SOLUTION DESCRIPTION

Energy Management System

Assignment 1

Distributed Systems 2023

Student: **Radu-Augustin Vele**

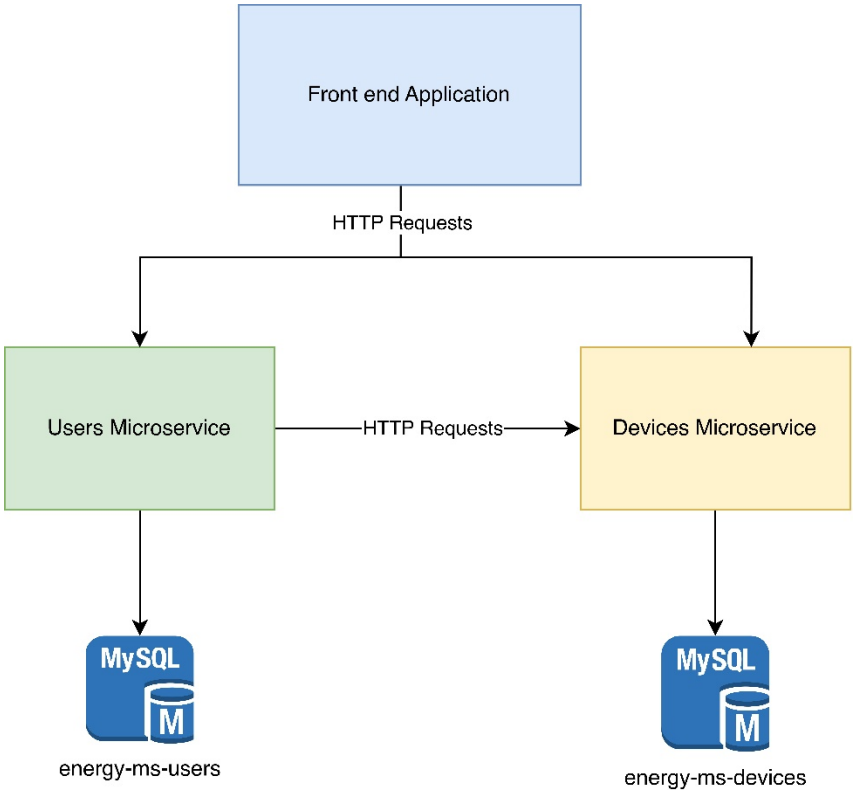
Group: **30442**

Table of Contents

1. Project objectives

This project's primary objective is to develop an energy management system that serves the needs of both end-users and administrators. In the current project phase, administrators are to be able to perform CRUD operation on users and devices and assign/remove devices to/from users. Also, users can view the information stored about them and the status of the devices they own.

2. Conceptual architecture



The system is follows a microservice architecture and is composed of five main components:

- Frontend application – built using React + Typescript.
- Users microservice – built using Spring Boot.
- Devices microservice – built using Spring Boot.
- Users MySQL DB
- Devices MySQL DB

