Online Energy Utility Platform
Assignment 2

Profesor îndrumător:
Alex Rancea

Student: Nicoară Marusea Ioana
Grupa: 30242

Universitatea Tehnică din Cluj-Napoca
Facultatea de Automatică și Calculatoare

Disciplina: Sisteme Distribuite

1. Cerintele aplicatiei

Prima a proiectului consta in realizarea unei paltforme online ce are ca scop a gestiona intr-un mod inteligent si eficient clientii acestei platforme si dispozitivele acestora ce au rolul de a monitoriza consumul de energie electrica.

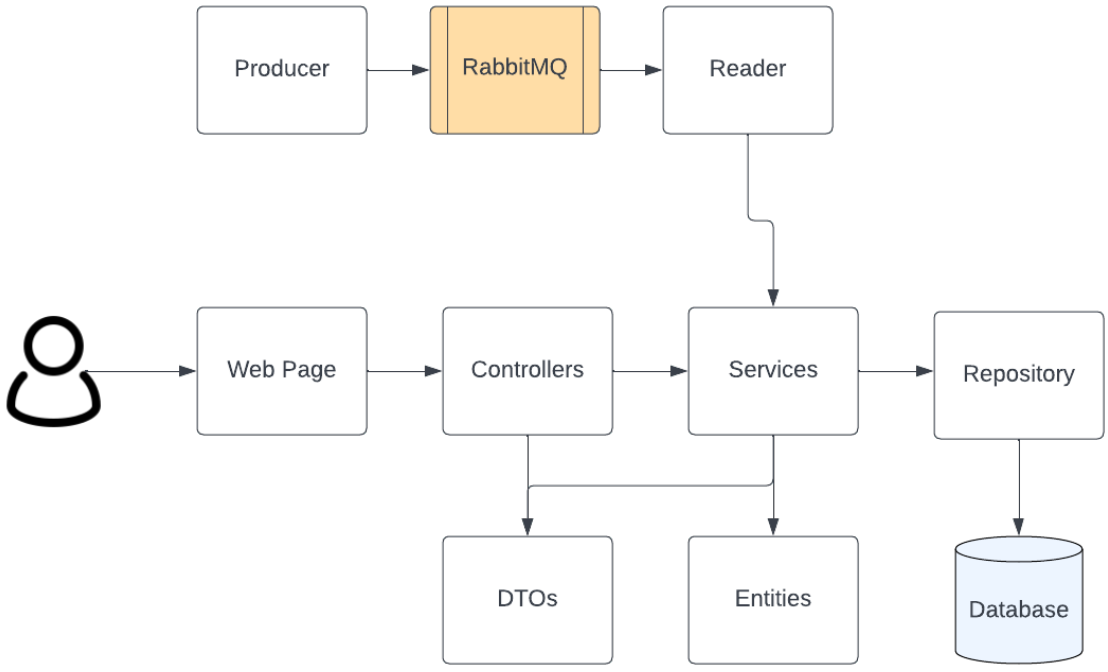
Cea de-a doua parte constă în simularea unui senzor, valorile pentru simulare (sunt citite din cadrul unui fisier csv (citite ca fiind măsuratorile de pe dispozitive).

S-a implementat un sistem de tip midleware de broker de mesaje. Acest tip de sistem este alcatuit din trei mari componente, producer, broker si consumer.

Producerul consta in aplicatia desktop ce simuleaza masuratorile, acesta genereaza mesajele ce vor fi trimise la coada din Cloud, care reprezintă parte de de broker a sistemului, Producerul construieste un json, care are drept campuri id-ul deviceului, valaorea citita din fisierul csv care e valoarea masurata a energiei si data la care s-a inregistrat acesta valoare, aceasta data fiind data locala luata din sistemul pe care ruleaza aplicatia.

Componenta de consumer este integrata in aplicatia de backend, acesta componenta, avand rolul de a citi mesajele din coada, de a le procesa si de a inregistra masuratoarea in baza de date.

2. Concept arhitectural



Pentru implementarea prime componente din cadrul sistemului am ales a utiliza o arhitectura de tip client-server. Astfel am structurat proiectul in doua componente majore si anume backend si frontend.

