**Energy Management System**

**DISTRIBUTED SYSTEMS**

Ilies Oana-Elena

Calculatoare, an IV, 30244

**Introducere**

Aplicatia este un sistem de gestionare a energiei bazat pe microservicii, care administreaza utilizatorii si dispozitivele acestora. Acesta include doua microservicii principale: user-service pentru utilizatori si device-service pentru dispozitive, care comunica prin API-uri REST. Fiecare microserviciu suporta operatiuni CRUD pentru gestionarea datelor. Aplicatia include un frontend usor de utilizat si un backend scalabil care asigura persistenta si integritatea datelor. Abordarea modulara faciliteaza extinderea si intretinerea sistemului.

**Implementare**

**Backend**

Backend-ul este construit folosind Spring Boot si PostgreSQL pentru stocarea datelor. Sistemul include doua baze de date separate: una pentru utilizatori si alta pentru dispozitive. Arhitectura este bazata pe doua microservicii independente, fiecare responsabil pentru un tip de entitate: user-service pentru utilizatori si device-service pentru dispozitive. Aceste microservicii comunica intre ele prin API-uri REST.

**Microserviciul user:**

* Gestioneaza datele utilizatorilor, inclusiv clienti si administratori.
* Ofera functionalitati CRUD (Create, Read, Update, Delete) pentru utilizatori.
* Clientii pot fi asociati cu mai multe dispozitive, iar administratorii pot gestiona atat utilizatorii, cat si dispozitivele din sistem.

**Microserviciul device:**

* Gestioneaza datele despre dispozitive.
* In cadrul bazei de date pentru dispozitive, exista doua tabele distincte:
  + **Tabelul de relatii**: Stocheaza asocierea dintre utilizatori si dispozitive, prin campurile userId si deviceId.
  + **Tabelul dispozitivelor**: Stocheaza atributele si caracteristicile fiecarui dispozitiv.
* Permite operatiuni CRUD pentru dispozitive si, prin intermediul unui endpoint REST din microserviciul user si biblioteca RestTemplate, verifica existenta unui utilizator inainte de a asocia un dispozitiv acestuia.

**Functionalitatea de deconectare:**

* Utilizatorii pot efectua o operatie de "deconectare" de la dispozitivele lor.
* Acest proces presupune stergerea asociatiei dintre utilizator si dispozitiv, eliminand inregistrarea din tabelul de relatii userId-deviceId.

**Frontend**

Frontend-ul aplicației este construit cu React, utilizând Axios pentru cererile HTTP și CSS pentru stilizare. La deschiderea aplicației, utilizatorul ajunge pe pagina de login, iar dacă nu are cont, poate să se înregistreze. Înregistrarea setează automat rolul ca "USER", iar utilizatorul poate apoi să se logheze.

Funcționalități cheie:

* Login și Înregistrare: Utilizatorii se pot autentifica sau înregistra cu rolul "USER". După înregistrare, se pot loga cu noile credențiale.
* Autentificare și Redirecționare: După logare, aplicația verifică rolul utilizatorului și redirecționează în funcție de rol (admin sau user).
* Admin: Adminul poate face operațiuni CRUD pe utilizatori și dispozitive, inclusiv asocierea unui dispozitiv unui utilizator.
* Utilizator (USER): Clientul poate vedea dispozitivele asociate contului său și se poate deconecta de la ele.
* Navigare și Redirecționare: react-router-dom gestionează redirecționările, protejând rutele de acces neautorizat.
* Stilizare: CSS este folosit pentru a crea un design simplu și intuitiv, ușor de utilizat.

Aceste funcționalități sunt gestionate eficient pentru a asigura o experiență fluidă și interactivă pentru utilizatori.

**Flux de Date - Energy Management System**

1. Login si Inregistrare:

* Frontend (React): Utilizatorul acceseaza pagina de login. Daca nu are cont, se poate inregistra, iar rolul va fi setat automat ca "USER". Dupa inregistrare, utilizatorul se logheaza cu noile credentiale.
* Backend (User Service): Microserviciul user-service proceseaza cererea de login sau inregistrare si returneaza un raspuns corespunzator. Dupa autentificare, utilizatorul este redirectionat pe baza rolului sau:
  + Admin - redirectionat catre pagina de administrare.
  + User - redirectionat catre pagina personalizata.

2. Microserviciul User:

* Operatii CRUD: Microserviciul user-service administreaza datele utilizatorilor si rolurile acestora (client sau admin). Adminii pot gestiona utilizatori si dispozitive, iar clientii pot vizualiza dispozitivele asociate.
* Frontend: Datele utilizatorilor sunt vizualizate pe frontend, iar utilizatorii clienti pot accesa dispozitivele lor asociate.

