UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Sisteme distribuite

Assignment 3

Nume: Huza Lisandra

Grupa: 30643

Cuprins

1. Introducere

Proiectul „Sistem de Management al Energiei” este dezvoltat pentru a facilita monitorizarea și gestionarea utilizatorilor și dispozitivelor printr-o aplicație bazată pe microservicii. Sistemul oferă două tipuri de conturi de utilizatori: administratori și clienți. Administratorii pot gestiona conturile și dispozitivele, în timp ce clienții pot vizualiza datele despre dispozitivele lor. Device-urile sunt simulate prin intermediul unui producer care trimite date sub forma de json. Administratorii și utilizatori pot comunica între ei, iar administratorii pot trimite mesaje mai multor utilizatori prin intermediul unor grupuri.

2. Cerințe funcționale

- Autentificare utilizatori: Utilizatorii trebuie să se autentifice pentru a accesa aplicația. Redirecționarea către pagini este realizată în funcție de rolul lor.
- Administrator: Operații CRUD (Creare, Citire, Actualizare, Ștergere) pentru gestionarea conturilor de utilizatori și a dispozitivelor, cât și gestionarea asocierii dispozitivelor clienților.
- Client: Vizualizarea dispozitivelor asociate pe pagina personală.
- Restricționarea accesului doar la paginile asociate rolului, fără permisiunea de a accesa celelalte pagini.
- Middleware-ul orientat pe mesaje permite Simulatorului Dispozitivului de Măsurare Inteligentă să trimită tupluri de date într-un format JSON.
- Componenta consumatorului de mesaje a microserviciului procesează fiecare mesaj și notifică aplicația client în mod asincron folosind WebSocket.
- Valorile de energie orară vor fi salvate de componenta consumatorului în baza de date de Monitorizare.
- Aplicația front-end afișează o casetă de chat în care utilizatorii pot introduce mesaje.
- Mesajul este trimis asincron către administrator, care primește mesajul împreună cu identificatorul de utilizator, putând începe un chat cu utilizatorul.
- Mesajele pot fi trimise înainte și înapoi între utilizator și administrator în timpul unui chat sesiune.
- Administratorul poate discuta cu mai mulți utilizatori simultan.
- O notificare este afișată pentru utilizator când celălalt administrator citește mesajul și viceversa.
- Se afișează o notificare pentru utilizator (de exemplu, tastând) în timp ce administratorul de la celălalt sfârșitul tipurilor de comunicare ale mesajului său și invers.

