UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Sisteme Distribuite

-Partea 2-

Sistem de monitorizare a senzorilor și notificare în timp real

NUME: Gavra
PRENUME: Anamaria
GRUPA: 302644

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Table of Contents

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Cerințele rezolvate

Această fază a proiectului presupune implementarea unei componente pentru aplicația „ Assignment 1” bazată pe un middleware de broker de mesaje care adună date de la dispozitivele de contorizare inteligentă, preprocesează datele pentru a calcula consumul orar de energie și le stochează în baza de date.

Un modul Smart Metering Simulator va fi Producatorul de mesaje. Acesta va simula un senzor prin citirea datelor de energie dintr-un fișier (sensor.csv - o valoare la fiecare 10 minute) și trimite date sub forma < timestamp, device_id, measurement_value > către Message Broker (adică, coada). Marca temporală este preluată din ceasul local, valoarea_mesuramentului este citită din fișier și reprezintă energia măsurată în kWh, iar id-ul dispozitivului este unic pentru fiecare instanță a Simulatorului dispozitivului de contorizare inteligentă și corespunde id-ului dispozitivului al unui utilizator din baza de date (așa cum este definit în sarcina 1). Simulatorul de senzori ar trebui dezvoltat ca o aplicație autonomă (adică, aplicație desktop).

O aplicație Message Consumer va preprocesa datele pentru a calcula consumul total de energie pe oră și le va stoca în baza de date. În cazul în care consumul total de energie pe oră calculat depășește valoarea maximă a dispozitivului inteligent (așa cum este definită în Misiunea 1), acesta anunță utilizatorul în mod asincron pe interfața sa web.

Arhitectura conceptuală a platformei online.

Aceasta aplicație a fost creeata la baza unei arhitecturi de tipul “client-server”; astfel, are in componenta sa două aplicații: Prima parte are rolul de a implementa operatiile specifice unui **server** iar cealaltă este specifica unui utilizator de tip **client**.

Arhitectura client / server este o arhitectură de calcul producător / consumator în care serverul acționează ca producător și client ca consumator. Serverul găzduiește și oferă la cerere servicii de înaltă performanță, consumatoare de calcul. Aceste servicii pot include acces la aplicație, stocare, partajare fișiere, acces la imprimantă și / sau acces direct la puterea de calcul brută a serverului.

Aceasta parte a aplicatiei este alcatuita din trei module principale:

- message producer
- message broker
- message consumer

Producatorul de mesje are rolul de a citi datele colectate de senzor din fisierul “Sensor.csv”, de a le transforma in format json adaugand informatiile necesare pentru un obiect de tip “Consumption” si de a le pune in coada “Coada2” din RabbitMQ la un interval de 10 secunde.

