Energy Management System

Assignment 2

Student: Balea Cristian Constantin

Prof. coordinator: Marcel Antal

Grupa: 30644

# Introducere

Scopul assignmentului constă în creearea unui serviciu de monitorizare a consumului. Acesta primește într-o coadă de mesaje, măsurători primite de la dispozitivele de măsurare a consumului.

În cazul în care se depășește consumul maxim admis pentru un dispozitiv, se notifică utilizatorul. De asemenea, acesta poate vedea consumul înregistrat de dispozitive în fiecare zi, sub formă de grafic.

# Cerințe funcționale și non-funcționale

* Implementarea serviciului de monitorizare
* Implementarea producerului și consumerului de mesaje
* Adaptarea interfeței utilizator pentru afișarea consumului
* Implementarea sistemului de notificări

# Tehnologii utilizate

S-a creat un nou microserviciu, ems-monitor, care stocheaza dispozitive și măsurători, și un simulator pentru măsurători, ems-simulator, care va fi rulat din linia de comanda, cu id-ul dispozitivului pentru care vor să se transmită măsurătorile. Pentru ambele s-au folosit Java și Spring.

Ca message broker, s-a folosit RabbitMQ, cu două cozi de mesaje – una pentru măsurători, și cealaltă pentru modificări ale dispozitivelor sau adăugarea lor.

Pentru frontend, s-a folosit librăria ChartJS.

Pentru notificări, s-a folosit implementarea Spring a Websocketurilor, si librăria StompJS pentru frontend.

# Arhitectura conceptuală a sistemului

Ems-Monitor:

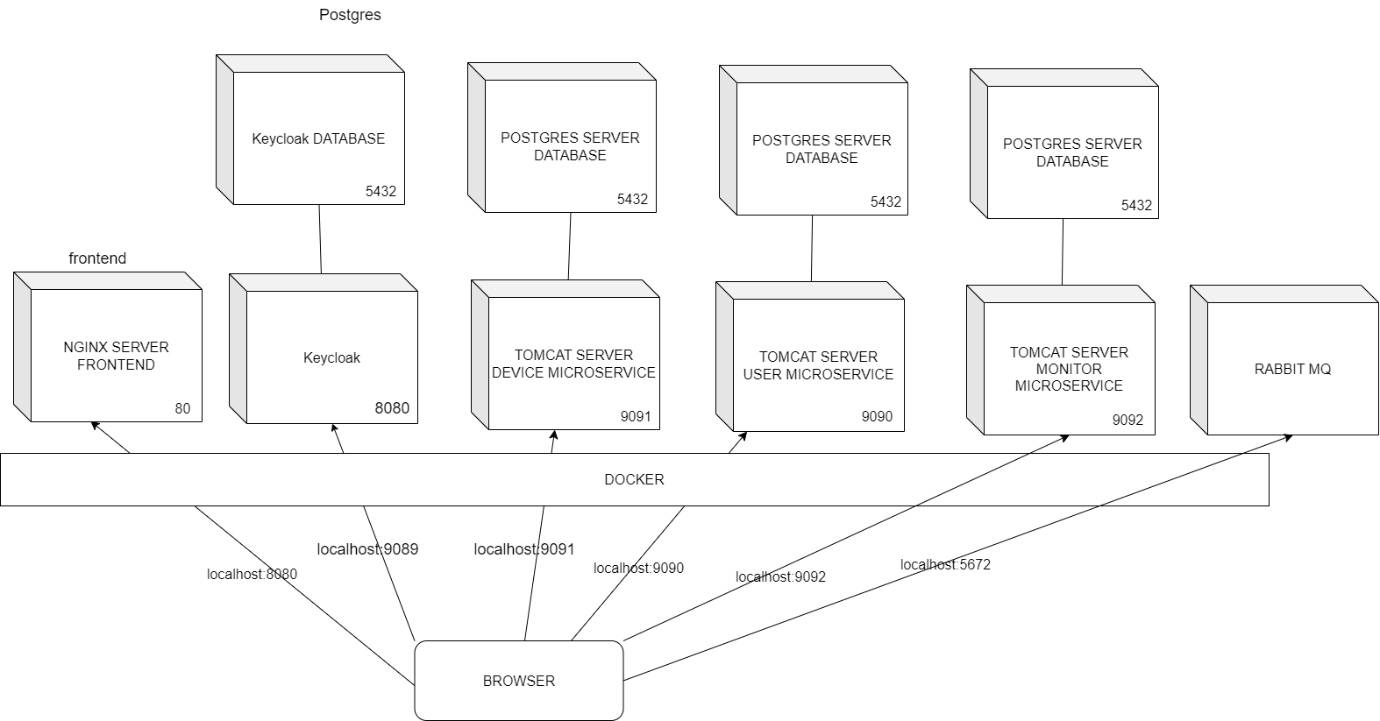
* Folosește o bază de date postgres, pentru stocarea informațiile despre dispozitive, și pentru stocarea măsurătorilor.
* Folosește 2 cozi de mesaje – pentru primirea măsurătorilor, și pentru modificări asupra dispozitivelor – un producer s-a implementat in Ems-Devices, care trimite mesaje când se produc operații asupra dispozitivelor.
* Pune la dispoziție un endpoint, care returnează măsurătorile dintr-o zi, pentru un anumit dispozitiv.
* Structura microserviciului este identică cu cea a microserviciilor din tema anterioară.

Ems-Simulator:

* Aplicație rulată în afara dockerului. Se rulează din linia de comandă, obligatoriu cu parametrul device.id, ID-ul device-ului pentru care se vor trimite măsurători.
* Implementează un producer de mesaje, ce produce măsurători, citite din sensor.csv

RabbitMQ – broker de mesaje, conține 2 cozi, una pentru modificări asupra dispozitivelor, una pentru măsurători.

# Diagrama de deployment

Se va adăuga diagramei din tema anterioară, RabbitMQ, respectiv serviciul de Monitor și baza de date a acestuia.

# Instrucțiuni de instalare

Se vor executa comenzile următoare:

În folderele rădăcină ale microserviciilor:

./gradlew clean build

docker build -t [devices | users | monitor] .

În folderul rădăcină al proiectului de frontend:

ng build

docker build -t frontend.

În folderul cu docker-compose.yml

docker-compose up

**Opțional**:

* deoarece serverul nginx are o problemă la reîncărcarea paginilor, se va intra în containerul de frontend, în **/etc/nginx**, și în **default.conf** se va adăuga la **location**, folosind vim sau un alt tool asemănător, linia try\_files $uri /index.html, iar apoi se va reporni containerul.
* În cazul în care RabbitMQ nu creează cozile de mesaje – se va intra în interiorul lui, se va activa managementul folosind rabbitmq-plugins enable rabbitmq\_management și apoi, de pe localhost:15672, se vor adăuga queue-urile: device, simulator, se vor crea exchange-urile device\_exchange și simulator\_exchange iar apoi se va face bindingul folosindu-se device\_routingKey și simulator\_routingKey.