Energy Management System

Assignment 1

Student: Balea Cristian Constantin

Prof. coordinator: Marcel Antal

Grupa: 30644

# Introducere

Scopul assignmentului constă în creearea unei platforme Web cu scopul de a facilita managementul consumului de energie, prin instalarea unor dispozitive inteligente de măsurare a consumului de energie.

Există 2 tipuri de utilizatori în cadrul aplicației, clienții și adminii. Adminul poate să realizeze operații de tip CRUD asupra utilizatorilor și a device-urilor, și poate realiza maparea dintre aceștia. Clienții își pot vedea dispozitivele asociate.

# Cerințe funcționale și non-funcționale

Cerințe funcționale pentru rolul de administrator:

* Operații CRUD pe useri
* Operații CRUD pe dispozitive
* Mapare user-dispozitiv

Cerințe funcționale pentru rolul de user:

* Vizualizarea dispozitivelor associate

Cerințe non-funcționale:

* Împărțirea pe microservicii
* Securitate: autentificare pentru a restricționa accesul userilor pe anumite pagini

# Tehnologii utilizate

Pentru frontend s-a folosit Angular cu TypeScript, iar pentru backend s-a folosit Java 17 cu framework-ul Spring. Pentru managementul dependințelor s-a folosit gradle.

S-a folosit Keycloak, atât pe frontend, cât și pe backend, pentru a gestiona userii și rolurile lor, logarea, sesiunile, delogarea, și accesul pe pagini.

Pentru stocarea datelor s-au folosit baze de date relaționale PostgreSQL.

S-a folosit Flyway pentru migrări pe bazele de date.

Pentru deployment s-a folosit Docker, construindu-se containere pentru fiecare serviciu în parte.

# Arhitectura conceptuală a sistemului

Sistemul conține 3 baze de date de tip PostgreSQL, una pentru utilizatori, una pentru dispozitive și una pentru keycloak. Acestea rulează în containere și sunt conectate la rețeaua Docker „backend”.

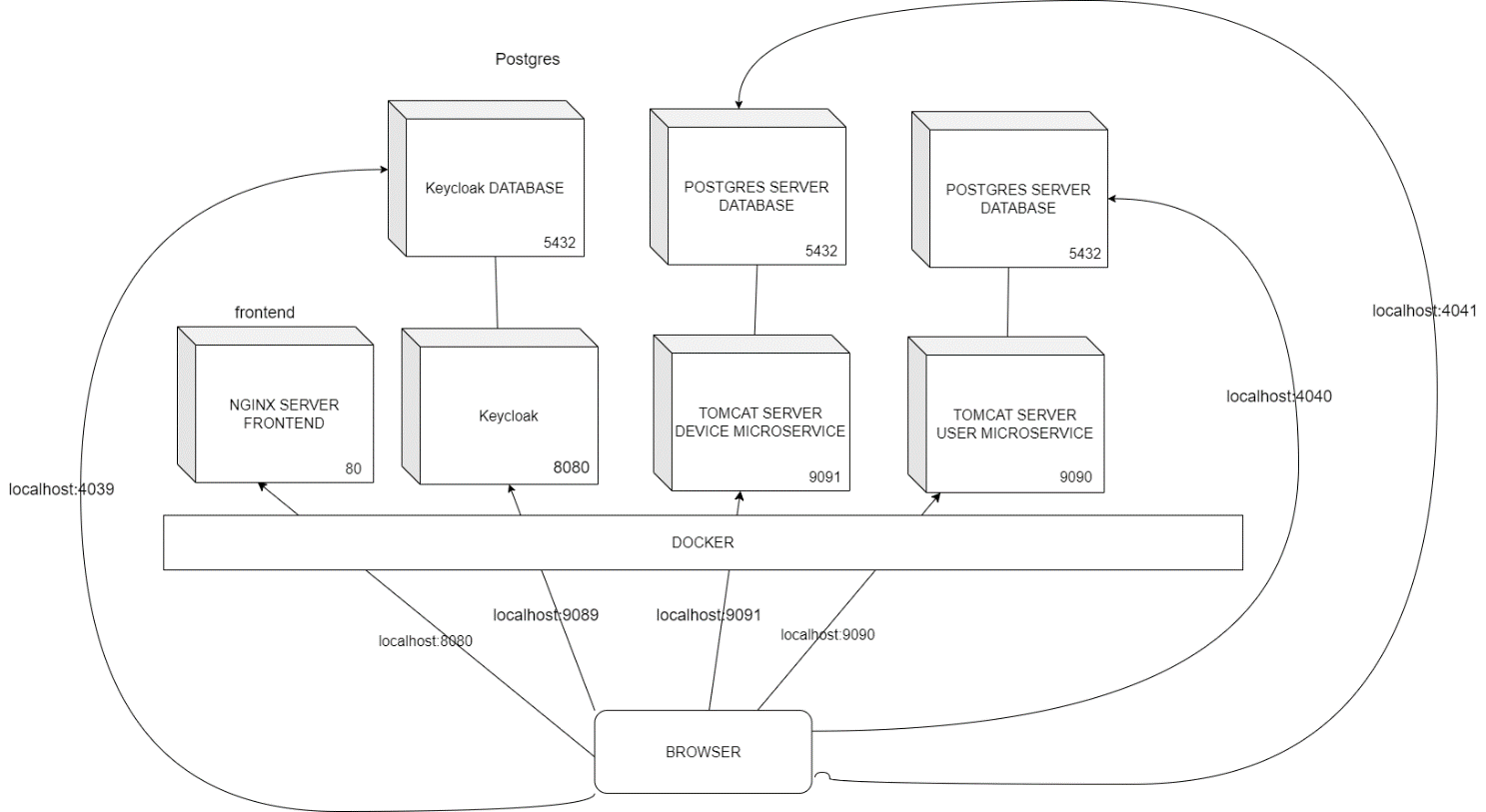
Microserviciile:

