SISTEME DISTRIBUITE

Assignment 1

Documentație

Anton Camelia

Grupa: 30241

1. **Arhitectura Conceptuală a Sistemului Distribuit**

Proiectul utilizează o arhitectură de tip microservicii, creând un sistem distribuit format dintr-o aplicație client și două microservicii, fiecare responsabil pentru gestionarea unor resurse specifice:

Aplicația client este o interfață web dezvoltată în React, care oferă funcționalități precum autentificare și redirecționare automată pe baza rolului utilizatorului. În funcție de rol, utilizatorii sunt redirecționați către pagini dedicate:

* Utilizatorii obișnuiți au acces la o pagină unde pot vizualiza și gestiona propriile dispozitive
* Administratorii au acces extins la toate resursele din sistem, putând vizualiza și administra dispozitivele și profilurile tuturor utilizatorilor.

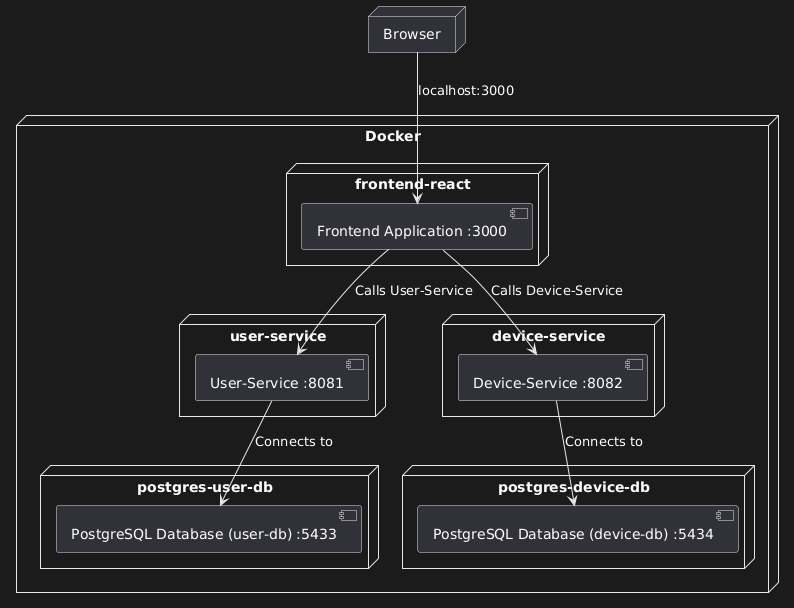
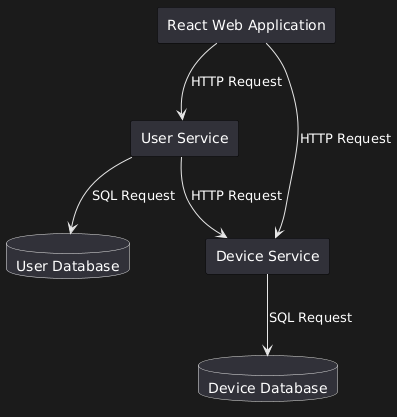
Microserviciile sunt dezvoltate în Java Spring și funcționează ca API-uri REST, fiecare având propria bază de date dedicată pentru gestionarea utilizatorilor și dispozitivelor. API-ul destinat utilizatorilor include și funcționalități de autentificare și înregistrare bazate pe token-uri pentru securitatea datelor.

Comunicarea între componente se realizează printr-un model de cereri și răspunsuri HTTP, microserviciile expunând endpoint-uri REST accesibile prin cereri HTTP de la client. Pentru a menține coerența datelor între cele două microservicii, serviciul de utilizatori inițiază cereri către serviciul de dispozitive în momente cheie, precum adăugarea sau ștergerea unui utilizator. Această sincronizare asigură că baza de date a dispozitivelor conține referințe corecte la utilizatorii existenți. Dacă o operațiune de adăugare a unui utilizator eșuează în serviciul de dispozitive, baza de date a utilizatorilor rămâne nemodificată, asigurând astfel coerența globală a datelor și prevenind inconsecvențele. Această arhitectură distribuită și modulară maximizează scalabilitatea sistemului, fiind potrivită pentru un mediu de producție distribuit.

**2. Implementare (Deployment Diagram)**

Pentru implementarea aplicației, am folosit Docker pentru a crea containere separate pentru aplicația web, servicii și baze de date. Fiecare componentă este distribuită într-un container individual, cu porturi expuse pentru a permite atât comunicarea între ele, cât și accesul extern, după cum urmează:

* **user-service**: Conține serviciul backend pentru gestionarea utilizatorilor, implementat în Spring Boot, și este expus pe portul 8081.
* **device-service**: Conține serviciul backend pentru gestionarea dispozitivelor, de asemenea implementat în Spring Boot, și este expus pe portul 8082.
* **user-db** și **device-db**: Conțin bazele de date PostgressSQL pentru stocarea datelor specifice fiecărui serviciu, expuse pe porturile 5433 și, respectiv, 5434.
* **my-app**: Conține aplicația web frontend realizată în React, expusă pe portul 3000 pentru accesul utilizatorilor.



**3. Instrucțiuni de rulare a aplicației**

a) Clonați repository-ul pe calculatorul dvs.

b) În proiectele User și Device, deschideți fișierul application.properties și actualizați valorile database.user și database.password cu credențialele dvs. pentru utilizatorul PostgreSQL.

c) În PostgreSQL, creați două baze de date noi, numite user-db și device-db.

d) Pentru a construi și lansa containerele Docker, executați comanda docker-compose up --build din directorul principal al proiectului. Aceasta va construi și va porni automat containerele Docker pentru toate componentele aplicației.