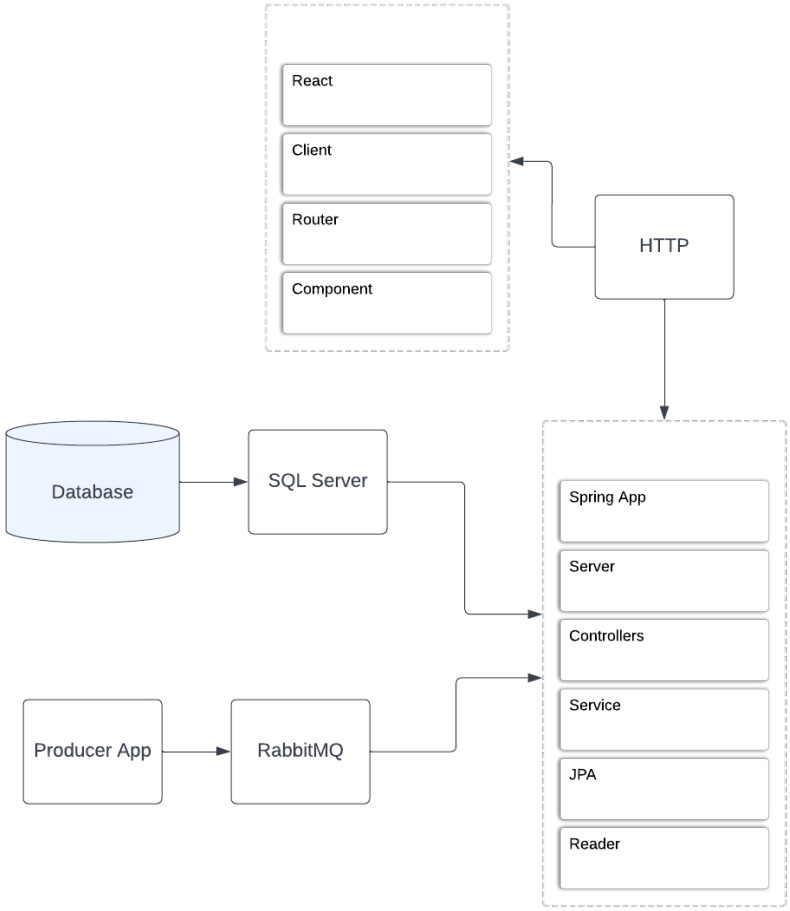
La cea de-a doua componentă mai avem și o aplicație pentru generarea mesajelor și coada RabbitMQ.

In proiectul ce se ocupa de partea de generare de mesaje exista o singura clasa cu cate 2 metode, o metoda se ocupa cu construirea mesajului, iar o alta metoda se ocupa cu crearea conexiuni cu RabbitMQ si transmiterea mesajului către coada.

In cadrul proiectului de backend exista o clasa cu o metoda ce se ocupa de crearea conexiuni cu coada, procesarea mesajelor primite si salvarea simularilor de masuratoare in baza de date.

3. Diagrama UML pentru deployment

Acesta diagrama reprezinta comunicarea intre componenta client si server din cadrul arhitecturii client server. Componenta de backend comunica cu partea de frontend prin data transfer objects (DTO). Serverul de baze de date cu care se comunica prin intermediul repositoriurilor, acesta preia informatiile de la aceste interogari. Clientul comunica cu serverul prin intermediul protocolului HTTP.



Componenta Producer Appp are ca scop sa genereze mesajele pentru coada, ele reprezentand simularea masuratorilor. Componenta RabbitMQ este coada ce se ocupa de retinerea mesajelor, iar componenta Reader se ocupa cu citirea mesajelor din coada si procesarea acestora.

Partea de deployment a fost realizata utilizand *GitHub Actions,* iar aplicația rulează în cloud-ul oferit de Heroku. Aplicația de generare a mesajelor este rulată local, iar coada RabbitMQ se află în Cloud by default.

4. Readme

Pentru rularea aplicatiei local, este necesar a se crea 2 foldere, unul pentru backend si unul pentru frontend. Se cloneaza repositoriul git aferent fiecăruia. Se deschide proiectul de backend si se ruleaza din IDE, iar apoi se deschide proiectul de frontend si se ruleaza in terminalul din IDE comanda: npm start. Se accesează localhost:3000.

Pentru accesarea aplicatiei din Cloud, se accesează link-ul:

https://ds2-frontend.herokuapp.com

Aplicația de generare a mesajelor se ruleaza local.