3. Microserviciul Device:

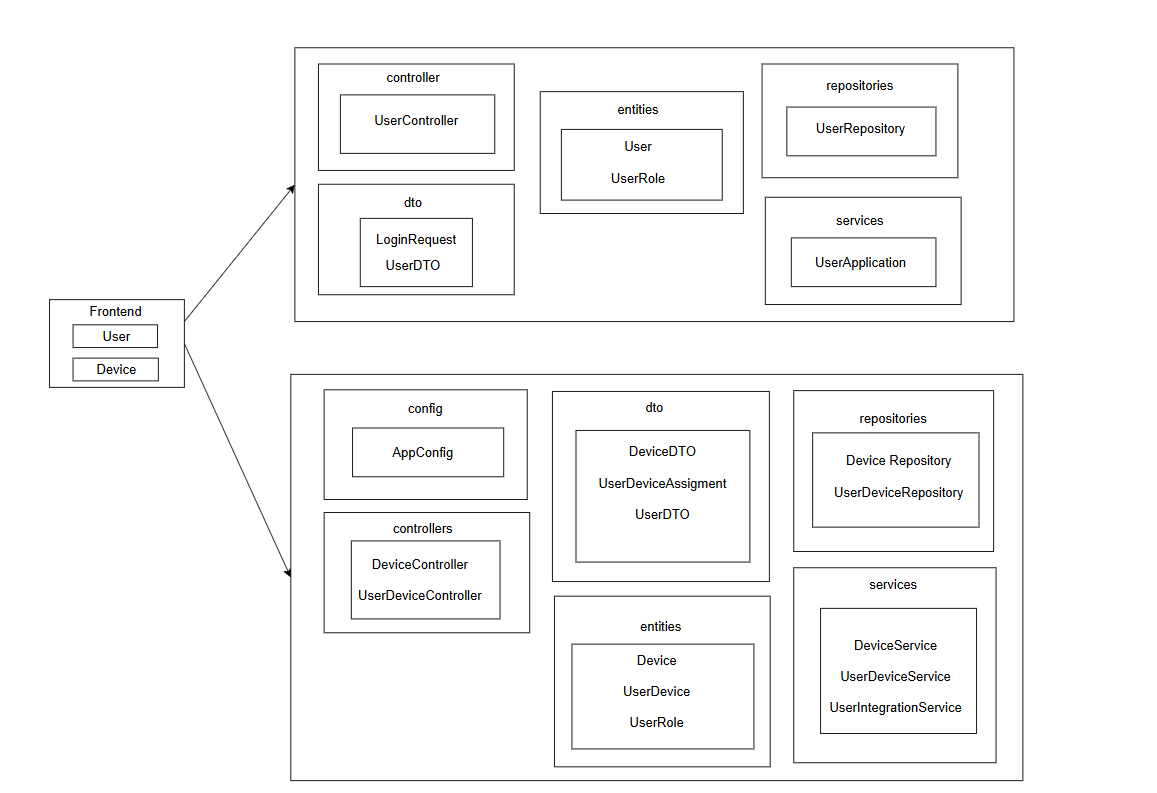
* Stocarea datelor: In baza de date a dispozitivelor sunt doua tabele:
  1. Tabelul de relatii - Stocheaza asocierea dintre utilizatori si dispozitive (userId, deviceId).
  2. Tabelul dispozitivelor - Contine detalii despre dispozitive (tip, caracteristici etc.).
* Operatii CRUD: Microserviciul device-service permite manipularea dispozitivelor si validarea utilizatorilor printr-un endpoint REST.
* Comunicare intre microservicii: Utilizand RestTemplate, microserviciul device-service trimite cereri catre user-service pentru a verifica existenta utilizatorului inainte de a asocia un dispozitiv acestuia.

4. Deconectare Dispozitiv:

* Frontend: Utilizatorul poate deconecta un dispozitiv asociat contului sau.
* Backend (Device Service): Microserviciul device-service sterge asocierea dintre utilizator si dispozitiv din tabelul de relatii.
* Frontend: Interfata se actualizeaza pentru a reflecta modificarea, iar dispozitivul nu mai este asociat utilizatorului.

5. Navigare si Redirectionare:

* Frontend (React Router): Navigarea intre pagini este gestionata prin react-router-dom, cu rute protejate pentru a restricționa accesul neautorizat.
  + Adminii au acces la pagina de administrare.
  + Utilizatorii sunt redirectionati catre pagina personalizata, in functie de rolul lor.



**Doker**

Pentru a realiza deploy-ul pe Docker, este necesar sa obtinem o imagine care va fi rulata local, pe un mediu cloud sau pe un server fizic. Pentru acest proiect, am utilizat 5 containere specifice si independente: unul pentru frontend, unul pentru baza de date db\_user, unul pentru baza de date db\_device (baze de date dedicate microserviciilor) si inca 2 pentru backend — unul pentru backendul device si unul pentru backendul user. Fiecare container ruleaza pe un port separat, asigurand o comunicare eficienta intre componentele aplicatiei.

