Integrated Energy Management System

Popovici Eusebiu-Ionut

1 Introducere

Proiectul se concentrează pe dezvoltarea unei aplicații care permite vizualizarea utilizatorilor și a dispozitivelor asociate fiecăruia. Folosind un frontend realizat în React, împreună cu două microservicii dedicate pentru gestionarea utilizatorilor și a dispozitivelor, sistemul oferă o experiență interactivă. Backend-ul este construit cu Spring Boot, iar baza de date utilizată este MySQL.

2 Structura aplicației

- Backend-ul este format din 2 microservicii, unul pentru utilizatori și celălalt pentru dispozitive. Microserviciul utilizatorilor este format dintr-o aplicație Spring Boot și o bază de date stocată în MySQL. Cel de-al doilea microserviciu este implementat pe aceeași idee, dar are două tabele în loc de unul.
- Frontend-ul este realizat în React și este structurat în 3 pachete, câte unul pentru fiecare tip de utilizator, iar cel de-al treilea reprezintă pagina de home a interfeței.
- Toate acestea sunt salvate în Docker, prin intermediul Dockerfile-urilor și a unui fișier docker-compose. Rularea se face direct din consolă, iar testarea se realizează din interfața creată în React.

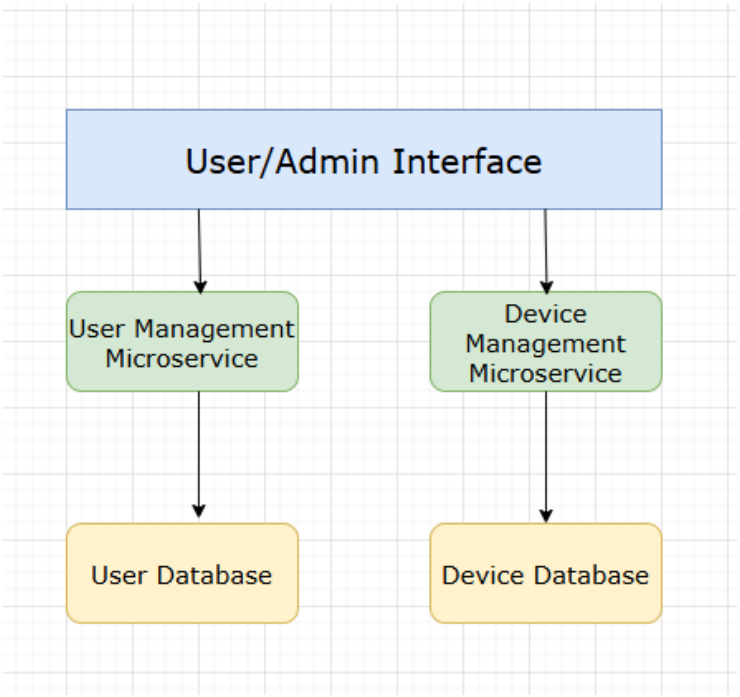

Figure 1: Diagrama de concepte

3 Implementare

1. Primul microserviciu este structurat in 7 pachete. Pachetul ‘model‘ defineste datele aplicatiei, iar celelalte pachete (‘repository‘, ‘serviceInterface‘, ‘service‘ si ‘controller‘) implementeaza functionalitatile principale pe partea de utilizator. Ultimul pachet este folosit pentru a stoca datele primite prin endpointuri in DTO-uri (Data Transfer Objects), facilitand astfel transferul datelor intre diversele componente ale sistemului.

   Pe langa acestea, microserviciul include functionalitati de comunicare intre servicii prin intermediul RestTemplate, care permite interactiunea cu celalalt microserviciu. Aceasta comunicare este folosita pentru a adauga ID-ul utilizatorului in tabelul secundar al microserviciului de dispozitive, precum si pentru a sterge utilizatorii din sistem. Procesul de stergere implica elim-

inarea datelor din baza de date a microserviciului de utilizatori, a dispozitivelor asociate utilizatorului curent si a ID-ului din tabelul secundar.

2. Al doilea microserviciu este dezvoltat pe aceeasi structura ca primul, insa nu include partea de RestTemplate, deoarece nu necesita comunicare cu alte microservicii. In fiecare pachet, sunt definite cate doua clase: una pentru tabela de utilizatori si alta pentru tabela de dispozitive, astfel incat functionalitatile acestui microserviciu sa fie structurate si clare.

