ENERGY MANAGEMENT SYSTEM

Tarța Antonia-Maria

2024

Cuprins

Capitolul 1

Introducere

1.1 Scopul și motivația proiectului

Scopul acestui proiect este dezvoltarea unui Sistem de Gestionare a Energiei care să permită utilizatorilor să monitorizeze și să gestioneze eficient consumul de energie prin intermediul unei aplicații web bazate pe microservicii. Având în vedere creșterea continuă a costurilor energiei și nevoia de a optimiza consumul pentru a reduce amprenta de carbon, acest sistem oferă o soluție viabilă pentru gestionarea energiei. Utilizatorii vor putea vizualiza datele de consum, să identifice tendințele și să ia decizii pentru economisirea energiei.

1.2 Contribuție personală

Contribuția personală la acest sistem include proiectarea și implementarea arhitecturii sistemului, dezvoltarea microserviciilor pentru gestionarea utilizatorilor și dispozitivelor, precum și integrarea interfeței utilizator pentru a asigura o experiență intuitivă. Am utilizat tehnologii precum Java Spring pentru microservicii și Angular pentru frontend, ceea ce a permis crearea unei aplicații eficiente și scalabile.

Capitolul 2

Studiu bibliografic

Soluția propusă se bazează pe arhitectura microserviciilor, care permite dezvoltarea și scalarea independentă a fiecărei componente a sistemului. Utilizatorii se vor putea autentifica în funcție de rolul lor, având acces la funcționalități specifice: administratorii vor putea gestiona utilizatorii și dispozitivele, în timp ce clienții vor putea vizualiza datele de consum ale dispozitivelor asociate.

Capitolul 3

Analiză și implementare

3.1 Descrierea generală a soluției

Sistemul este alcătuit din patru microservicii pentru: gestionarea utilizatorilor, gestionarea dispozitivelor, monitorizare consum și chat. Aceste microservicii comunică între ele prin intermediul API-urilor REST, asigurând o separare clară a responsabilităților. Microserviciile sunt

implementate folosind Java Spring. Frontend-ul aplicației este dezvoltat folosind Angular. Acesta a permis crearea unei interfețe de utilizator interactive și responsive. Utilizatorii pot accesa aplicația printr-un browser web, unde pot efectua acțiuni în funcție de rolurile lor (administrator sau client). Interfața este concepută pentru a fi intuitivă, facilitând navigarea între diferitele funcționalități ale sistemului, cum ar fi autentificarea, vizualizarea dispozitivelor și gestionarea conturilor de utilizatori. Frontend-ul se conectează la microserviciile de backend prin intermediul cererilor HTTP, trimițând și primind datele necesare pentru a îndeplini acțiunile utilizatorilor. Această arhitectură permite o gestionare eficientă a resurselor și o dezvoltare ulterioară a proiectului.

3.2 Conexiunea cu baza de date

Asigurarea funcționării corecte a sistemului de gestionare a energiei, avem nevoie de conexiuni baza de date. Sistemul utilizează două baze de date distincte: una pentru gestionarea utilizatorilor și alta pentru gestionarea dispozitivelor. Această separare permite o organizare mai bună a datelor și îmbunătățește securitatea, deoarece fiecare microserviciu are acces doar la informațiile necesare funcționării sale. Pentru a asocia utilizatorii cu dispozitivele lor în microserviciul de gestionare a dispozitivelor, sistemul folosește o entitate numită PersonReference. Rolul acestei entități este de a menține o referință la utilizator în cadrul microserviciului de dispozitive, fără a stoca întreaga entitate a utilizatorului sau informații personale detaliate.

Folosirea PersonReference permite microserviciului de dispozitive să își îndeplinească funcția de gestionarea a dispozitivelor fără să acceseze direct datele utilizatorilor, care sunt gestionate exclusiv în microserviciul de utilizatori. Această abordare îmbunătățește securitatea și modularitatea sistemului. Prin stocarea doar a unui id, în loc de întreaga entitate a utilizatorului, microserviciul de dispozitive nu depinde de structura internă a entităților din microserviciul de utilizatori. Astfel, dacă structura utilizatorilor se schimbă, microserviciul de dispozitive va fi mai puțin afectat. Stocarea doar a referințelor simplifică structura bazei de date și permite requesturilor pentru informațiile despre dispozitive să fie rapide și eficiente, fără a se accesa detalii suplimentare din microserviciul de utilizatori.

