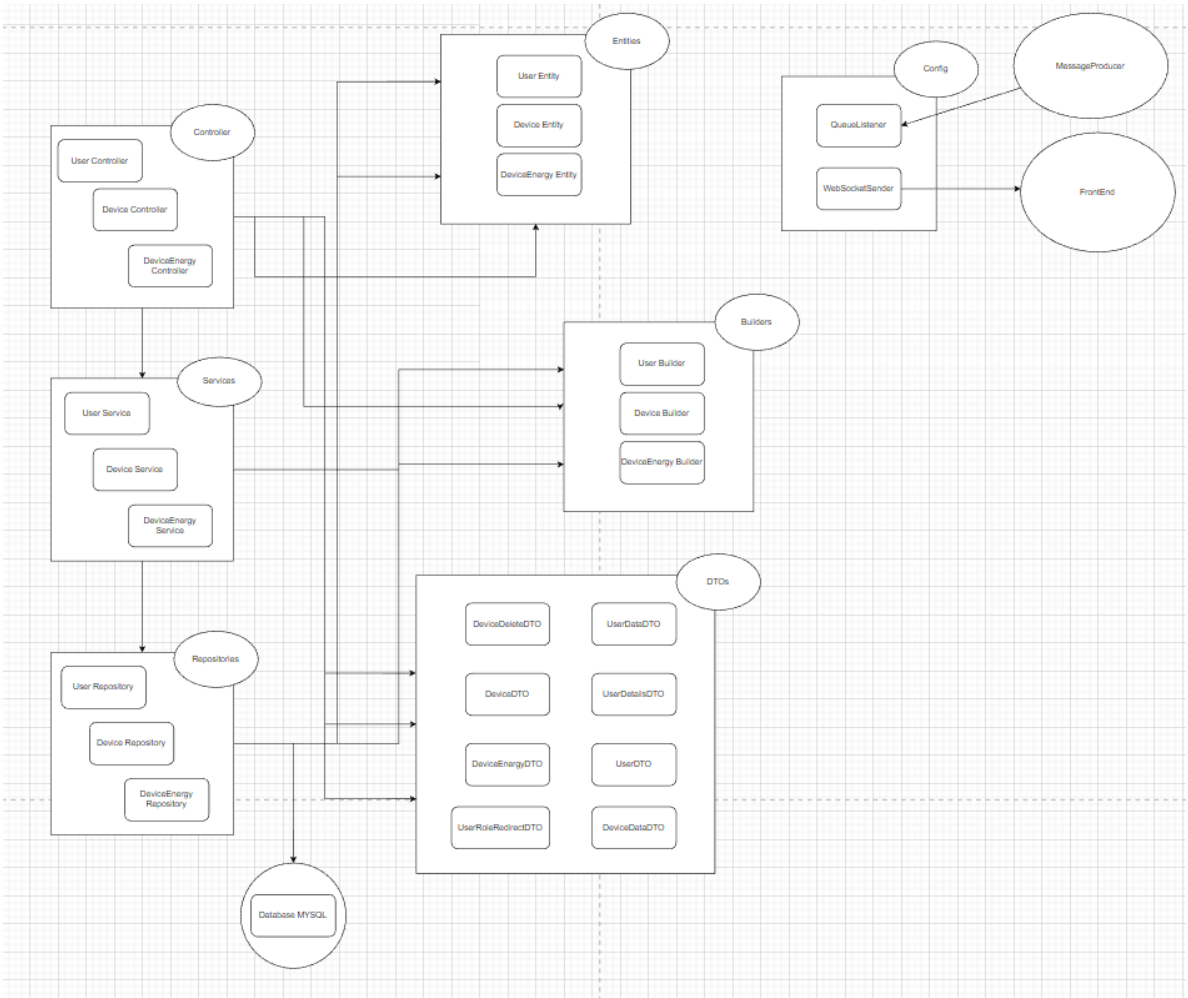
Aplicatie de monitorizare a energiei pentru device-uri

Orsan Tudor Alexandru, 30641, TI

Tema 2 SD

Aceasta aplicatie a fost creata pentru a monitoriza cat de multa energie consuma device-urile clientilor ce folosesc aplicatia. Scopul aplicatiei este sa poata fi rulata de pe cloud, si sa poata monitoriza aceste date in real time.

1) Arhitectura conceptuala: (Actualizata fata de tema 1)

