SISTEME DISTRIBUITE

Assignment 2

Documentație

Anton Camelia

Grupa: 30241

1. Arhitectura Conceptuală a Sistemului Distribuit

Sistemul distribuit utilizează o arhitectură bazată pe microservicii, unde fiecare componentă este responsabilă pentru o funcționalitate specifică și poate fi scalată independent. Comunicarea între microservicii este realizată prin intermediul API-urilor REST sau folosind un broker de mesaje (RabbitMQ).

Frontend-ul este construit ca o aplicație React care servește drept interfață cu utilizatorul, conectându-se la microservicii prin intermediul unui reverse proxy (Traefik).

Reverse proxy-ul Traefik gestionează rutarea și balansarea traficului între microservicii, asigurând un punct centralizat de acces.

Microserviciile sunt împărțite pe trei domenii:

* User Management: Gestionează datele utilizatorilor, inclusiv autentificarea și autorizarea, și comunică cu baza de date user-db.
* Device Management: Se ocupă de administrarea dispozitivelor conectate și stochează informațiile acestora în baza de date device-db.
* Monitoring Management: Primește date de la simulatoarele de senzori prin RabbitMQ și interoghează microserviciul de gestionare a dispozitivelor pentru date suplimentare folosind un API REST.

Brokerul de mesaje RabbitMQ facilitează comunicarea asincronă între simulatoarele de senzori și serviciul de monitorizare, decuplând astfel componentele pentru a crește reziliența sistemului. Baze de date distincte PostgreSQL sunt utilizate pentru stocarea datelor utilizatorilor, dispozitivelor și celor de monitorizare.

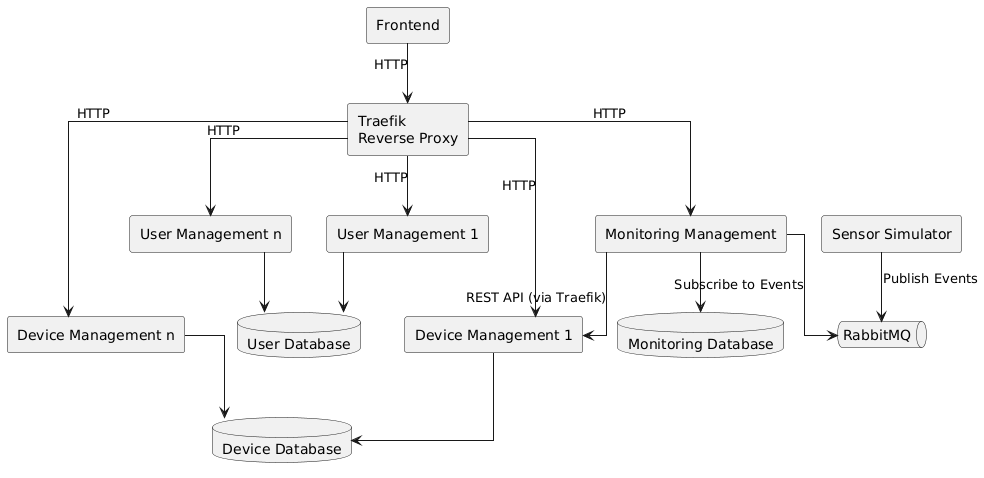
1. Implementare (Deployment Diagram)

Implementarea sistemului este descrisă printr-o diagramă de deployment care prezintă relațiile și conexiunile dintre componente.

Frontend-ul comunică prin HTTP cu Traefik, care rutează cererile către microserviciile backend. Fiecare microserviciu comunică cu baza sa de date specifică, iar Monitoring Management are o dublă responsabilitate:

* + Primește evenimente de la simulatoarele de senzori prin RabbitMQ.
  + Se conectează la Device Management printr-un REST API pentru a colecta informații suplimentare.

De asemenea, RabbitMQ acționează ca un punct central pentru procesarea mesajelor asincrone trimise de simulatoare și preluate de serviciul de monitorizare.



1. Instrucțiuni de Rulare a Aplicației
2. Clonați repository-ul pe calculatorul dvs.
3. În proiectele User, Monitoring și Device, deschideți fișierul application.properties și actualizați valorile database.user și database.password cu credențialele dvs. pentru utilizatorul PostgreSQL.
4. În PostgreSQL, creați două baze de date noi, numite user-db și device-db, monitoring-db.
5. Lansarea containerelor Docker. După configurarea bazelor de date, accesați directorul principal al proiectului și executați comanda docker-compose up --build pentru a construi imaginile Docker și a porni containerele aplicației, inclusiv pentru microservicii, RabbitMQ, baze de date și frontend.
6. Pentru a primi date în timp real de la simulatoare, trebuie să porniți scriptul Python responsabil pentru generarea și trimiterea datelor către RabbitMQ. Acesta transmite datele către microserviciul de monitorizare, care le procesează în timp real.
7. Verificați funcționarea componentelor accesând frontend-ul la http://localhost:3000, dashboard-ul Traefik la http://localhost:8080 și RabbitMQ Management Console la http://localhost:15672 (username: guest, password: guest).
8. Pentru a opri aplicația, utilizați docker-compose down, iar pentru a șterge complet datele, folosiți docker-compose down -v. Dacă apar probleme, analizați jurnalele serviciilor cu comanda docker-compose logs <service-name>.

Script sql insert admin + adaugat in docker-compose

De modificat in backend la cors

Deploy modificat add web socket