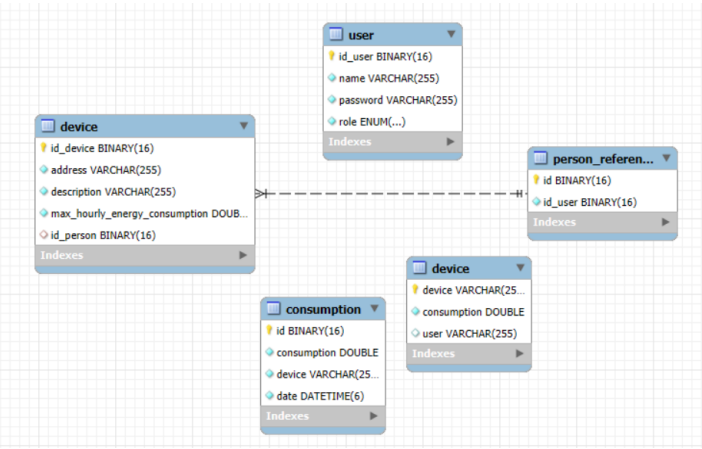


Figura 1. DB schemas

Figura 1 este o reprezentare vizuală a tabelelor din cele trei baze de date și a fost adăugată pentru o înțelegere simplificată a modului in care datele sunt gestionate.

3.3 Securitatea sistemului

Pentru a asigura un acces controlat și sigur în cadrul sistemului, am implementat un mecanism de autentificare și autorizare bazat pe JSON Web Token (JWT).

Autentificarea în sistem se realizează printr-un endpoint /api/auth/login, unde utilizatorii (administratorii și clienții) își introduc datele de login. Odată ce sunt autentificați cu succes, se generează un token JWT.

Adnotările @PreAuthorize sunt aplicate la nivel de controller pentru a restricționa accesul la anumite rute pe baza rolului utilizatorului. Astfel:

- **Administratorii** pot accesa și modifica resursele pentru utilizatori și dispozitive, cum ar fi crearea, ștergerea și asocierea utilizatorilor cu dispozitivele.
- **Clienții** pot vizualiza dispozitivele asociate contului lor.

Acest mecanism asigură că utilizatorii nu pot accesa funcțiile celuilalt rol.

În frontend, rutarea este configurată astfel încât fiecare utilizator să fie redirecționat către paginile specifice rolului său (admin sau client) după autentificare. Această separare a rutelor contribuie la securitatea aplicației, prevenind accesul clienților la funcțiile administrative. După login și primirea tokenului JWT de la backend, aplicația de frontend decodează tokenul pentru a obține informațiile despre rolul acestuia. Pentru a preveni accesul utilizatorilor la paginile ce nu corespund rolului lor, frontend-ul folosește gărzi de rutare (route guards). Rutele destinate clienților sunt vizibile doar utilizatorilor autentificați cu rol de client, iar administratorii nu au acces la aceste pagini. Această abordare previne orice încercare de acces neautorizat la paginile de administrare prin copierea URL-ului, deoarece aplicația va verifica rolul și va redirecționa utilizatorul dacă încearcă să acceseze o rută nepermisă.

Capitolul 4

Arhitectura conceptuală

- **Presentation Layer** - Frontend-ul (Angular).
- **Business Layer** – Controllers.

Prelucrează cererile de la frontend și determină dacă acestea sunt valide conform regulilor de stabilite.

