**Universitatea Tehnica Cluj-Napoca**

**Facultatea de Automatica si Calculatoare**

**Departamentul Calculatoare**

Logo

Description automatically generated

**Assigment 1 : Sisteme Distribuite React + Spring Boot**

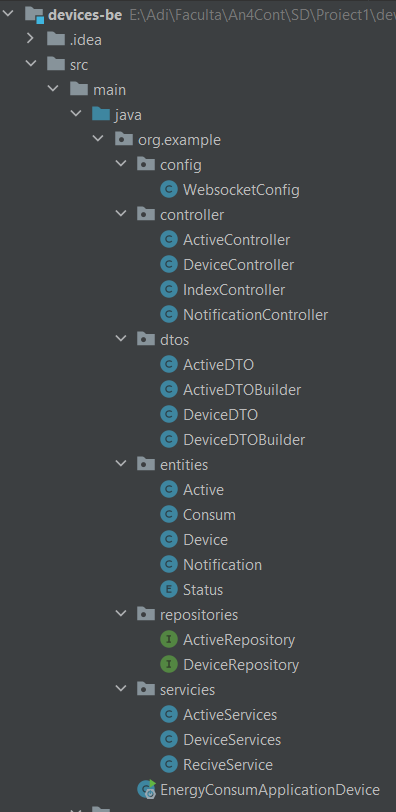
**Biz Adrian 30241**

**Asistent indrumator: Anamaria Raita**

# Cerinta

O platformă online ar trebui să fie proiectată și implementată pentru a gestiona utilizatorii, dispozitivele acestora si măsurare a energiei și datele monitorizate de la fiecare dispozitiv. Sistemul poate fi accesat de două tipuri de utilizatori după un proces de conectare: administrator (manager) și clienți. Administratorul poate efectua operațiuni CRUD (Creare-Read-Update-Delete) pe conturile de utilizator (definite prin ID, nume, rol: admin/client), dispozitive inteligente de contorizare a energiei înregistrate (definite prin ID, descriere, adresa, consumul maxim de energie pe oră) și pe maparea utilizatorilor la dispozitive (fiecare utilizator poate deține unul sau mai multe dispozitive inteligente în locații diferite). După ce maparea este făcută, pentru fiecare dispozitiv, consumul de energie este stocat pe oră sub formă de tupluri de forma <timestamp, energie consum> în baza de date.

# Functionalitati

* User login cu redirectionare
* Administrator(admin)
  + CRUD pentru useri
  + CRUD pentru devices
  + Mapare user-device
* User-ul poate vedea device-urile sale.

# Implementare Backend

Pentru implementarea backend am folosit Spring Boot + PostquestSQL. Spring Boot va folosi portul 8081 pentru utilizatori si 8082 pentru device si consum, iar PostquestSQL portul 5433. Bazele de date au fost create folosind libraria java “hipernate”.

Pentru reprezentarea datelor am folosi urmatoarele clase si entitati: Active, Device, Person, Role.

* Active cuprinde: id, id de device, timestamp si consum.
* Device cuprinde: id, id owner, descriere, adresa, consum standard.
* Person cuprinde: id, nume, parola, rol.
* Rol este o enumeratie de cele 2 posibilitati user sau admin.

A screenshot of a computer

Description automatically generatedRelatiile dintre tabelele bazei de date sunt urmatoarele:

* Fiecare Device are un owner, o persoana poate avea mai multe device-uri.
* Fiecare Active este dependent de un device, un device poate fi activ in mai multe momente.

In cadrul aplicatiei server am folosit o arhitectura MVC pentru a separa functionalitatile si modulariza procesele. Astfel am diferite pachete: “controller” se ocupa de endpoint si inpachetarea datelor in formatul necesar raspunsului, “dtos” defineste obiectele de transfer pentru front-end, “entities” sunt entitatile din baza de date, “repositories” cuprinde requesturi pentru baza de date, iar “services” defineste partea de business logic.

# Implementare Frontend

**Graphical user interface, application

Description automatically generated**

Aplicatia client a fost dezvoltata cu functionalitatea de pagina web in react.

Primul lucru pe care un utilizator-l va vedea va fi pagina de login. De unde poate crea cont de utilizator sau sa se logere.

Daca ofera credentialele de admin va fi trimis la pagina de admin ce poate face operatii CRUDE pe users sau devices. Iar daca ofera credentiale de utilizator va fi trimis pe o pagina unde va putea vedea device-urile associate utilizatorului dat si va putea selecta o data in care sa vada un graphic cu consumul din acea zi bazat pe activele associate device-ului.

Daca cineva va incerca sa acceseze pagina de admin fara sa ofere credentialele adecvate va fi trimis pe o pagina de eroare “Nu mai fi bulangiu”.

# Diagrame

## DataBase

A diagram of a data flow

Description automatically generated with medium confidence

## Arhitectura

Diagram

Description automatically generated

## Deploy

Chart, box and whisker chart

Description automatically generated