3. Arhitectura Conceptuală a Sistemului Distribuit

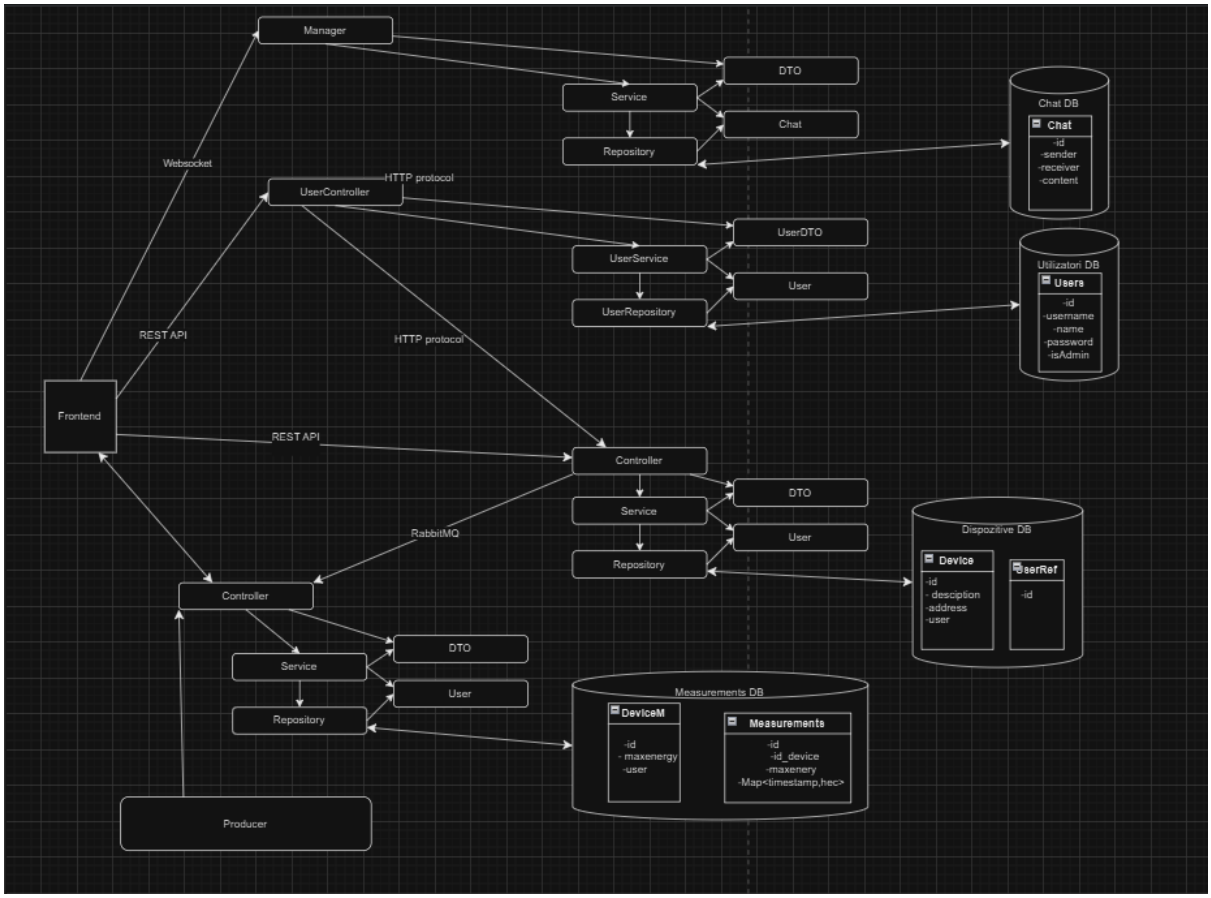
Arhitectura sistemului distribuit se bazează pe o structură de microservicii, fiecare microserviciu având o funcționalitate specifică. Sistemul utilizează un frontend pentru interacțiunea cu utilizatorul și două microservicii REST pentru gestionarea utilizatorilor și a dispozitivelor.

Arhitectura este împărțită astfel:

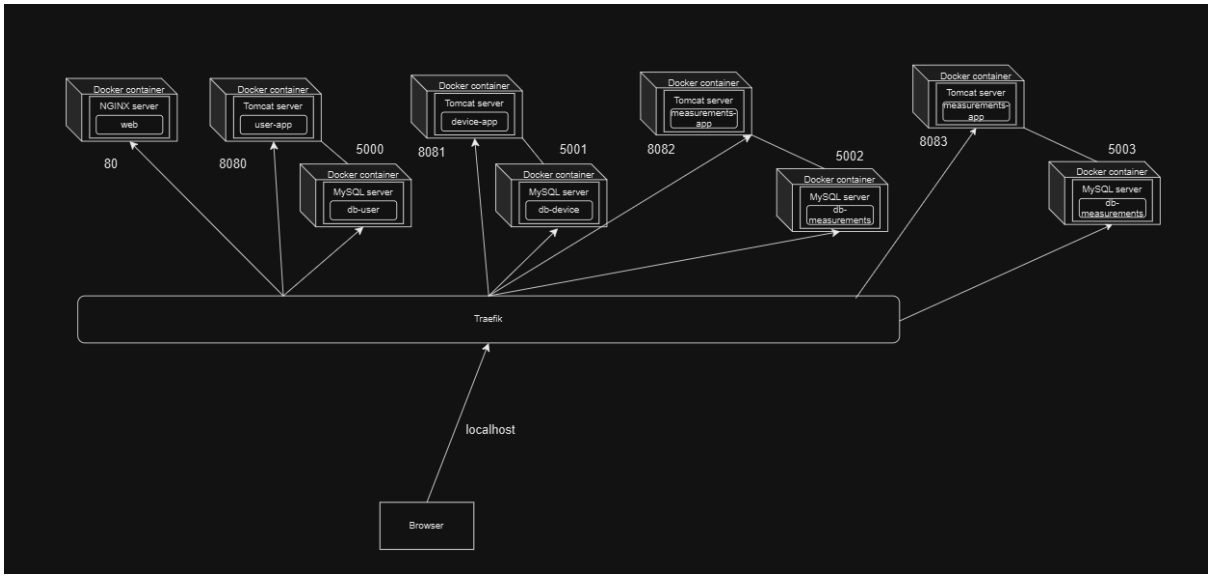
- Frontend – aplicație web, implementată cu un framework JavaScript modern, React
  - Oferă o interfață grafică interactivă pentru fiecare tip de utilizator în parte, precedată de o autentificare care asigură redirecționarea aferentă
  - Comunicarea cu backend-ul se realizează prin apeluri de API REST
- Backend – format din 2 microservicii care se ocupă de logica, procesele li gestionarea datelor
  - Microserviciul de Management al Utilizatorilor - MMU
  - Acest microserviciu este responsabil pentru gestionarea conturilor de utilizatori
  - CRUD pentru utilizatori
  - Autentificare – validează identitatea utilizatorilor, restituind rolul acestora
  - Sincronizarea cu Microserviciul de Management al Dispozitivelor – realizat prin trimiterea unei cereri de creare/ stergere a unei referințe de utilizator
    - Microserviciul de Management al Dispozitivelor - MMD
  - Acest microserviciu este responsabil pentru gestionarea dispozitivelor

