ASSIGNMENT 1

1. REACT APP

Aplicația pe partea de front end este facuta in React JS. Aceasta este împărțită în mai multe componente, în file-uri diferite. În file-ul admin avem toate componentele specifice administratorului. Aceasta este împărțită în adminHome și navigation-bar, 2 componente react specifice doar administratorului și în alte 2 fișiere, user și device. În acele file-uri se găsesc componente specifice pe partea de user (show users, update, delete, insert) cât și pe partea de device. Un alt file principal este cel de assets unde se găsesc imaginile, iar ultimul este cel de login care va fi comun atat pentru administrator cât și pentru basic user. Ultimul file principal este cel de commons, unde gasim impartite api urile integrate impartite pe user si device.

Aplicatia front end apeleaza api-uri pentru toate aceste actiuni, de unde isi ia datele din backend, respectiv de la baza de date. Se folosesc requesturi de post, get si delete.

2. SPRING APP

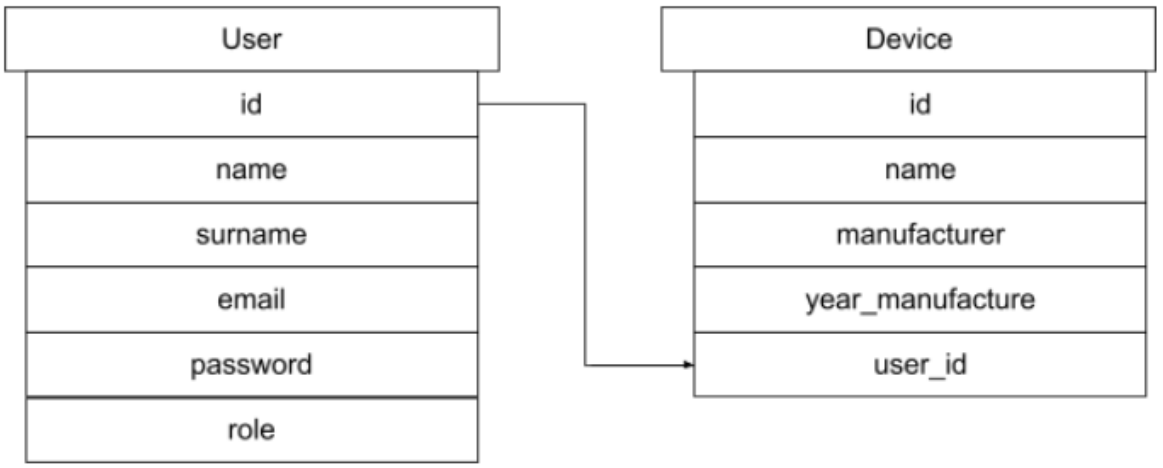
Partea de backend este facuta cu ajutorul spring boot. Aceasta aplicatie construieste api-urile pe care le apelam pe front end.

- localhost:8080/users (GET - get all users)
- localhost:8080/user/userByEmailPass (POST - primeste email-ul si parola si returneaza id-ul utilizatorului si tipul acestuia)
- localhost:8080/users/insertProsumer (POST - insereaza un nou user in tabela users)
- localhost:8080/users/{id} (GET - returneaza userul cu id-ul specificat ca parametru)
- localhost:8080/users/delete/{id} (DELETE - sterge din tabela userul cu id-ul specificat ca parametru)
- localhost:8080/users/update (POST - face update la userul pe care il primeste in body. Acesta trebuie sa contina id-ul conform caruia il va cauta in tabela pentru a updata datele trimise)
- localhost:8080/device (GET - get all devices)
- localhost:8080/device/insertProsumer (POST - insereaza un nou devicein tabela device)
- localhost:8080/device/{id} (GET - returneaza deviceul cu id-ul specificat ca parametru)
- localhost:8080/device/delete/{id} (DELETE - sterge din tabela deviceul cu id-ul specificat ca parametru)
- localhost:8080/device/update (POST - face update la deviceulpe care il primeste in body. Acesta trebuie sa contina id-ul conform caruia il va cauta in tabela pentru a updata datele trimise)