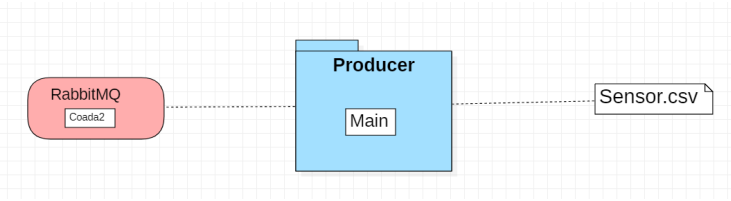

Figure 1: Producer

1 Frontend

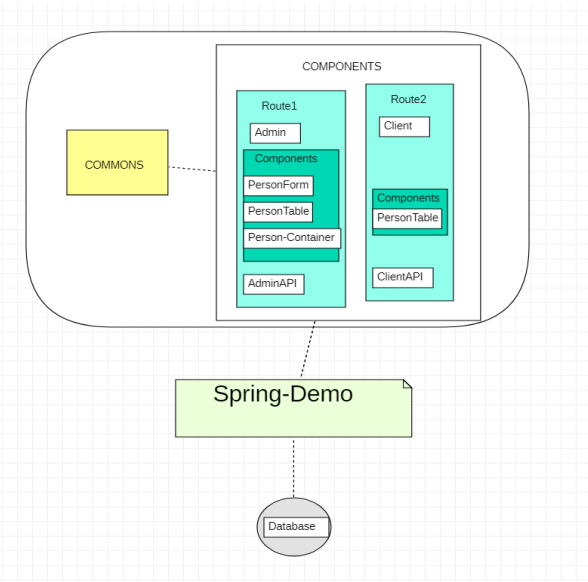

Figure 2: Frontend

2 Backend(Consumer)

Consumatorul de mesaje are rolul de a extrage datele transmise de producatorul de msaje din coada, de a le procesa si introduce in baza de date. Pe langa acestea, mai are si responsabilitatea de a valida datele ca, in cazul in care nu sunt in concordanta cu valorile existente, sa transmita un mesaj de eroare catre partea de front-end pentru a fi afisat in interfata administratorului.

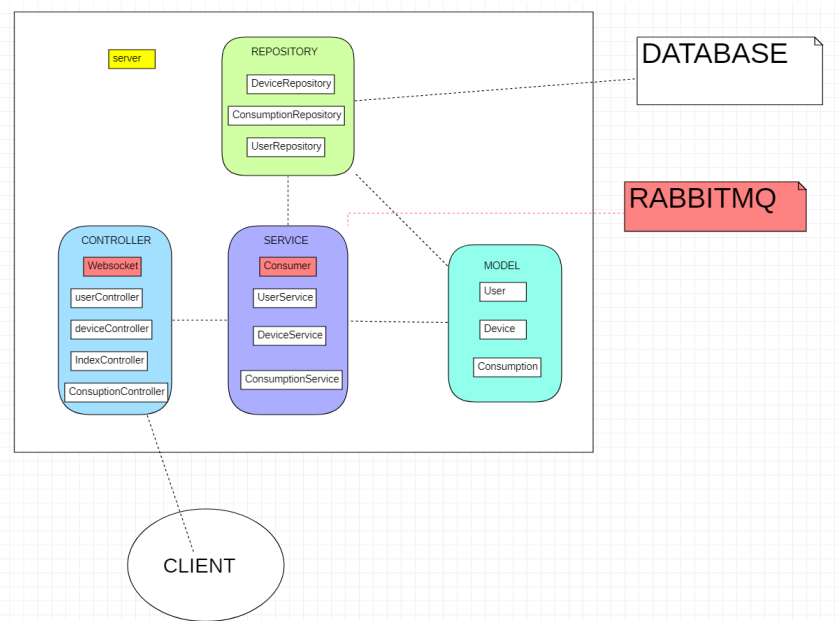

Figure 3: Back-end

Proiectare Bazei de date.

Pentru salvarea datelor am utilizat un sistem de baze de date relationale si anume PostgrreSQL.

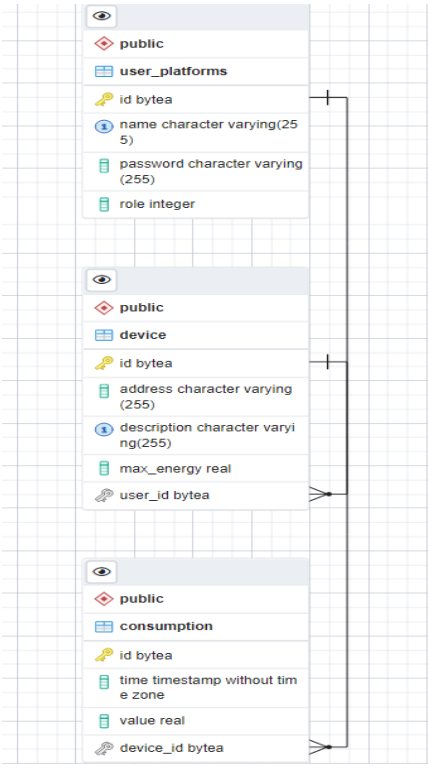

Figure 4: Database

Tabele

Baza de date a acestui proiect contine trei tabele:

- Tabelul “user_platforms” care contine credentialele utilizate de fiecare user pentru autentificare, dar si rolul acestuia.
- Tabelul “device” in care sunt memorate date specifice dispozitivelor inteligente care sunt monitorizate (descrierea, adresa la care este amplasat, consumul maxim de energie /ora)
- Tabelul “consumption” care memoreaza valorile masuratorilor periodice ale fiecarui device.

