UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

SISTEME DISTRIBUITE

Energy Management System

Balint Cătălin 30642/1

Facultatea de Automatica
si Calculatoare

2024

Cuprins

1 Assignment1

1.1 Descriere generală

Aplicația este o platformă web dedicată gestionării utilizatorilor și dispozitivelor asociate sistemului de măsurare inteligentă a energiei, oferind funcționalități pentru înregistrare, autentificare și acces la diverse pagini în funcție de rolul utilizatorului.

Administratorul are acces la pagina de administrare , unde poate vizualiza toți utilizatorii aplicației și dispozitivele acestora sub formă de tabele si poate efectua operatiile CRUD.

Utilizatorul are acces restricționat, fără autorizarea de a accesa pagina de admin, și poate vizualiza doar propriile dispozitive pe pagina dedicată utilizatorului standard.

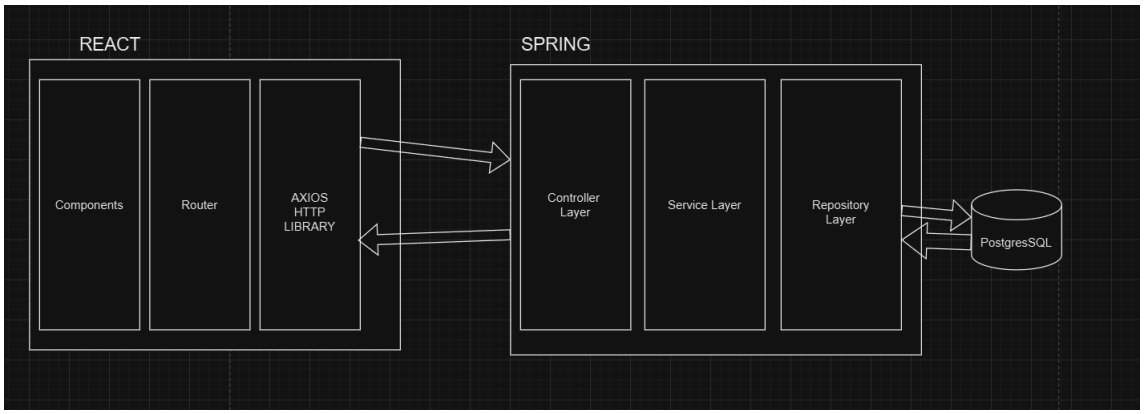
1.2 Structura aplicatiei

Aplicația este compusă dintr-un **frontend** care include un sistem de autentificare pentru utilizatori. După logare, utilizatorii sunt redirecționați în funcție de roluri către paginile `/admin` sau `/persondevice`. Fiecare utilizator poate accesa pagina dedicată, unde își poate vizualiza dispozitivele asociate.

Administratorul are capacitatea de a efectua operațiuni CRUD (Creare, Citire, Actualizare, Ștergere) atât pe tabela dispozitivelor, cât și pe cea a utilizatorilor. Acesta poate adăuga noi utilizatori și poate asocia dispozitivele cu utilizatorii în funcție de ID, conectând astfel două baze de date.

Backend-ul aplicației este construit pe Spring Boot, oferind endpoint-uri REST pentru autentificare, adăugare, inserare, ștergere, vizualizare și verificarea câmpurilor, cum ar fi validarea rolului de administrator.

Conexiunea cu baza de date permite stocarea și gestionarea utilizatorilor, asigurând o administrare eficientă a acestora.



1.3 Conexiunea la baza de date

Conexiunea la baza de date este gestionată prin fișierul de configurare `application.properties`. Parametrii de configurare definiți sunt următorii:

- `database.ip`: Aceasta este adresa IP a serverului unde este găzduită baza de date.
- `database.port`: Este portul pe care serverul PostgreSQL ascultă cererile. Valoarea implicită este `5432`.
- `database.user`: Utilizatorul implicit este `postgres`.
- `database.password`: Este parola asociată contului utilizatorului .

- `database.name`: Numele bazei de date la care aplicația se va conecta. În cazul nostru, avem `persons-db` pentru microserviciul de management al utilizatorilor și `devices-db` pentru managementul dispozitivelor.

1.4 Login

Clientul trimite o cerere `POST` la `/auth/login` . `PersonService` caută utilizatorul în baza de date. Dacă utilizatorul există, se verifică dacă parola introdusă corespunde cu cea stocată.

În funcție de rezultatul login-ului, se verifică rolul utilizatorului folosind o cerere `GET` la `/auth/isAdmin`. În funcție de rol, utilizatorul este redirecționat către pagina corespunzătoare. De asemenea, rolul utilizatorului este salvat în `localStorage` în aplicația React, permițând astfel implementarea accesului restricționat pe baza rolului utilizatorului.

1.5 Sincronizare

Pentru simplificare, am duplicat datele legate de utilizatori (ID-urile) din tabela `Person` din microserviciul de utilizatori prin crearea unei entități noi, `PersonReference`, în microserviciul de management al dispozitivelor. Aceasta va conține toate ID-urile aferente utilizatorilor.

Atunci când administratorul adaugă un utilizator nou în frontend, se va trimite o cerere `POST` la `/person-references` din microserviciul de management al dispozitivelor pentru a adăuga ID-ul utilizatorului. Similar, atunci când administratorul șterge un utilizator din baza de date, se va trimite o cerere `DELETE` pentru a șterge ID-ul utilizatorului din `personReferences`.

Este definită o relație de tip Many to One între `Device` și `PersonReference`, fiecare utilizator având posibilitatea de a avea o listă cu multiple dispozitive.

