Assignment 3
Distributed Systems Laboratory

Energy Management System

Voicu Sara-Ioana
Group: 30444

Introduction

The third assignment focused on improving the energy management system by implementing a Chat microservice that allows administrators to interact with clients. Additionally, the update included the implementation of JWT Security to safeguard all microservices, ensuring that unauthorised individuals, including clients, are unable to access the actions performed by administrators.

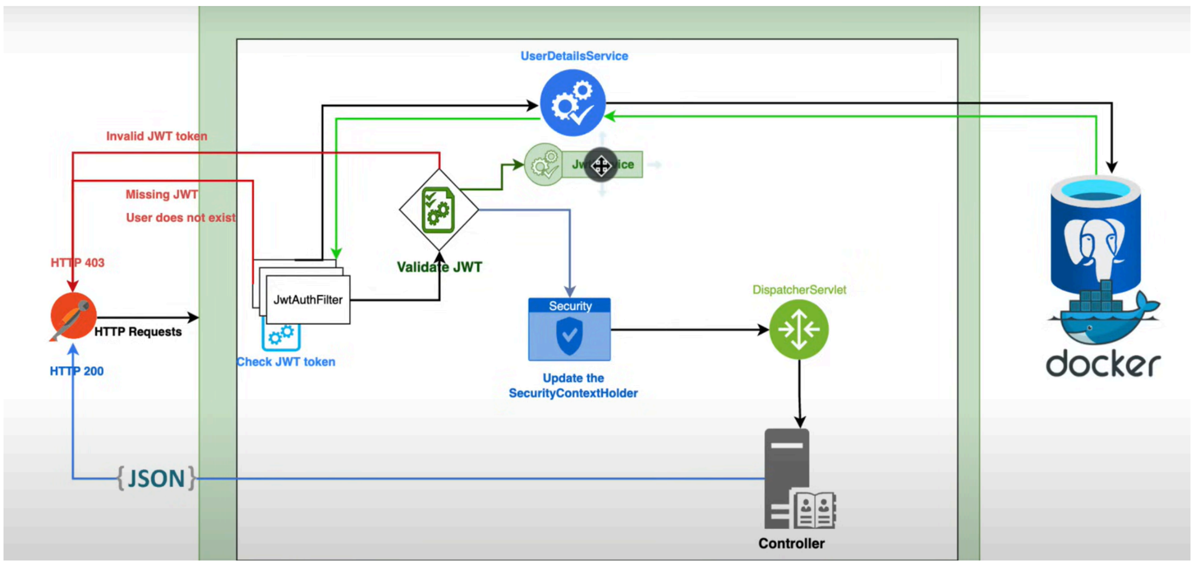
JWT Security

JSON Web Token (JWT) security in Spring is a popular approach for securing applications and microservices. Spring Security, a powerful authentication and access control framework, supports integration of JWT for token-based authentication. With JWT, a digitally signed token is generated upon successful user authentication, containing encoded information about the user and their roles. This token is then passed between the client and server for subsequent requests, allowing secure and stateless communication. Spring Security simplifies the implementation of JWT security, providing configurable options for token creation, validation, and user authentication.

Term we need to address:

- *Authentication:* refers to the process of verifying the identity of a user, based on provided credentials. It is represented by entering a username and a password when you log in to a website. You can think of it as an answer to the question: Who are you?
- *Authorization*: refers to the process of determining if a user has proper permission to perform a particular action or read a particular data, assuming that the user is successfully authenticated. You can think of it as an answer to the question: Can a user do that?
- *Granted authority*: refers to the permission of the authenticated user
- *Role*: refers to a group of permissions ot the authenticated user

This encapsulates the logic for token generation and it subsequent utilisation:



Within this application, the Security Configuration class specifies certain endpoints that are accessible exclusively to the authenticated users. Additionally, specific requests are restricted to either administrators or clients, ensuring differentiated access based on user roles.

WebSockets

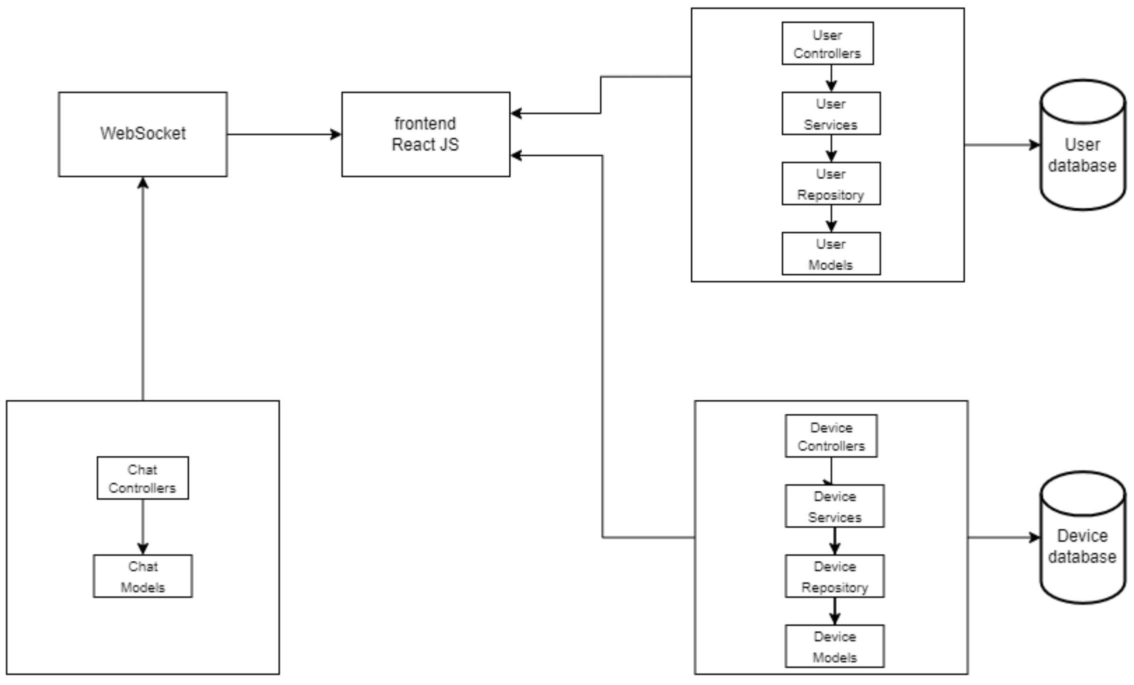
Implementing WebSocket in the monitoring microservice enables real-time communication between the backend (monitoring microservice) and the frontend of the application. WebSocket is a communication protocol that provides full-duplex communication channels over a single TCP connection, allowing for bidirectional communication between the server and the client.