3. Partea de interfata (frontend) este realizata in React si este organizata in trei pachete principale:

   - Primul pachet este destinat paginii de home, unde fiecare utilizator isi poate accesa contul. In functie de rolul stocat in baza de date, utilizatorul este redirectionat catre interfata corespunzatoare: fie componenta de utilizator, fie cea de administrator.
   - Componenta pentru utilizator permite doar vizualizarea dispozitivelor asociate contului propriu. Astfel, utilizatorii obisnuiti au acces doar la informatiile proprii, fara posibilitatea de a modifica alte date.
   - Componenta pentru administratori ofera functionalitati de gestionare a utilizatorilor si a dispozitivelor. Administratorul poate vizualiza toti utilizatorii si toate dispozitivele din sistem, poate modifica datele acestora si poate adauga elemente noi in baza de date.

4. In final, toate aceste componente au fost integrate in Docker pentru a permite o configurare usoara si un mediu de rulare uniform. Fiecare componenta dispune de un Dockerfile propriu, iar toate trei sunt legate printr-un fisier docker-compose care specifica si configuratiile pentru serviciile bazei de date. Aceasta structura permite rularea simultana a componentelor intr-un mediu izolat si facilita integrarea aplicatiei in diverse sisteme de productie.

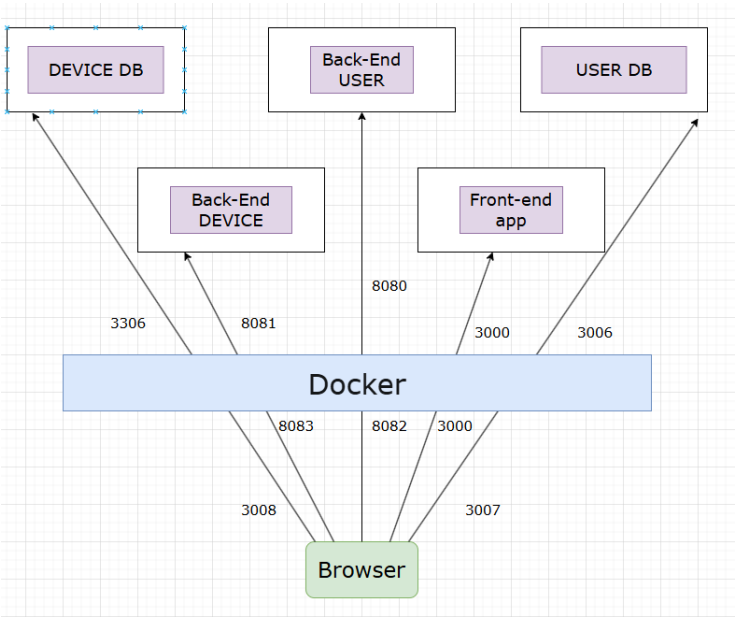

Figure 2: Diagrama de deployment

4 Funcționalități

4.1 Funcționalități interfață

4.1.1 Utilizator (User)

- Se poate loga în sistem.
- Poate vizualiza dispozitivele asociate lui.

4.1.2 Administrator (Admin)

- Se poate loga în sistem.
- Poate vizualiza toți utilizatorii.
- Poate modifica informațiile utilizatorilor existenți.
- Poate șterge utilizatori din sistem.
- Poate adăuga utilizatori noi în sistem.

- Poate vizualiza toate dispozitivele.
- Poate modifica informațiile despre dispozitivele existente.
- Poate șterge dispozitive din sistem.
- Poate adăuga dispozitive noi în sistem.

4.2 Endpointuri

4.2.1 Endpointuri : UserController

- **POST** `/user/insert`: Adaugă un utilizator nou în sistem.
- **DELETE** `/user/delete`: Șterge un utilizator din sistem.
- **POST** `/user/login`: Autentifică un utilizator.
- **POST** `/user/getID`: Obține ID-ul utilizatorului pe baza informațiilor furnizate.
- **POST** `/user/getUsers`: Obține toți utilizatorii din sistem.
- **PUT** `/user/update`: Actualizează informațiile unui utilizator existent.

4.2.2 Endpointuri : DeviceController

- **POST** `/device/insert`: Adaugă un dispozitiv nou în sistem.
- **POST** `/device/getByUser`: Obține toate dispozitivele asociate unui utilizator.
- **POST** `/device/getAll`: Obține toate dispozitivele din sistem.
- **PUT** `/device/update`: Actualizează informațiile unui dispozitiv existent.
- **DELETE** `/device/delete`: Șterge un dispozitiv din sistem.

5 Concluzie

In concluzie, aplicația dezvoltată pentru gestionarea utilizatorilor și a dispozitivelor oferă o interfață și funcționalități complete atât pentru utilizatori, cât și pentru administratori. Utilizatorii pot accesa ușor dispozitivele asociate contului lor, iar administratorii au la dispoziție un set extins de funcții pentru gestionarea utilizatorilor și a dispozitivelor. Integrarea aplicației în Docker permite o implementare rapidă și simplă, oferind în același timp flexibilitate în gestionarea resurselor.