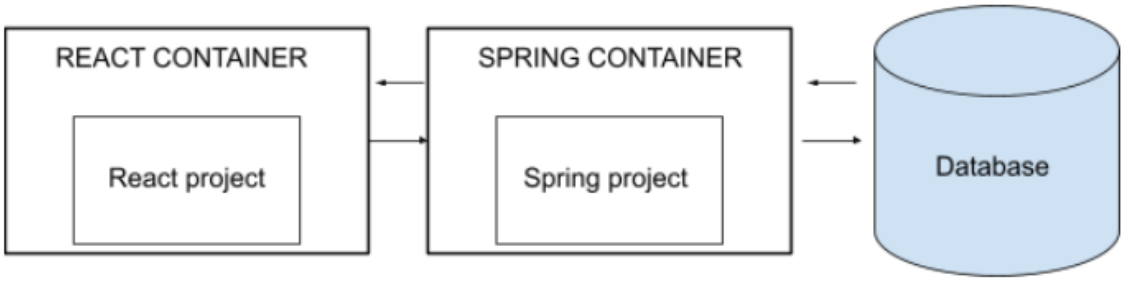
Arhitectura : Layered

3. DATABASE

Baza de date pe care o folosesc este postgres. In baza de date sd avem tabelele : user si device



DEPLOY DIAGRAM



ASSIGNMENT 2

1. REACT APP

In plus fata de primul assignment, am adaugat partea de uses. Fiecare user se va loga folosind credentialele create de admin, iar la intrarea in aplicatie va observa graficele cu device-urile sale (consumption-oy si timestamp-ox)

2. SPRING APP

La partea de backend in Spring am adaugat un consumer. M-am folosit de rabbitMQ si am citit din coada generata de catre producer. Se conecteaza aplicatia la coada si se citeste. Datele primite se parseaza si se introduc in baza de date.

3. SIMULATOR QUEUE

La partea de backend legata de queue simulator dupa conectarea la RabbitMQ, se seteaza un interval de timp la care sa se trimita un mesaj. Mesajul este trimis upa un sablon setat, in forma de json. Se trimit deviceId(harcodat in apllicatie), timestamp (data curenta la care s-a trimis mesjul) si measurements (o marime citita din sensor.csv).

4. DATABASE

Baza de date pe care o folosesc este postgres. In baza de date, in plus fara de assgn 1 am adaugat tabela de measurements :

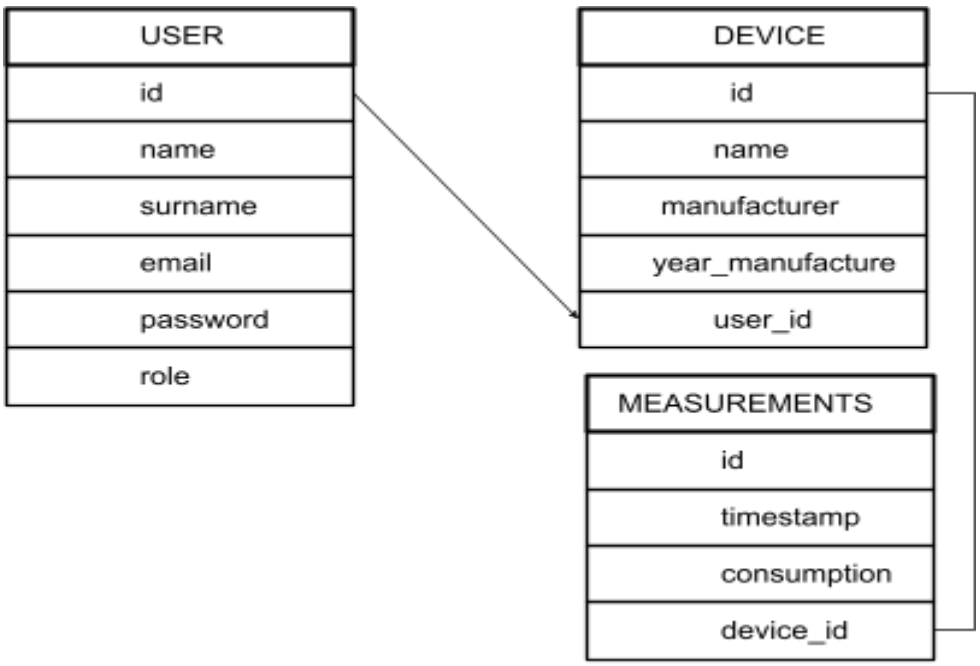
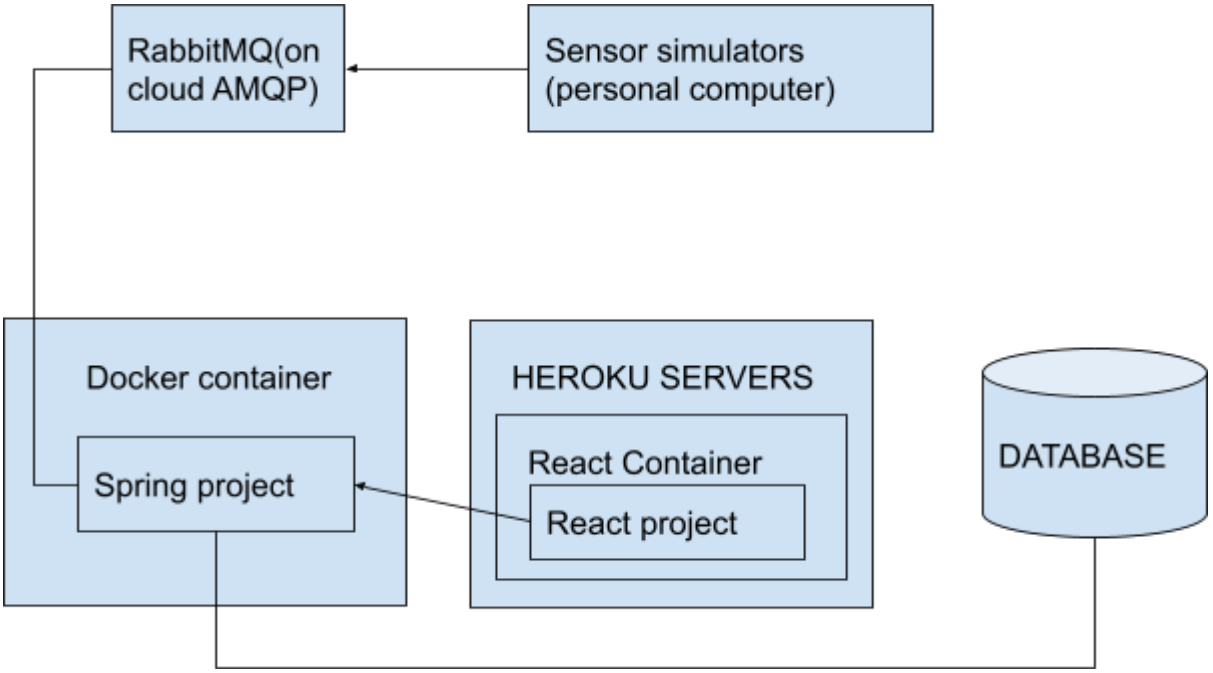


