**Universitatea Tehnica Cluj-Napoca**

**Facultatea de Automatica si Calculatoare**

**Departamentul Calculatoare**

Logo

Description automatically generated

**Assigment 2 : Asynchronous Communication**

**and Real-Time Notification**

**Biz Adrian 30241**

**Asistent indrumator: Anamaria Raita**

# Cerinta

# Implementati un Microserviciu de Monitorizare și Comunicare pentru Sistemul de Management al Energiei.

# Microserviciul se bazează pe un middleware de broker de mesaje care colectează date de la dispozitivele de măsurare inteligente, procesează datele pentru a calcula consumul energetic pe oră și le stochează în baza de date a Microserviciului de Monitorizare și Comunicare.

# Sincronizarea între bazele de date ale Microserviciului de Management al Dispozitivelor și noul Microserviciu de Monitorizare și Comunicare se realizează printr-un sistem bazat pe evenimente care folosește un topic pentru modificările dispozitivului (trimite informații despre dispozitiv printr-o coadă către Microserviciul de Monitorizare și Comunicare).

# O aplicație Simulator de Dispozitiv de Măsurare Inteligentă va fi implementată ca Producător de Mesaje. Va simula un contor inteligent citind date energetice dintr-un fișier sensor.csv (adică o valoare la fiecare 10 minute) și trimite datele în formă <timestamp, device\_id, measurement\_value> către Brokerul de Mesaje (adică o coadă). Timestamp-ul este luat de la ceasul local, iar device\_id-ul este unic pentru fiecare instanță a Simulatorului de Dispozitiv de Măsurare Inteligentă și corespunde device\_id-ului unui utilizator din baza de date (așa cum este definit în Asignarea 1). Simulatorul de dispozitiv ar trebui să fie dezvoltat ca o aplicație independentă (adică o aplicație desktop). Fișierul sensor.csv poate fi descărcat de la https://dsrl.eu/courses/sd/materials/sensor.csv.

# Măsurătorile sunt trimise către coadă folosind următorul format JSON:

# {

# "timestamp": 1570654800000,

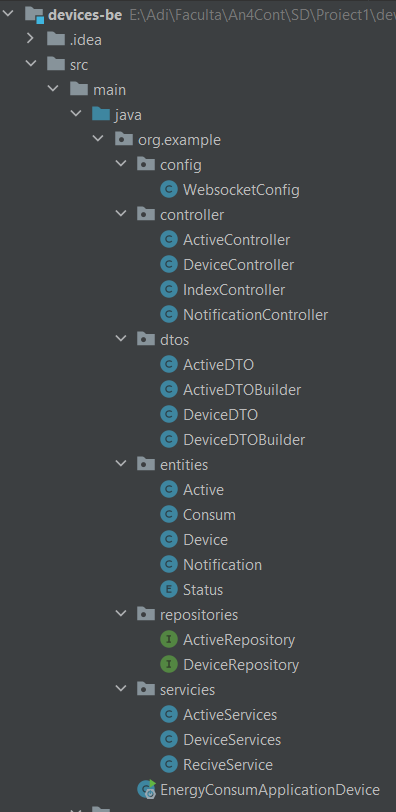
# "device\_id": "5c2494a3-1140-4c7a-991a-a1a2561c6bc2",

# "measurement\_value": 0.1,

# }

# Microserviciul de Monitorizare și Comunicare va avea un component Consumator de Mesaje care va procesa măsurătorile pentru a calcula consumul total de energie pe oră și îl va stoca în baza de date. Dacă consumul total de energie pe oră calculat depășește valoarea maximă definită pentru dispozitiv (așa cum este definit în Asignarea 1), acesta notifică asincron utilizatorul pe interfața sa web.

# Functionalitati

* User login cu redirectionare
* Administrator(admin)
  + CRUD pentru useri
  + CRUD pentru devices
  + Mapare user-device
* User-ul poate vedea device-urile sale.
* Middleware-ul orientat către mesaje permite Simulatorului de Dispozitiv de Măsurare Inteligentă să trimită tuple de date în format JSON.
* Componenta consumator de mesaje a microserviciului procesează fiecare mesaj și notifică asincron aplicația client folosind WebSocket.
* Valorile energetice pe oră vor fi salvate de componenta consumator în baza de date de Monitorizare.

# 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.

A screenshot of a computer

Description automatically generatedDaca 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”.

# Implementarea Monitor Measurements

Aplicatia measurements citeste mesajele dintr-o coada din servarul CloudAMQP .

Aceasta le proceseaza si salveaza consumul pe ora si actualiseaza valoarea curenta in treshholdul deviceului.

Interfata trimite un mesaj de actualizare de treshhold sau masuratori atuunci cand se fac schimbari la device.

# A screenshot of a computer Description automatically generatedImplementare Senzor

Senzorul citeste valori din un fisier .csv si le trimite in coada de rabbitMQ ca masuratori din ora in ora incepand cu prima ora a zilei curente.

# Diagrame

## DataBase

A diagram of a data flow

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated

## Arhitectura

Diagram

Description automatically generated

## Deploy

Chart, box and whisker chart

Description automatically generated

## RabbitMQ

A diagram of a diagram of a diagram

Description automatically generated