Assignment 1
Distributed Systems Laboratory

Energy Management System

Voicu Sara-Ioana
Group: 30444

Introduction

The application I have developed is an Energy Management System, comprising two microservices built using Java Spring, and a frontend component developed in React JS. The purpose of an EMS is to improve energy efficiency, reduce energy consumption, and lower energy-related costs.

Backend components

User Microservice

This microservice is responsible for managing user data, including both clients and administrators. This microservice has its own database to store information about users: username, password, role and unique id. Users with the ‘ADMINISTRATOR’ role have the ability to perform CRUD (Create, Read, Update, Delete) operations on user records.

Device Microservice

This microservice focuses on device information, which includes details and an user_id attribute to enable communication between the microservices. In this way, there is a One-To-Many relationship between users and devices: a user can have many devices, but a device can only have one user. Here administrators are able to execute CRUD operations on device records, while clients can retrieve information about their devices.

Communication

RestTemplate is a component provided by the Spring Framework (particularly Spring Web) that facilitates the communication between microservices in this architecture. It has the ability to send and receive HTTP requests, allowing the microservices to interact effectively through RESTful APIs.

Frontend component

The fronted component is straightforward:

- Home page
- Login page
- Client page: the users can see their devices
- Admin page: the administrators can perform operations on users and devices

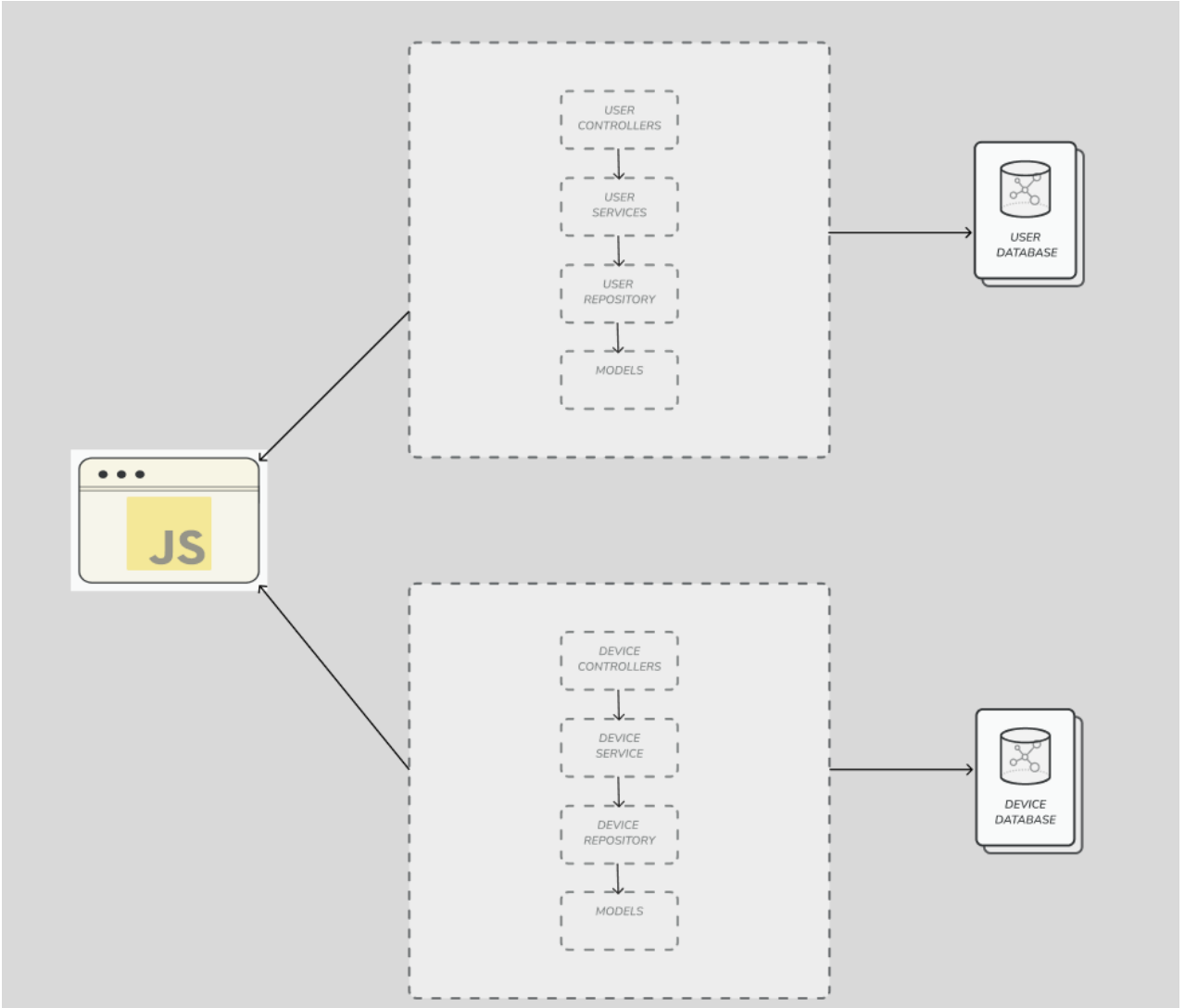
After login, the application redirects the user based on their role, which is extracted from a JWT (JSON Web Token). The JWT is a method used for securing data and authenticating users in web applications, including microservices and API-based systems. Its functionality is currently implemented only in the User Microservice.