How does WebSocket work in my application?

WebSocket functionality has been incorporated into the chat microservice to enable real-time communication with the frontend. The microservice accepts messages from the frontend, which include details such as the sender, receiver, and the message content. Subsequently, it dispatches these messages to an endpoint along with the recipient's name.

On the client side, the application establishes a WebSocket connection to the chat microservice. Through this connection, the frontend is notified in real-time if the user receives a message from the backend side.

Conceptual Architecture



Deployment

The application incorporated 4 components that were deployed: the frontend, user microservice, device microservice and chat microservice.

The ports mappings are configured to ensure synchronisation according to the following scheme:

User:

- User-microservice: 8080:8080
- User database: 5432:5432

Device:

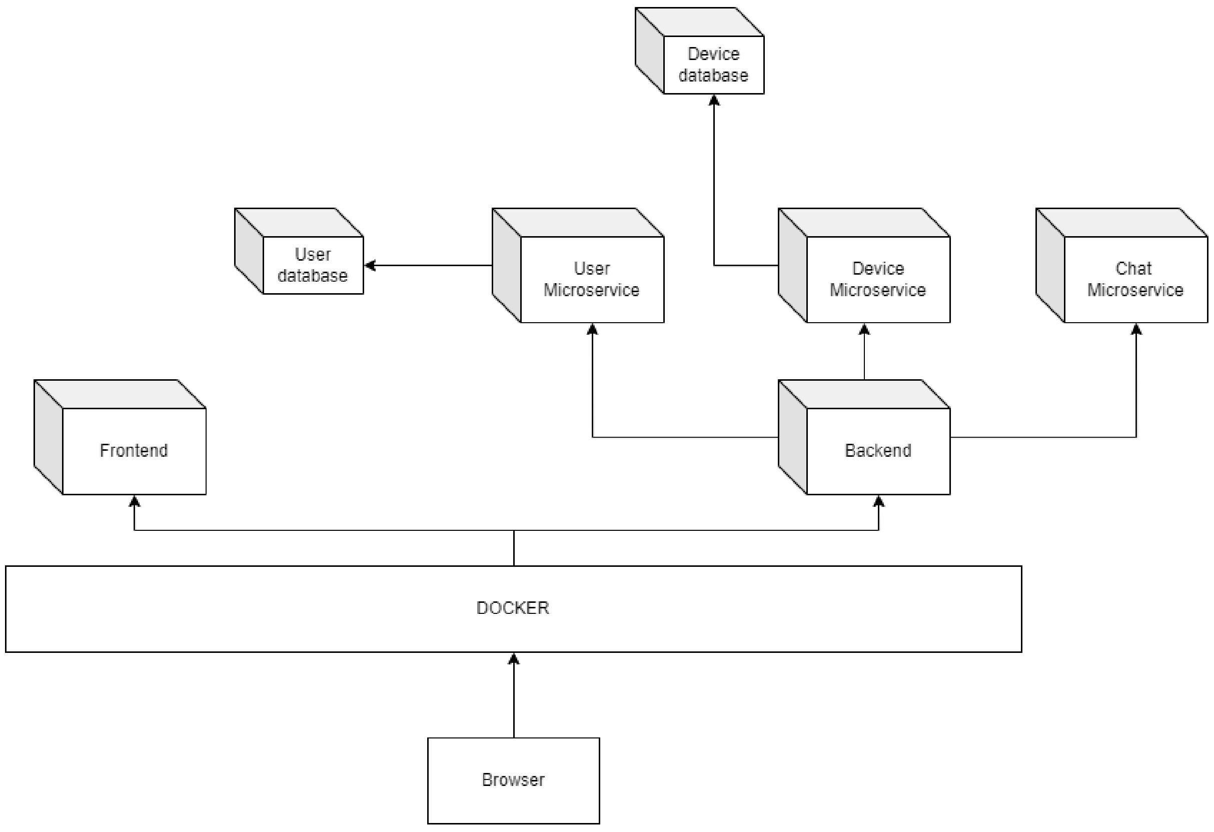
- Device-microservice: 8081:8081
- Device database: 5433:5432

Frontend:

- Frontend: 3000:3000

Chat:

- Chat: 8082:8082


Device database
User database
User Microservice
Device Microservice
Chat Microservice
Frontend
Backend
DOCKER
Browser