Documentație pentru Aplicații cu Sisteme Distribuite

Stefan Jiroveanu

January 11, 2024

1 Introducere

O aplicație bazata pe sisteme distribuite este o aplicație software care funcționează pe mai multe servere sau dispozitive care interacționează intre ele pentru a oferi funcționalități complexe. Aceste sisteme sunt dezvoltate pentru a împărți sarcinile și datele între diferite componente pentru a îmbunătăți performanța aplicatiei.

2 Microservicii

Microserviciile reprezinta un pattern arhitecutral de dezvoltare software care împarte o aplicație în componente independente numite microservicii. Fiecare microserviciu este specializat în furnizarea unei anumite funcționalități putand fi dezvoltat, implementat și scalat independent. Această abordare aduce beneficii precum:

- Scalabilitate
- Rezistență la defecțiuni
- Dezvoltare și livrare rapide

2.1 Arhitectura sistemului disribuit

Figura 1 reprezinta o arhitectură bazata pe microservicii. Aceasta arata modul in care o aplicatie client intercomunica cu doua micro-servicii dezvoltate in Node.Js. Serviciul pentru Utilizatori contine, de asemenea, un layer de Autentificare/Autorizare ce are ca rol de gateway pentru a securiza serviciul pentru useri. Cele doua servicii sunt conectate la doua baze de date dezvoltate in Postgres.

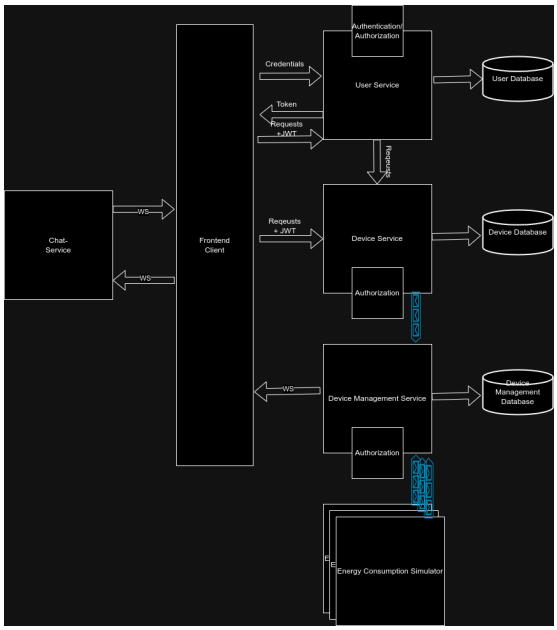

Figure 1: Arhitectură Bazată pe Microservicii

3 Docker

Docker este o platformă de containerizare care permite împachetarea și rularea aplicațiilor și serviciilor în containere izolate. Un container Docker include toate dependintele necesare pentru aplicație, cum ar fi biblioteci, fișiere de configurare și codul aplicației în sine. Aceste containere pot fi ușor distribuite și rulate pe mai multe medii.

3.1 Deployment

Figura 2 ilustrează modul în care microserviciile, bazele de date si aplicatia client sunt lansate prin intermediul docker. Docker-ul a fost ales datorita urmatoarelor avantaje:

- Portabilitate
- Izolare
- Reproducibilitate

4 Securitate

JSON Web Token (JWT) este un protcol pentru transmiterea de informații între aplicatii într-un format compact și sigur. Aceste token- uri sunt utilizate pentru

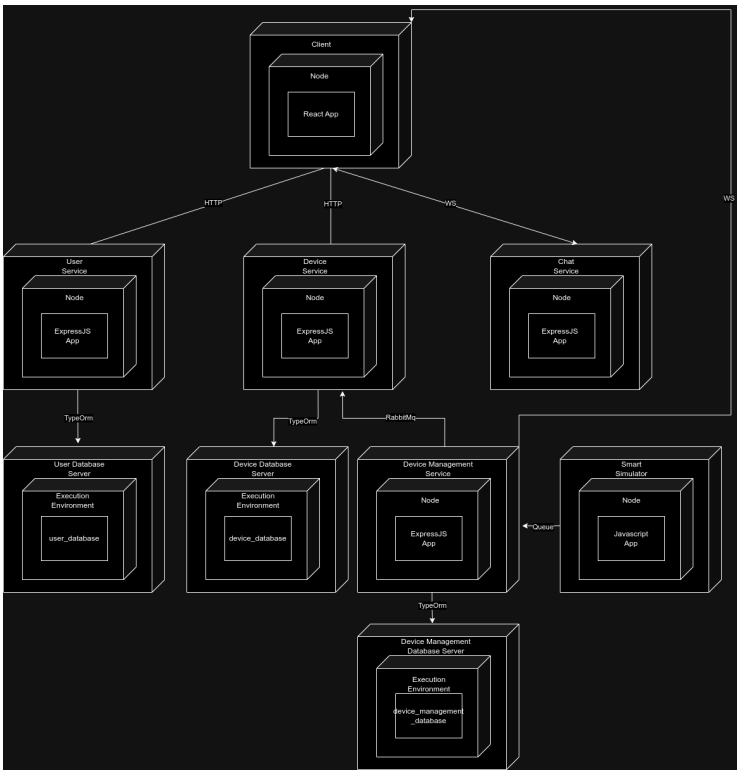

Figure 2: Deployment

autentificare și autorizare în aplicațiile web și serviciile web.