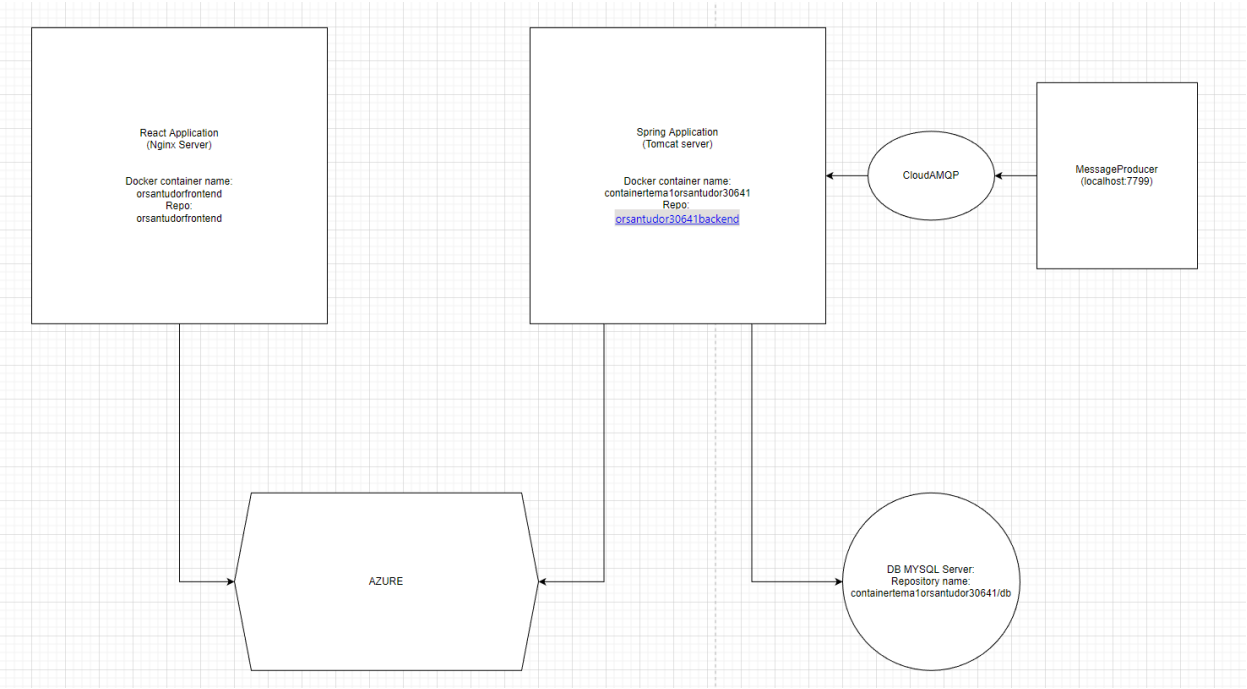


Pentru aceasta diagrama:

Acestea sunt aditiile aplicatiei:

- De aceasta data, ce am creat si dezvoltat a fost o aplicatie de Message Producer, unde am stocat date luate dintr-un fisier CSV. Am citit din CSV file pe rand date, si in functie de un TXT unde salvez ID-ul device-ului dorit, creez DTO-uri si le trimit mai departe, convertindu-le in JSON-uri, catre RMQ (Acesta fiind facut in CLOUD).
- Acum ca avem datele in AMQP, ele sunt stocate asa cum se primesc. In loc de 10 minute si 1 ora, am ales sa trimit o data la fiecare 10 secunde, si sa procesez datele in MessageListener o data pe minut (pentru o testare mai rapida). Pana cand se deschide aplicatia de Listener, datele raman salvate in RMQ si sunt gata de procesare.
- Atunci cand se deschide listener-ul, datele vin din RMQ si se proceseaza deodata. Se creeaza o lista in controller-ul DeviceEnergy, unde sunt stocate toate JSON-urile primite din RMQ. La fel ca in MessageProducer, o metoda Scheduled va procesa datele la fiecare minut. Se aranjeaza toate datele primite in functie de minutul in care au fost primite, si se face suma energiei consumate pe minut. Astfel se poate observa daca este mai mare sau nu decat maximul acceptat de acel device.
- Acum ca am procesat datele (adica am si inserat datele in BD pentru a putea fi observabile din Frontend la chart-ul unui device detinut de un user), tot ce mai ramane de facut este comunicarea prin WebSocket. Ce am facut aici este sa trimit acel DTO pentru care valoarea pe minut este mai mare decat maximul, si restul este procesat in Frontend. Daca este intrecut maximul, pe pagina clientului de frontend, se va primi un mesaj cu informatiile device-ului anume, la ce timp si cu ce value a intrecut. Astfel, clientul va stii sa faca management mai bine data viitoare.
- Inafara de folosirea cozii cu Cloud RMQ si de comunicarea prin WebSocket, nu am adaugat alte functionalitati.

2) Diagrama de deploy:



Pentru aceasta diagrama:

Ce am adaugat nou pentru deploy:

- De aceasta data, pentru deploy am folosit aceleasi componente ca pentru tema 1. Backend-ul si Frontend-ul raman neschimbate. In schimb, pentru ca folosim coada de procesare a mesajelor, a trebui sa folosesc Cloud RMQ, deci am creat cozile tot intr-un mediu cloud, unde se primesc si unde se trimit dupa datele mai departe.
- Doar partea de coada s-a facut pe Cloud, pentru a putea fi trimise datele mai departe pe partea de backend si pentru a putea fi persistate in BD pe cloud (nu era corect daca pe Cloud se primeau direct date de la localhost RMQ). Partea de citire a datelor din CSV si de trimitere mai departe la RMQ, s-a facut local. O alta aplicatie locala, atunci cand este pornita, trimite datele mai departe la RMQ, si de acolo totul se intampla pe cloud.

In final, mi s-a parut foarte interesant faptul ca toate aceste componente lucreaza impreuna pe cloud, fara sa trebuiasca manage-uite aplicatiile local. Chiar daca se trimit datele de pe local, tot mi se pare foarte interesant modul de procesare a datelor si eficienta cozii RMQ.