Conceptual Architecture

The architecture used is Layered Architecture which divides the application into distinct, well-defined layers, each with a specific responsibility:

- *Presentation Layer:* top layer, responsible for handling user interface and user interaction, it includes controllers
- *Business Layer:* where the business logic of the application resides, it contains services
- *Persistence Layer:* deals with data storage and retrieval, it has repositories
- *Database Layer:* contains the entities

Both microservices use this Layered Architecture. The diagram of the whole application is illustrated below:



Deployment

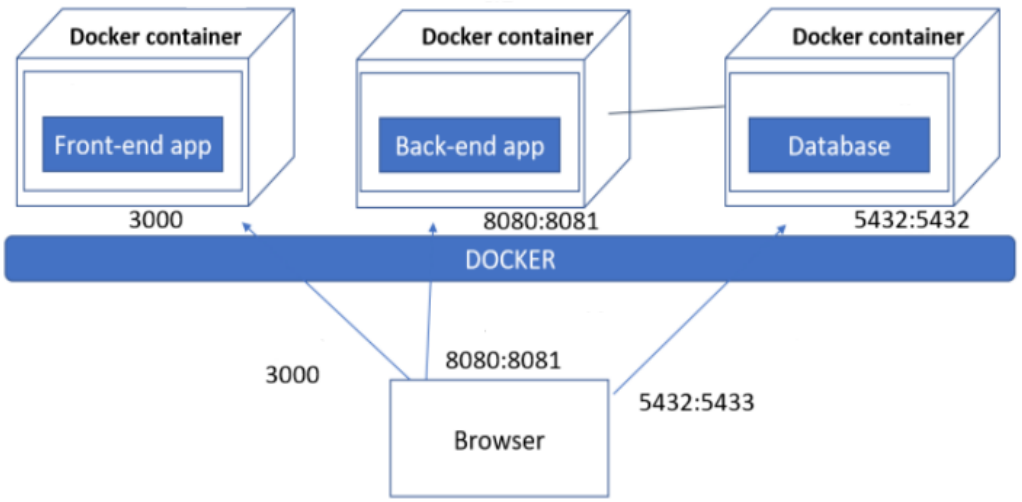
In the context of deploying the application, Docker offers a powerful and efficient solution for containerization. It enables developers to package applications and their dependencies into units called containers. These containers can run consistently across different environments.

There are three containers:

- *User backend:* including the user microservice image and user database image

- *Device backend:* including the device microservice image and device database image
- *Frontend:* including the react image

Deployment diagram



The ports are mapped in order to ensure synchronisation:

- User-microservice: 8080:8080
- Device-microservice: 8081:8081
- Frontend: 3000:3000
- User database: 5432:5432
- Device database: 5433:5432