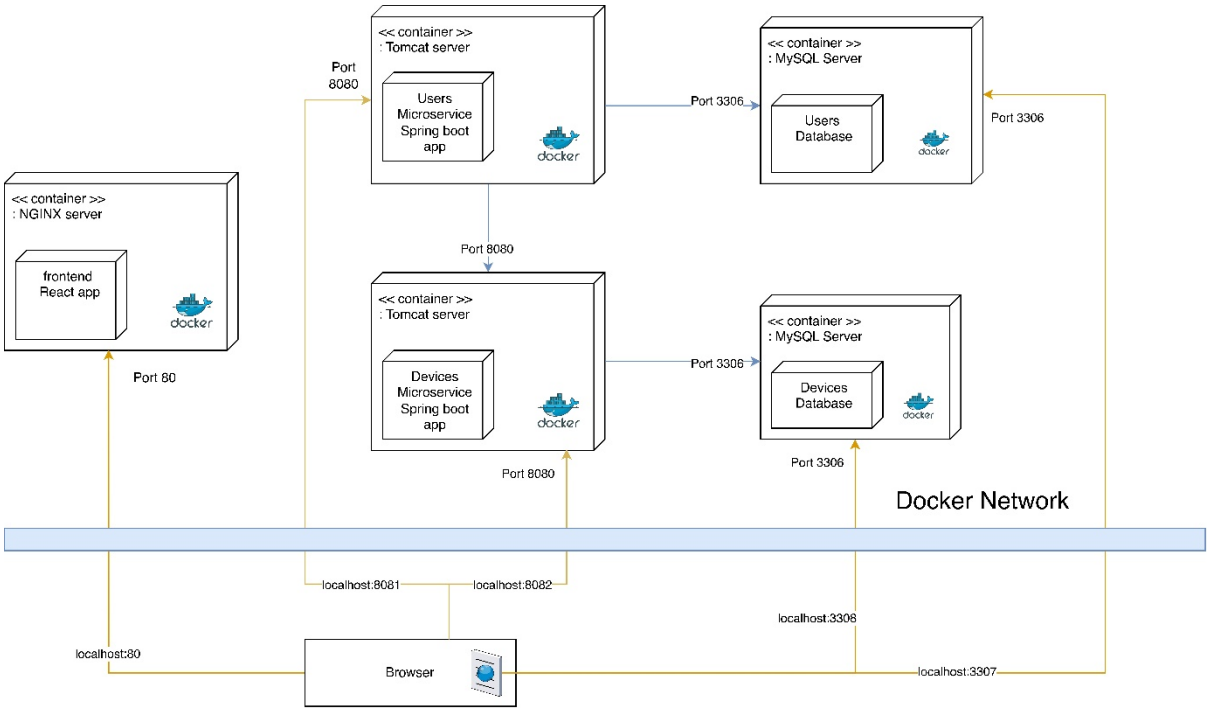
2. 1. The coupling between the microservices

The goal of microservice architecture is to create individual composable components that are coupled to the least extent possible. However, in our specific case, there is a certain level of coupling between the two microservices. This coupling arises from the necessity to maintain and persist the relationship between users and devices, which is a one-to-many relationship.

To mitigate this coupling, we've restricted synchronization requests to be initiated solely by the Users microservice. This approach ensures that the Devices microservice only interacts with requests from the Users microservice for actions such as registering new users with their associated devices and deleting users along with their existing device relationships.

3. UML Deployment Diagram

The system is deployed using Docker containers, which facilitates their inclusion in a network of interconnected Docker containers. At this stage, the Docker containers are executed locally and are made accessible through specific local host ports.



4. Readme – build and execution considerations.

4.1. Getting the project source code

Prerequisites:

- Git

Clone the following resources:

- Frontend repository:
  - *https://gitlab.com/ds2023_30442_vele_radu-augustin/ds2023_30442_vele_radu-augustin_frontend*
- Users Microservice repository:
  - *https://gitlab.com/ds2023_30442_vele_radu-augustin/ds2023_30442_vele_radu-augustin_users_uservice*
- Devices Microservice repository:
  - *https://gitlab.com/ds2023_30442_vele_radu-augustin/ds2023_30442_vele_radu-augustin_devices_uservice*

4.2. Local development

Prerequisites:

- Java
- Node version 18, npm
- MySQL

Recommended tools:

- IntelliJ (spring projects) (the projects are using the maven build tool)
- VSCode (react project) (the projects are using the Vite build tool)

4.3. Deployment

Prerequisites:

- Docker

Note: make sure you are on the docker_prod branch

For the spring projects

- generate the JAR files using the command mvn clean package.
- Run docker build -t <image_name> . in the project directory

For the react project

- Run docker build -t <image_name> . in the project directory

Creating the containers

- Locate the compose.yml file in the Users project directory
- Run docker compose -p <container stack name> up -d
- Check if the containers were created successfully – you may have to manually restart the spring containers (they may be initialized before the DB servers => error)