1.6 Arhitectura conceptuală

Aplicația urmează o arhitectură pe layere, incluzând layerele `Controller`, `Service` și `Repository`. Fiecare microserviciu — unul pentru utilizatori (`persons-microservice`) și unul pentru dispozitive (`devices-microservice`) — implementează această structură și are propria bază de date.

- **Controller**: gestionează cererile HTTP și definește endpoint-urile.
- **Service**: conține logica de business și gestionează sincronizarea datelor între microservicii.
- **Repository**: oferă acces la baza de date pentru operațiuni CRUD.

Structura modulară asigură scalabilitatea și independența fiecărui microserviciu.

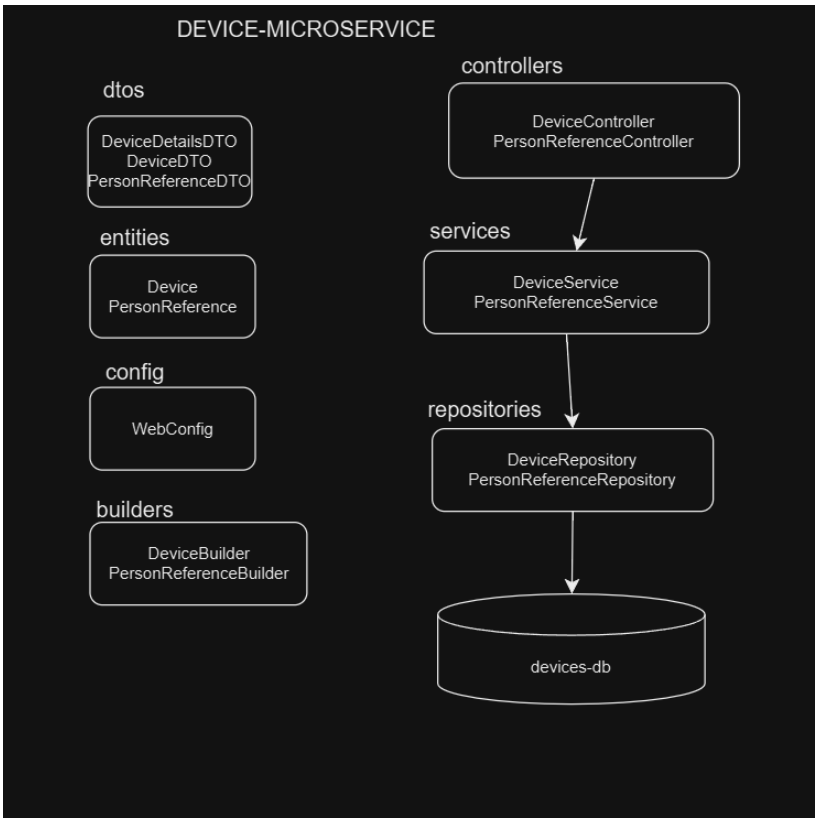


Figura 1: Device Microservice

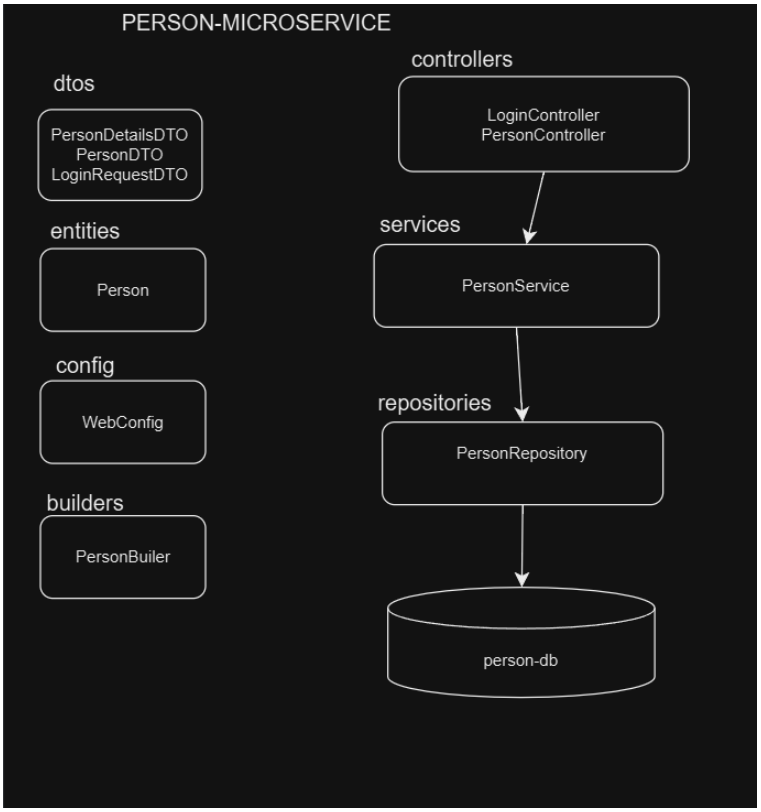


Figura 2: Person Microservice

1.7 Diagrama UML Deployment

- **FRONTEND (React - `sd-frontend`)**: Rulează într-un container Docker separat, accesibil utilizatorului la adresa `localhost:3000` prin intermediul browserului.
- **BACKEND - Microservicii**:
  - `persons-microservice`: Rulează pe portul `8080` și poate fi accesat de browser la adresa `localhost:8080`.
  - `devices-microservice`: Rulează pe portul `8081`, accesibil de asemenea în browser la adresa `localhost:8081`.
- **BAZA DE DATE (PostgreSQL)**: Rulează într-un container Docker dedicat și este accesibilă pe portul `5432`. Aceasta conține două baze de date, `persons-db` și `devices`.