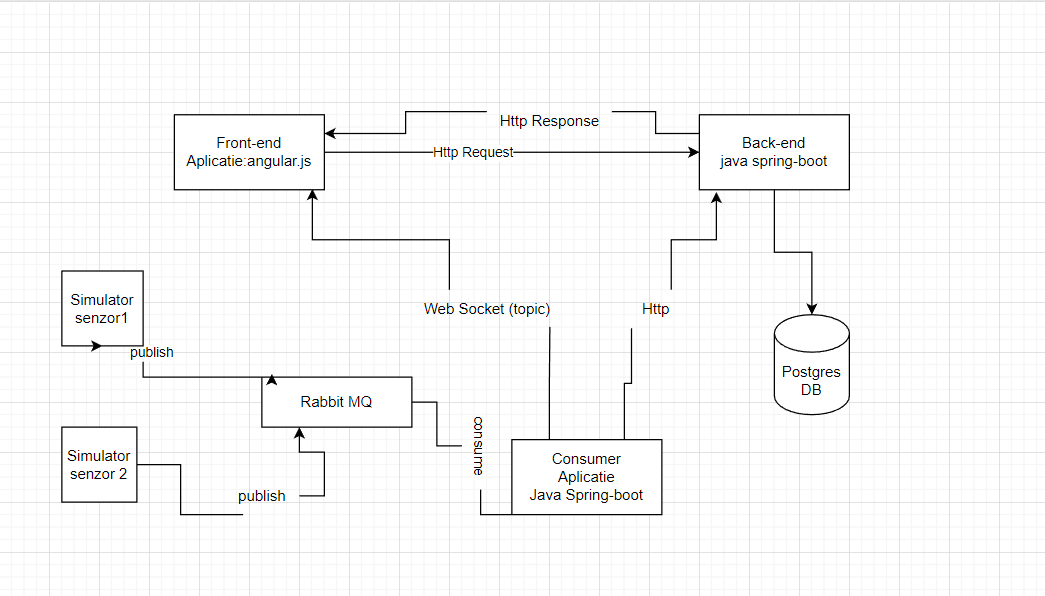
Assigment 2. Sisteme distribuite

Andercou Alexandru

1.Conceptual architecture

Proiectul este format din 6 module : un front end care are pagini pentru functionalitati de administrator si de user, un server web aplicatie java spring boot ,o baza de date posstgres, un simulator de senzor proiect java maven, care genereaza mesaje , o coada RabbitMQ care primeste mesaje de la senzor si un consumer aplicatie java spring boot care citeste mesajele din coada le preproceseaza , le salveaza in baza de date apeland back-endjul si comunica cu interfata web prin Web socketuri.

 Fig1.Conceprual Architecture

Front-endul este format dintr-o aplicatie web scrisa in angular.js si html care ofera utilizatorului obisnuit posibilitatea sa vizualizeze cantitatea de energie consumata de deviceurile asignate acestuia , intr-un grafic care reprezinta energia consumata in functie de timp pe parcursul unei zile alese si sa vada notificari referitoare la intrecerea limitei de energie. Administratorul aplicatiei are acces la toate datele din aplicatie : utilizatori,deviceuri, perechi deviceuri –utilizatori si masuratori pentru care are capacitati de crud. Frontendul comunica cu backendul prin mesaje din protocolul HTTP de get si post.Front-endul comunica cu consumerul folosind web-sockets , in care front-endul are rol de client.

Back-endul are accesul la baza de date postgres si implementeaza toate functionalitatiile de CRUD peste datele aplicatiei si are si functionalitatea de login. Acesta implementeaza logica de business si ofera toate functionalitatiile afisate in interfata grafica, front-end.