DIAGRAMA DE DEPLOY



ASSIGNMENT 3

1. REACT APP

In plus fata de primele 2 assignmenturi am adaugat o noua functionalitate, anume un chat. Chatul este facut cu ajutorul bibliotecii grpc. Screenul de chat este compus din 2 parti, in partea stanga se afla lista de useri, iar in partea dreapta este chatul cu userul pe care l-am selectat din lista. Administratorul poate sa vorbeasca cu toti userii, iar userii pot vorbi doar cu administratrul.

2. SPRING APP

La partea de Spring totul a ramas neschimbat

2.1. NODE APP

Am adaugat inca un nou backend in node js in care am facut un server care se foloseste de grpc pentru a procesa datele primite de pe frontend. Acesta contine un fisier de chat.proto in care se afla “interfetele” serviciilor pe care le folosesc, annume sendMessage si receiveMessage, si interfata mesajului care contine from (persoana care a trimis mesajul), to (persoana care a primit mesajul) si msg (mesajul in sine). SendMessage si receiveMessage sunt implementate in fisierul principal, server.js in care se si porneste serverul. Practic userii se conecteaza la server si de fiecare data cand se trimite un mesaj, toti userii vor primi acel

mesaj, insa din partea de frontend este restrictionat sa se proceseze doar mesajele care sunt destinate userului in cauza.

4. DATABASE

Baza de date ramane aceeasi ca si la celelalte 2 assignmenturi.

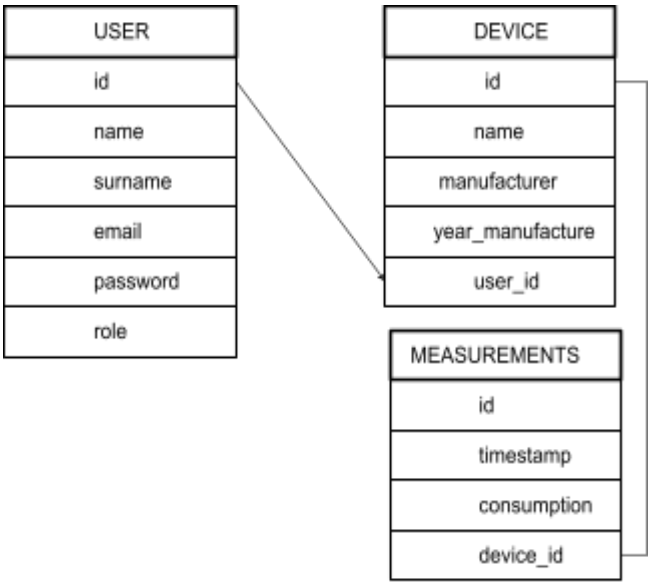


DIAGRAMA DE DEPLOY