- CRUD pentru dispozitive
- Asocierea dispozitivelor cu utilizatorii – pe baza referințelor de utilizatori transmise de către MMU se poate crea o asociere între aceștia si dispozitive
    - Microserviciul de Monitorizare si Comunicare - MMC
- Acest microserviciu implementează un sistem de monitorizare și comunicare pentru gestionarea energiei.
- Calculează consumul orar de energie , la fiecare ora acesta este stocat, iar atunci când consumul maxim este depasit trimite un warning frontend-ului print-un webSocket
- Consistența acestuia este menținută prin stocarea datelor importante despre dispozitive, aceste date sunt primite prin intermediul unui rabbitmq
- Datele despre consumul dispozitivelor este primit tot prin intermediul unui rabbitmq
    - Simuator pentru device-uri
- Pentru a simula consumul de energie am citit datele dintr-un fisier csv
- Trimiterea datelor se face la fiecare 5 secunde, iar calculul pentru ora se face la un minut pentru a se putea testa mai usor
    - Microserviciul de chat
- Acest microserviciu implementeaza o logică de primire si trimitere a mesajelor între utilizatori și administratori
- Se stochează in baza de date toate mesajele, până în momentul în care acestea sunt citite
- Convorbirea cu front-end ul se realizează prin intermediul websocket-urilor deschise pentru fiecare ubilizator

- Baza de date
    - Fiecare microserviciul dispune de propria bază de date relațională pentru stocarea datelor, permițând o separare clară a responsabilităților
    - Baza de date pentru MMU conține tabela
        - users – id, username, name, password, isAdmin
    - Baza de date pentru MMD conține 2 tabele pentru
        - device -id,description, address, user
        - userReference -id
    - Baza de date pentru MMC conține 2 tabele pentru
        - device -id,mhec, user_id
        - measurements – id, id_device,mhec, Map<timestamp,hec>
    - Baza de date pentru MC conține 1 tabelă pentru
        - chat -id,sender,receiver,content
- Fluxul de date și comunicare
    - Frontend-ul comunica cu microserviciile MMU , MMD prin apeluri Rest, iar cu MMC prin webSocket
    - Microserviciile folosesc protocoale standard HTTP pentru transferarea datelor (referența pentru utilizatori) și rabbitmq pentru trimiterea datelor spre MMC
- Securitatea
    - Securitatea este asigurată folosind jwt
    - La logarea unui utilizator se creează cu ajutorul unei chei secrete un tocken care este trimit front-end -ului
    - La fiecare cerere din front către back acest tocken este trimis în header, fiind apoi verificat în backend
    - Daca acest tocken lipsește sau nu este conform, cererea va fi respinsă



4. UML Deployment diagram



5. Readme file

- Dacă se dorește rularea simplă a aplicației se ruleaza cele 4 aplicații de backend, după care se scrie în linia de comandă a aplicației de frontend npm start
- Dacă se dorește rularea aplicației folosind docker

- Se deschide un terminal in folder-ul care contine docker-compose
- Se ruleaza docker-compose build
- Se ruleaza docker-compose up -d
- Se poate verifica starea containerelor docker-compose ps
- Pentru vizualizare log-uri docker-compose logs -f
- Se acceseaza aplicata la http://localhost