Energy Management System

Documentație

Anton Camelia

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.
* Chat Management: Permite comunicarea între utilizatori și administratori în timp real printr-o conexiune WebSocket. Integrarea se realizează cu ajutorul unei noi baze de date chat-db, iar notificările sunt gestionate asincron. Toate comunicațiile sunt securizate prin SSL.

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/HTTPS cu Traefik, care acționează ca un reverse proxy și rutează cererile către microserviciile backend. Acest lucru asigură o distribuție corectă a traficului, securitate prin TLS și posibilitatea de a expune diferite microservicii folosind domenii sau reguli specifice. Fiecare microserviciu este implementat independent și comunică cu propria bază de date dedicată, ceea ce permite o separare clară a responsabilităților și o mai bună scalabilitate.

**Responsabilitățile Monitoring Management:**

Primește evenimente de la simulatoarele de senzori prin RabbitMQ: Acest microserviciu se abonează la mesaje transmise prin RabbitMQ, care colectează date de la simulatoarele de senzori. Astfel, Monitoring Management procesează și gestionează aceste evenimente în mod asincron.

Se conectează la Device Management printr-un REST API: Pentru a colecta informații detaliate despre dispozitivele asociate evenimentelor primite, Monitoring Management utilizează endpoint-uri REST expuse de Device Management.

**Rolul RabbitMQ:**

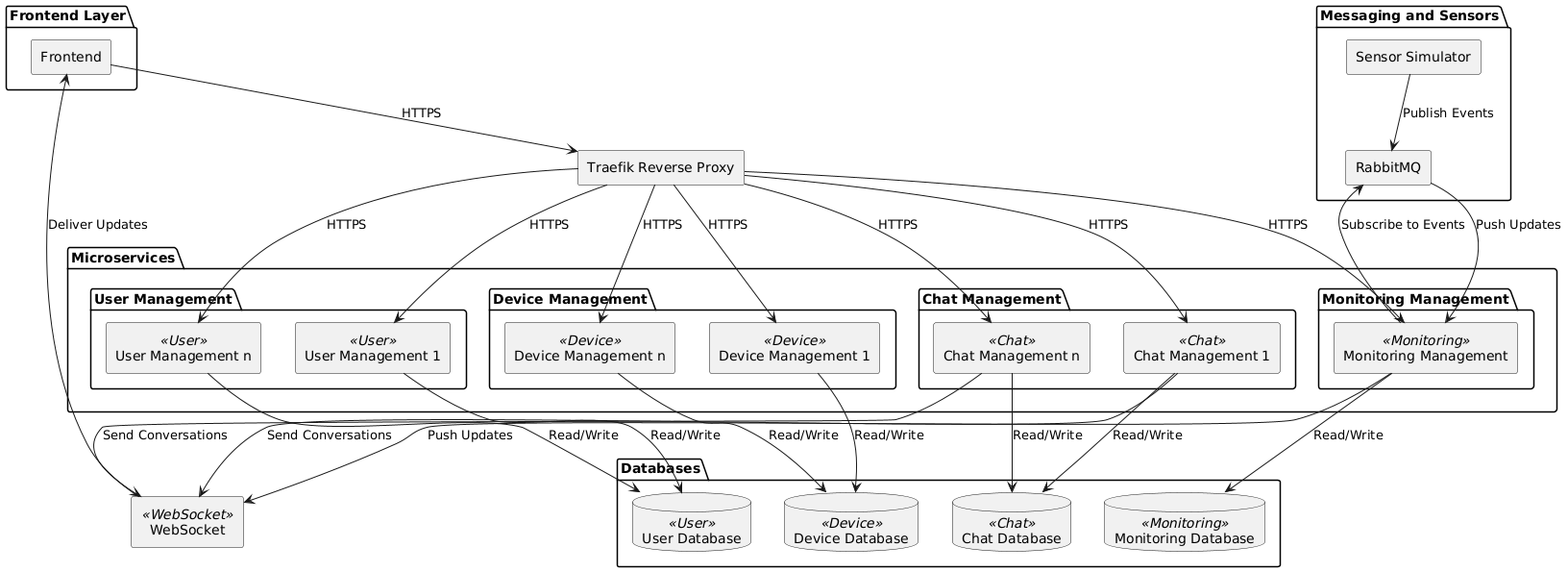
RabbitMQ acționează ca un punct central pentru procesarea mesajelor asincrone. Simulatoarele de senzori publică evenimente către RabbitMQ, care sunt ulterior preluate de Monitoring Management pentru analiză, procesare și stocare în baza sa de date. Acest mecanism oferă flexibilitate, reziliență și posibilitatea de a scala procesarea evenimentelor fără a afecta restul sistemului.

**Funcționalitățile Chat Management:**

Chat Management este responsabil de gestionarea comunicării în timp real între utilizatori prin:

* WebSocket: Utilizează un protocol bidirecțional pentru a facilita transmiterea rapidă a mesajelor și notificărilor între utilizatori. Acest lucru elimină necesitatea de a face polling frecvent.
* Notificări: Transmite notificări despre mesaje noi participanților la o sesiune de chat, inclusiv celor care nu sunt activi în acel moment.
* Indicatoare de scriere (Typing Indicators): Informațiile despre participanții care tastează sunt transmise în timp real celorlalți utilizatori din conversație.
* Stocarea conversațiilor: Mesajele și alte detalii ale sesiunilor de chat sunt salvate în baza de date dedicată (Chat Database) pentru acces ulterior sau analize.

Chat Management este crucial pentru a oferi o experiență fluidă și interactivă utilizatorilor. Prin integrarea cu WebSocket, notificările și indicatorii de scriere sunt gestionate în timp real, reducând latența și îmbunătățind utilizarea resurselor. De asemenea, WebSocket permite gestionarea eficientă a sesiunilor active și scalarea sistemului pentru a suporta un număr mare de utilizatori.



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 securizat la https://localhost:3000, dashboard-ul Traefik la https://localhost:8080 și RabbitMQ Management Console la https://localhost:15672 (username: guest, password: guest). Notă importantă: De acum înainte, toate cererile trebuie efectuate prin HTTPS, deoarece am adăugat suport pentru conexiuni securizate. Asigurați-vă că utilizați adresele HTTPS în toate interacțiunile cu aplicația.P
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>.
9. Pentru a folosi chatul urmam urmaotorii pasi: După ce aplicația este funcțională, accesați frontend-ul la https:// frontend-react.localhost și autentificați-vă sau înregistrați-vă. Odată conectat, accesați secțiunea de chat pentru a iniția o conversație. Chat-ul folosește WebSocket pentru comunicare în timp real, ceea ce înseamnă că mesajele sunt transmise instantaneu între utilizatori. Indicatorii de scriere (typing indicators) permit vizualizarea în timp real a participanților care scriu mesaje. Notificările pentru noi mesaje sunt afișate automat în interfață, chiar dacă nu sunteți activ în fereastra de chat. Toate conversațiile sunt salvate în baza de date chat-db, iar istoricul este accesibil ulterior. Dacă doriți să interacționați cu mai mulți utilizatori, puteți crea sesiuni de chat de grup, iar notificările și mesajele vor fi transmise simultan către toți participanții. Chat-ul este optimizat pentru latență scăzută și poate gestiona eficient un număr mare de conexiuni active, folosind exclusiv HTTPS pentru a asigura securitatea datelor transmise.