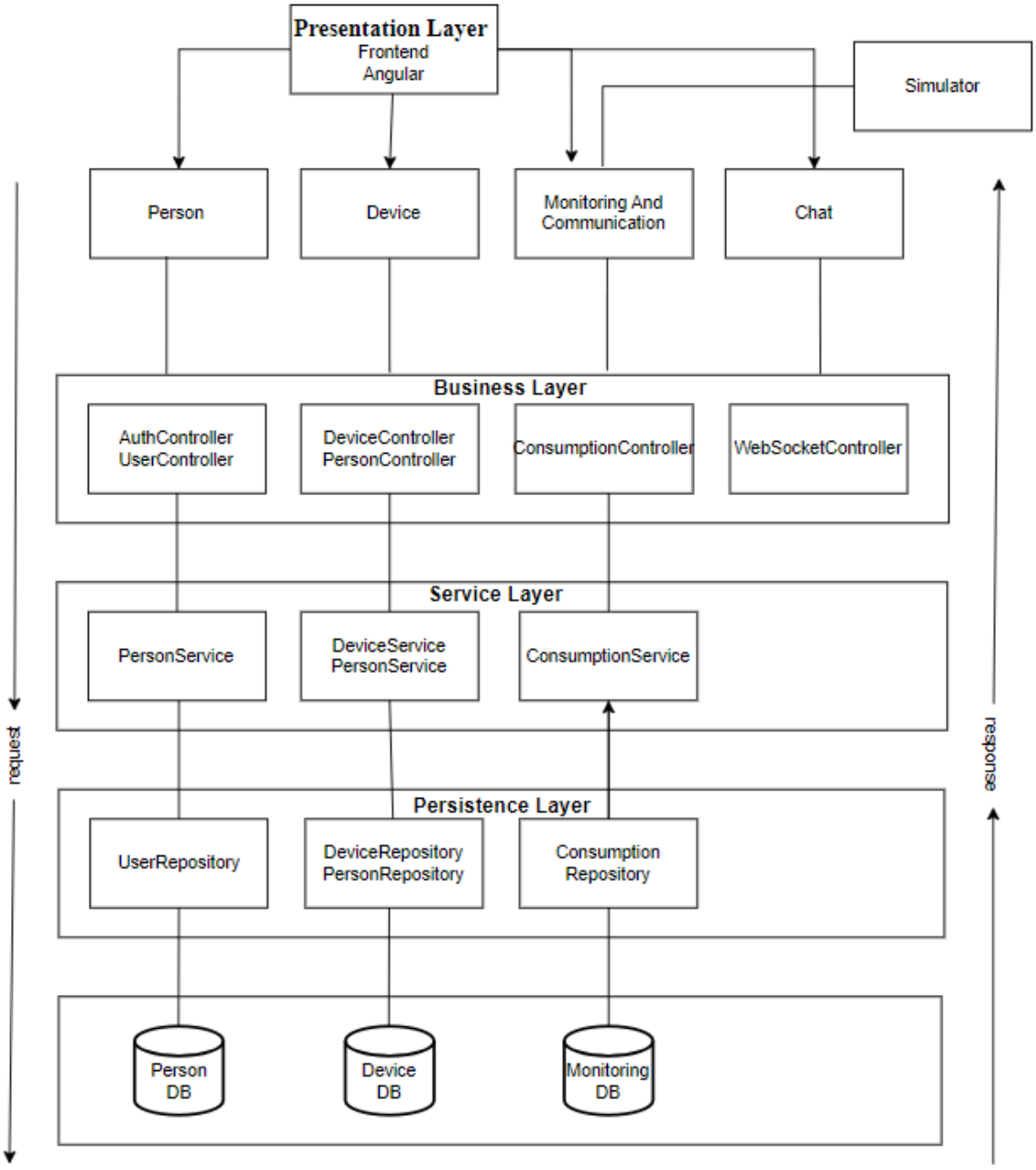
- **Service Layer** – Services.

Intermediază interacțiunile între business layer și persistence layer. Aceste servicii sunt cele care preiau cererile, le prelucrează și apoi apelează repository-urile pentru a interacționa cu baza de date.

- **Persistence Layer** – Repositories.

Gestionează operațiile CRUD pe baza de date (selectare, inserare, actualizare și ștergere).

