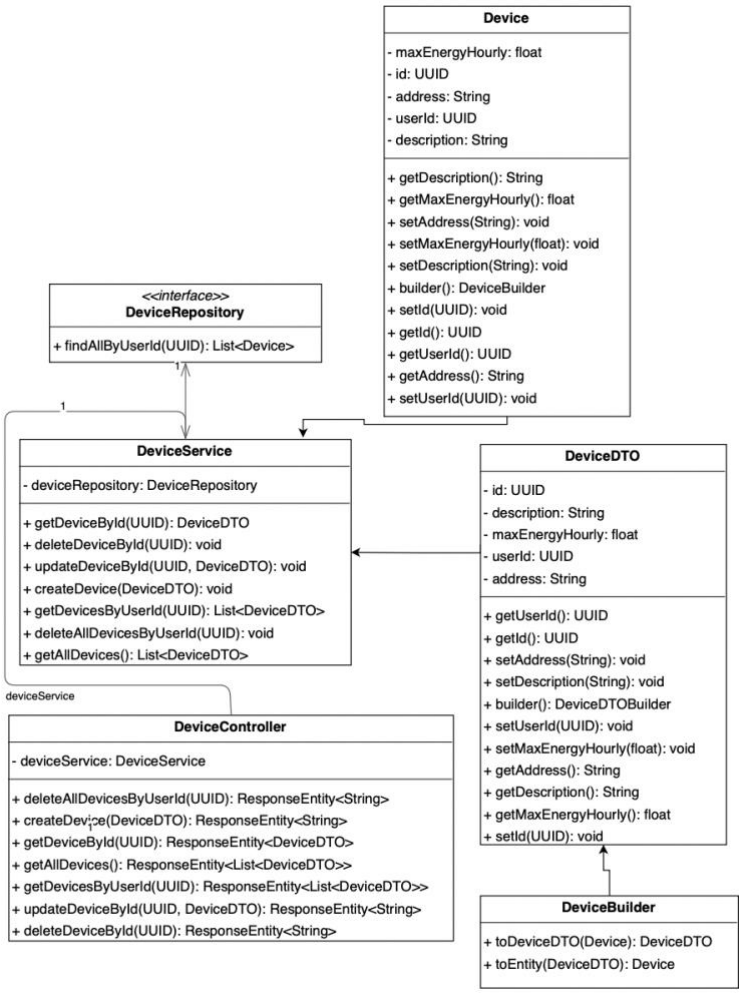


2.2.Device Microservice

The device controller handles managing of devices. Each device record includes a userId field, which acts like a foreign key, linking devices to specific users. However, even though the userId allows device data to be associated with users, the user and device services are set up as separate microservices, each with its own database. The association between the user and device will be done in frontend.

The corresponding endpoints are:

- /getAllDevices - Retrieves all devices in the device database.
- /getDeviceById/{id} - Fetches a specific device by its unique ID.
- /getAllDevicesByUserId/{userId} - Retrieves all devices linked to a specific user by userId.
- /createDevice - Adds a new device to the device database.
- /updateDeviceById/{id} - Updates details for an existing device based on its ID.
- /deleteDeviceById/{id} - Removes a specific device by its ID.
- /deleteAllDevicesByUserId/{userId} - Deletes all devices associated with a given userId.



3.Frontend

The frontend application is made by using the Angular framework. It has a login page, from where the user will be send to its corresponding Role page. The client role page consists in viewing a list of devices that the user owns. The admin page allows the user to perform CRUD operations on both the users and devices. To communicate with the backend, it uses the bult-in httpService from Angular.

4.Doker

Each database, microservice and frontend have a separate container. The setup ensures isolated development for each microservice and consistent deployment across environments.