Autentificarea cu JWT se desfășoară în următorii pași:

1. Autentificare
2. Generarea Tokenului
3. Transmiterea Tokenului
4. Verificarea Tokenului

In final, s-a decis folosirea securitatii asupra tuturor serviciilor asa ca am ales strategia, in urma careia, toate micro-serviciile au acces la cheia de codificare/decodficare a token-ilor, fiecare dintre acestea ocupandu-se de decodificare. Singurul care

5 Websocket

WebSocket-urile reprezintă un protocol de comunicare bidirecțională, ce permite transferul de date între un server și un client în timp real, într-o conexiune per-

sistentă. Ele au fost create pentru a depăși limitările comunicării unidirecționale oferite de HTTP, facilitând schimbul rapid de informații între browser și server fără a necesita cereri repetitive din partea clientului.

În aplicația mea, am folosit WebSocket-uri pentru a realiza comunicarea între server și client, având ca scop afișarea în client a unor grafice ce arată consumul unui dispozitiv pe oră.

De asemenea, atunci când un dispozitiv depășește limita maximă stabilită de utilizator, o notificare va apărea în meniul dedicat secțiunii dispozitivului.

6 Smart Meter Simulator

Smart Meter Simulator este o aplicatie stand-alone dezvoltat in Node, care citeste dintr-un fisier CSV datele corespunzatoare pentru a genera informatii despre senzorii device-urilor. Pentru a simula functionarea mai multor device-uri, ar trebui sa pornim mai multe instante ale acestei aplicatii si in fisierul JSON trebuie modificat ID-ul device-ului pentru care pornim simulatorul.

7 RabbitMq

RabbitMQ reprezintă un sistem de mesagerie open-source care facilitează schimbul de mesaje între diferite componente ale unei aplicații, permițând comunicarea eficientă și fiabilă între acestea. În contextul unei aplicații ce gestionează măsurătorile consumului de energie, RabbitMQ poate juca un rol esențial în transmiterea datelor între diversele componente ale sistemului.

Aplicația desktop care generează valorile pentru consumul de energie poate folosi RabbitMQ pentru a publica aceste date către un anumit "queue" (coadă). Acest queue este accesat de către alte componente sau servicii din infrastructură pentru a prelua și procesa aceste date. Această arhitectură asigură o separare clară a responsabilităților și permite scalabilitate, deoarece multiple servicii pot consuma datele publicate în queue-ul respectiv.

În ceea ce privește transportul de date între serviciul de măsurare al consumului și serviciul pentru dispozitive, RabbitMQ a fost utilizat pentru a facilita această comunicare. Serviciul de măsurare a consumului poate publica date relevante într-un queue specific, iar serviciul dedicat dispozitivelor poate consuma aceste mesaje pentru a actualiza informațiile referitoare la dispozitive și pentru a gestiona situațiile în care consumul depășește limitele stabilite.

8 Chat Micro-Service

Cu acest microserviciu s-a implementat functionalitatea de chat prin folosirea tehnologiei WebSocket si a unei librarii React dedicate, numite ChatScope. Pentru implementarea chat-ului s-a creeat un sistem, unde doua WebSocket-uri (unul respectiv pentru un Admin si unul pentru un User) sunt pair-uite pentru a trimite mesaje unul altuia. In momentul in care se doreste conectarea altui

Admin la un User, conexiunea cu primul admin este retinuta, iar mesajele se trimit in continuare catre acesta, deoarece toti adminii prezenti la discutie sa aiba toate informatiile pe care User-ul le ofera.

Exista o functionalitate de "Typing..." in interiorul unei conexiuni, aceasta a putut fii facilitata de catre libraria ChatScope ce are o componenta React special pentru acesasta functionalitate. De asemenea, exista si o functionalitate de "Seen", care se activeaza atunci cand unul dintre participantii la conversatie citeste mesajul.

9 Concluzie

Am ales tech-stack-ul prezentat mai sus datorita mai multor factor precum: dezvoltare si lansare rapida, portabilitate, scalare si securitate marita.