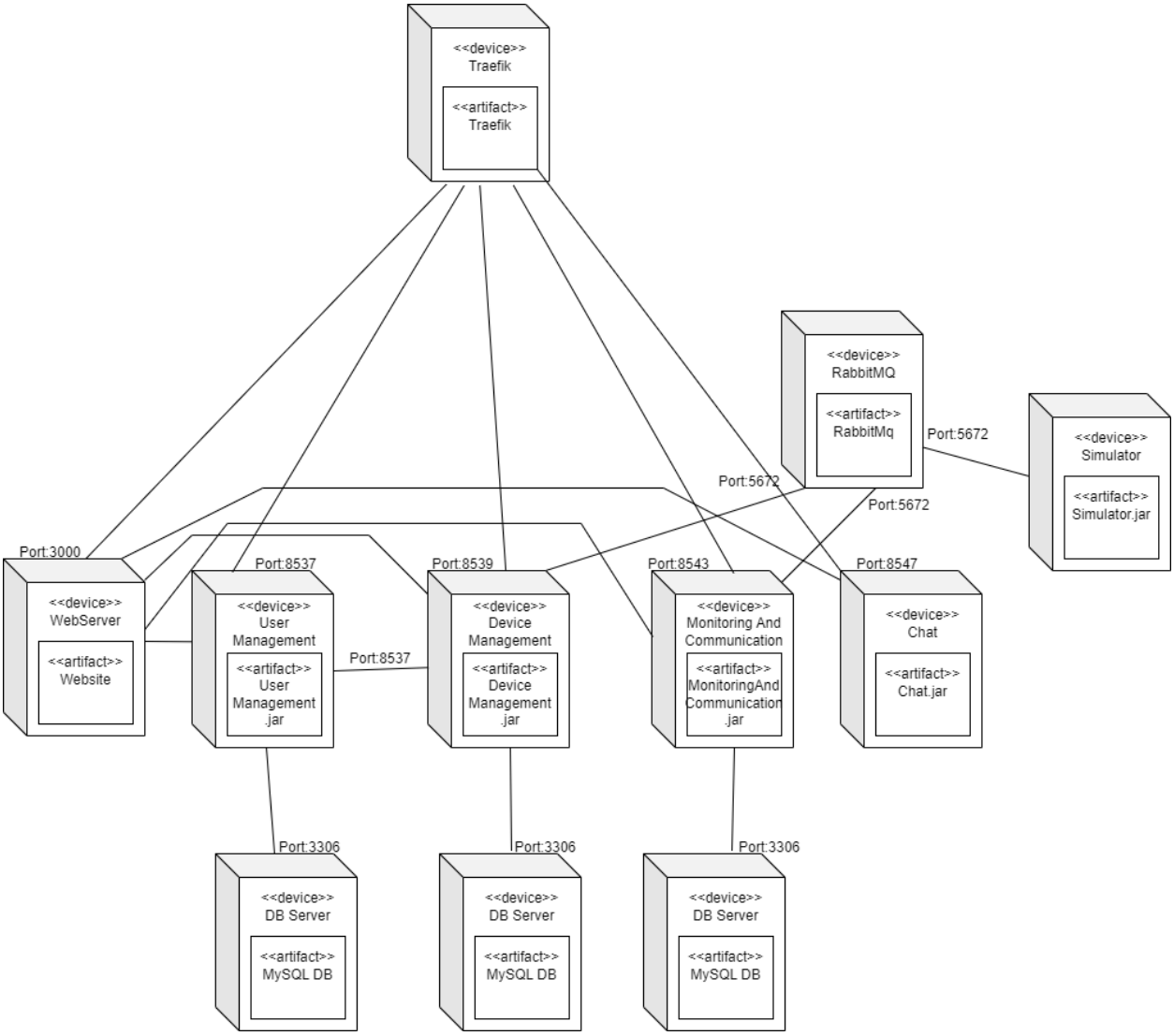
- **Database Layer** - Baza de date MySQL.



Figură 2. Conceptual architecture diagram

Capitolul 5

UML Deployment Diagram



Figură 3. UML Deployment Diagram

Sistemul este configurat să ruleze într-un mediu Docker, folosind containere Docker distincte.

1. **Containerul Docker pentru aplicația frontend**:

   Aplicația frontend rulează un server NGINX și este accesibilă prin Traefik la:

   URL: http://angular.localhost

Port: 4200

2. **Containerul Docker pentru aplicația backend de management al utilizatorilor**:

   Aplicația backend pentru gestionarea utilizatorilor este disponibilă la:

   URL: http://users.localhost

   Port: 8537

3. **Containerul Docker pentru aplicația backend de management dispozitivelor**:

   Aplicația backend pentru gestionarea dispozitivelor poate fi accesată la:

   URL: http://devices.localhost

   Port: 8539

4. **Containerul Docker pentru aplicația backend de monitorizare a consumului:**

   Serviciul de monitorizare este disponibil la:

   URL: http://monitoring.localhost

   Port: 8543

5. **Containerul Docker pentru bazele de date**: Serverul MySQL rulează pe porturile 3307, 3308 și 3309 ale containerului, iar acest port este mapat la portul 3306 al computerului gazdă, astfel încât să poți accesa baza de date MySQL la adresa localhost:3306.

6. **Traefik:** Traefik pentru monitorizarea și gestionarea serviciilor.

   Port: 8080

7. **Containerul Docker pentru RabbitMQ:**

   Interfața de administrare RabbitMQ poate fi accesată la:

   URL: http://localhost:15672

   Port intern RabbitMQ: 5672

   Port interfață management: 15672

8. **Containerul Docker pentru aplicația backend - chat:**

   Serviciul de monitorizare este disponibil la:

   URL: http://chat.localhost

   Port: 8547