* Keycloak – serviciu de autentificare și autorizare open-source, dezvoltat de Red Hat. Acesta oferă diverse servicii de securitate, precum autentificare, autorizare, administrarea sesiunilor, autentificare sociala. Acesta conține un realm, ems (Energy Management System), care la rândul lui conține clienții – celelalte microservicii, userii, roluri, sesiuni șamd. Keycloak are la bază Oauth2 pentru autorizare și OpenID Connect pentru autentificare. Fiecare microserviciu se identifică cu un clientId și un client secret.
* Microserviciul Users – este format din 2 module, app și api. Modulul api conține interfața pentru API-ul aplicației, și DTO-urile folosite în comunicarea cu exteriorul. Modulul app este bazat pe o arhitectură de tip layered. De jos în sus, avem următoarele:
  + Repository-urile, extind JpaRepository, sunt folosite pentru conectarea la baza de date, și pentru operații de persistare și interogare a datelor. Baza de date conține o tabelă pentru persistarea entității de User.
  + Service-urile sunt următorul layer și conțin logica de business. Se folosește MapStruct pentru conversia dintre DTO-uri și entități, și WebClient pentru comunicarea externă, cu serviciile Keycloak și Devices. Am folosit WebClientul pentru sincronizarea dintre cele 2 baze de date, care se realizează la modificări precum update sau delete pe entitatea de User.
  + Controllerele – conțin endpointurile, metodele utilizate pentru comunicarea cu exteriorul, prin requesturi de tip GET, PATCH, DELETE și POST. De asemenea, se folosește un ExceptionController, pentru a gestiona eventualele excepții din interiorul programului, și transmiterea lor către exterior.
* Microserviciul Devices – structură foarte asemănătoare cu cea a microserviciului de Users. Diferența o face faptul că se persistă date și despre utilizatori, precum nume, prenume, e-mail, pentru accesul mai rapid la aceste informații.
* Frontendul – am ales Angular deoarece structura este una mai restrictivă decât a React-ului. Am structurat proiectul pe pachete, core-ul, care conține headerul, footerul, gărzi și interceptori, și feature-urile, care mapează cerințele proiectului. Avem mai multe module, pentru pagina de admin, de utilizator, si pentru pagina principală. Fiecare conține la rândul ei diferite componente. Pentru login și logout m-am folosit de Keycloak, și de asemenea pentru accesul pe pagini, având un RoleGuard care blocheaza spre exemplu, accesul unui utilizator obișnuit, pe pagina destinata administratorului.

# Diagrama de deployment

Am folosit Dockerfile-uri, docker build și docker compose pentru crearea și managementul containerelor. Sistemul nostru este format din 7 containere care comunică într-un network, după cum urmează:

* Frontendul – mapat la portul 8080, rulează pe un server NGINX
* Serviciul de dispozitive – mapat la portul 9091, rulează pe un server Tomcat.
* Baza de date pentru dispozitive – mapată la portul 4041 – PostgreSQL.
* Serviciul de utilizatori – mapat la portul 9090, rulează pe Tomcat.
* Baza de date pentru utilizatori – mapată la portul 404 – PostgreSQL.
* Serviciul Keycloak – mapat la portul 9089, folosește imaginea quay.io
* Baza de date pentru Keycloak – mapată la portul 4039 – PostgreSQL. Am ales să folosesc una externă în detrimentul celei interne implicită pentru un acces mai facil la conținutul ei.



# Instrucțiuni de instalare

Se vor executa comenzile următoare:

În folderul rădăcină al proiectului ems-users:

./gradlew clean build

docker build -t users .

În folderul rădăcină al proiectului ems-devices:

./gradlew clean build

docker build -t devices .

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.