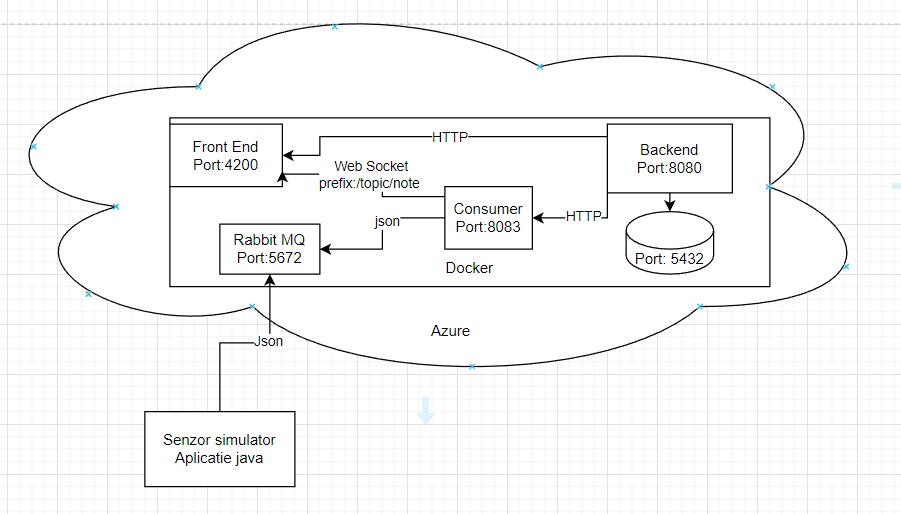
Simulatorul de senzor reprezinta un device inregistrat in baza de date. Simulatorul foloseste id-ul unui device existent preluat dintr-un fisier de configurare si simuleaza functionalitatea unui device prin citirea informatiilor dintr-un fisier CSV(sensor.csv) si prin trimiterea de mesaje JSON prin coada RabbitMQ ,acesta trimite id-ul acestuia, valoarea numerica al energiei consumate si un timestamp generat local. Aplicatia va rula in mai multe instante ale acesteia cu deviceuri diferite.Senzorul geenreaza mesaje si le trimite o data la 10 minute.(simulat la 2 minute)

Coada RabbitMQ este un server cu rolul de message broker ,acesta pastreaza mesajele trimise de senzor si care vor fi preluate de consumer. Dupa ce mesajele sunt cionsumate acestea dispar din coada. Coada Rabbit MQ cu numele Sensor\_sends

Consumerul este o aplicatie java spring-boot care prelueaza informatiile din coada RabbitMQ , proceseaza mesajele pentrtu a calcula energia totala pentru fiecare ora( simulat la 12 minute, deoarece mesajele sunt generate de senzor o data la 2 minute si 6 mesaje formeaza o ora) , informeaza interfata web daca energia a intrecut limita device-ului precum este salvata in baza de date , folosind un Web socket de tip server si salveaza masuratoriile in baza de date trimitand mesaje din protocolul http la back-endul aplicatiei.

Web socketul creaza o cale full-duplex intre Consumer si Interfata web. Consumerul foloseste java springboot-websocket.

2. Deployment

 Fig2. Deployment diagram

Ftont-endul se afla debloyed on docker (on azure) si ruleaza pe portul 4200 ,backendul este de asemenea deployed pe docker(and azure) serverul web asculta pe portul 8080 si comunica cu baza de date postgres care asculta pe portul 5432 . Back-endul foloseste baza de date energy-db ,cu utilizatorulk postgres si parola:1234. RabbitMQ este deployded ca imagine pe docker( on azure) si asculta pe portul 5672. Consumerul se afla deployed pe docker(on azure) si ruleaza pe portul 8083 dar comunica cu coada rabbitMQ pe portul 5672.

Spre deosebire de celalate componente ale aplicatiei Senzorul nu este o aplicatie spring boot ci numai java si nu ruleaza sau asculta pe un port ,aceasta este o aplicatie stand alone si se afla local pe calculator dar comunica cu coada RabbitMQ pe portul 5672 care este deployed on azure.[[1]](#endnote-4664)

Web Socketurl java-spring server are enabled Simple Broker cu prefixul “/topic/note/user\_id” si prefixul de destinatie “.gkz”. Acesta este conectat cu instantele de interfata web cu prefixul “gks-stomp-endpoint" si are allowed originea : http:/localhost:4200.

In interfata web clientul stomp este abonat pe calea cu prefixul “topic\note/user\_id” acesta este creat in serviceul din aplicatia angular WebSocketService si injectat din root in componenta utilizator.

1. [↑](#endnote-ref-4664)