Relatii

Intre tabelele “user_platforms” si “Device” exista o relatie de tipul “one to many”, deoarece fiecare utilizator poate sa aiba unul sau mai multe duspozitive atribuite, dar fiecare device poate sa aiba un singur utilizator.

Acelasi tip de relatie exista si intre tabelele “device”, respectiv “consumption”, intrucat fiecarui dispozitiv ii corespund mai multe masuratori, dar o masuratoare este caracteristica unui singur dispozitiv inteligent.

Diagrama UML de Deployment

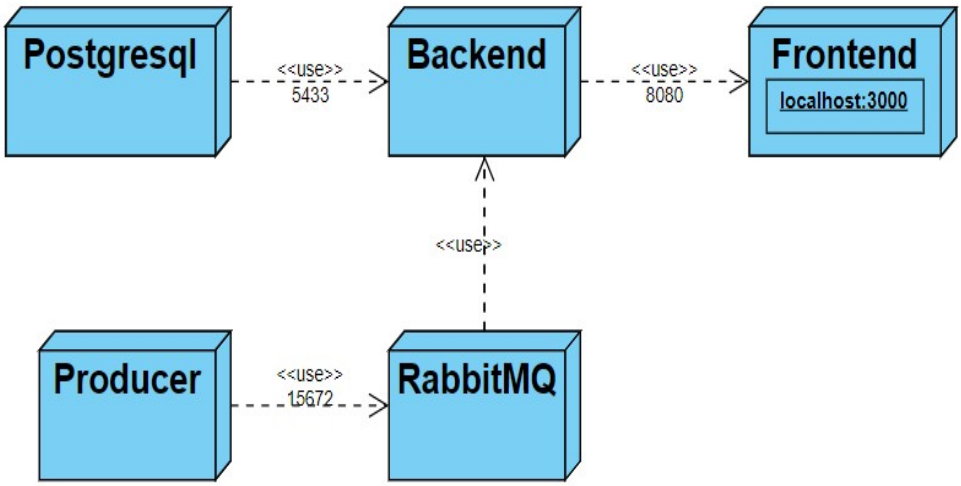

Figure 5: Deployment diagram

Considerente de construcție și execuție

Se lansează pe rând în execuție cele doua componente principale ale proiectului: clientul și serverul.

1 Baza de date

Pentru a putea porni proiectul, avem nevoie de o baza de date, creeata cu ajutorul sistemului de baze de date PostgreSQL, cu denumirea „energy”

2 RabbitMQ

Se începe prin pornirea containerului „RabbitMQ” pentru a putea utiliza coada „coada”. Pentru a creea o noua coada sau a modifica coada existența se acceseaza link-ul „localhost:15672”, se introduc credentialele: username=„user” și password=”password” .

3 Backend Spring Boot

Partea de backend a fost implementata în limbajul Java 11.

Pasii pentru pornirea proiectului sunt următorii:

- descarcat codul sursa de pe GitHub
- deschiderea proiectului într-un mediu de dezvoltare integrat ca IntelliJ IDEA
- setarea variabilelor de conexiune la baza de date (ip, username, password, etc.)

Aplicația ruleaza pe portul 8080. În cazul în care acesta este ocupat se poate întrerupe executia procesului de pe acel port deschizand „Command Prompt” ca administrator și ruland comenzile:

- „netstat -ano |findstr 8080” - pentru găsirea procesului ce ruleaza pe portul 8080
- „taskkill /PID id /F” - pentru oprirea procesului găsit anterior

4 Frontend React

Pentru realizarea acestei părți s-a utilizat Node Js, versiunea 14.

Pentru a putea rula proictul, trebuie urmați următorii pași:

- descarcat codul sursa de pe GitHub
- deschiderea proiectului într-un mediu de dezvoltare integrat ca IntelliJ IDEA sau Visual Studio Code
- deschiderea unui terminal și instalarea pachetelor necesare, executand comanda: npm install
- se lansează aplicația cu ajutorul comenzii : npm start
- acesarea din browser a adresei: localhost:3000