DISTRIBUTED SYSTEMS

Energy Management System

Assignment 3

Student: Molnar Zsanett-Ingrid

Grupa: 30644

# Arhitectura conceptuală a sistemului distribuit

Sistemul de management al energiei este o aplicație full-stack concepută pentru a satisface două roluri distincte de utilizator: clienți și administratori. Pentru clienți, aplicația servește pentru a accesa informații despre dispozitivele lor, în timp ce pentru administratori, servește ca un instrument cuprinzător pentru a supraveghea gestionarea dispozitivelor, gestionarea clienților și relațiile dintre ei.

Această aplicație cuprinde două microservicii, fiecare cu un accent specific: unul pentru gestionarea utilizatorilor și celălalt pentru gestionarea dispozitivelor. Ambele microservicii sunt dezvoltate folosind Java Spring. Frontendul aplicației a fost dezvoltată cu Angular.

În cadrul fiecărui microserviciu, se utilizează o arhitectură stratificată bine definită, organizată în următoarele straturi:

1. **Domain Layer**: Acest strat încapsulează entitățile de bază ale aplicației și definește interfețe care conturează modul în care diverse componente interacționează.
2. **Infrastructure Layer**: Stratul de infrastructură găzduiește repository-urile responsabile pentru interacțiunile cu baza de date, facilitând stocarea și extragerea datelor.
3. **Application Layer**: În acest strat, logica aplicației este centralizată, guvernând modul în care funcțiile și caracteristicile diverse ale aplicației funcționează.
4. **API Layer**: Stratul API conține controlere care expun puncte finale, permițând entităților externe (de exemplu, clienților și administratorilor) să interacționeze cu microserviciile.

Această arhitectură asigură o abordare structurată și organizată pentru dezvoltarea și întreținerea Sistemului de Management al Energiei, facilitând gestionarea eficientă a utilizatorilor, dispozitivelor și asocierea lor.

Microserviciul de Gestionare a Utilizatorilor oferă un set de puncte terminale proiectate pentru a gestiona operațiunile legate de utilizatori, inclusiv înregistrarea, autentificarea, actualizarea și ștergerea utilizatorilor. Iată o prezentare a funcționalității pe care o oferă:

1. **Înregistrarea Utilizatorilor**: Această punct de terminale permite utilizatorilor să se înregistreze furnizându-și numele, adresa de email și parola. Microserviciul criptează în mod sigur parola înainte de a stoca informațiile utilizatorului în baza de date.
2. **Autentificarea Utilizatorilor**: Utilizatorii se pot autentifica furnizând adresa lor de email și parola. După autentificare reușită, microserviciul emite un token care conține rolul utilizatorului ca reivindicare. Acest token este folosit pentru acțiunile autorizate ulterioare.
3. **Actualizarea și Ștergerea Utilizatorilor**: Capacitatea de a actualiza și șterge conturile de utilizator este accesibilă exclusiv utilizatorilor cu rolul de administrator. Pentru actualizarea utilizatorilor, administratorii pot modifica numele și adresa de email. Ștergerea utilizatorilor permite administratorilor să elimine conturile de utilizator.
4. **Sincronizare cu Microserviciul de Gestionare a Dispozitivelor**: Pentru a asigura coerența, orice modificări efectuate prin intermediul Microserviciului de Gestionare a Utilizatorilor (creare, actualizare sau ștergere) sunt sincronizate cu Microserviciul de Gestionare a Dispozitivelor. Microserviciul de Gestionare a Dispozitivelor expune puncte terminale corespunzătoare pentru a propaga aceste modificări în tabelul de utilizatori în propriul său serviciu.

În microserviciul de Gestionare a Utilizatorilor am integrat partea de **securitate** a aplicației. Am implementat o autentificare bazată pe JWT, prin care serviciul de Gestionare a Utilizatorilor generează token-uri recunoscute de celelalte microservicii care partajează aceeași cheie secretă ca și serviciul de autorizare. Această arhitectură asigură o autentificare și autorizare eficientă a utilizatorilor în întregul sistem. Alegerea acestei abordări consolidează securitatea sistemului prin utilizarea unui mecanism centralizat de autorizare și autentificare. În implementarea securității, două clase au un rol principal: JwtService, LoginService.

* **Clasa JwtService:** aceasta reprezintă un serviciu pentru manipularea token-urilor JWT utilizate în procesul de autentificare și autorizare. Are două metode. Metoda **generateJwtToken** generează un token JWT pe baza numelui de utilizator și a identificatorului de utilizator (userID) furnizate. Token-ul este configurat cu următoarele atribute:
* Subiectul token-ului este setat pe numele de utilizator
* ID-ul token-ului este setat pe valoarea String-ului reprezentând userID.
* Data emiterii (Issued At) este setată la data și ora curentă.
* Algoritmul de semnare (Signature Algorithm) folosit pentru generarea token-ului este HS512 (HMAC SHA-512).

Metoda **isTokenValid** verifică dacă un token JWT furnizat este valid sau nu. Aceasta efectuează următoarele verificări:

* Parsează token-ul utilizând cheia secretă specificată (secretKey).
* Verifică dacă token-ul nu a expirat.
* Gestionează excepțiile generate de biblioteca io.jsonwebtoken, inclusiv ExpiredJwtException pentru cazul în care token-ul a expirat și SignatureException pentru cazul în care semnătura token-ului nu se potrivește cu cheia secretă.
* **Clasa LoginController:** Funcționalitatea principală a acestei clase este de a gestiona procesul de autentificare a utilizatorilor și de a furniza un token JWT (JSON Web Token) în cazul în care autentificarea este reușită. Are un endpoint: **loginRequest.** Metoda primește un obiect de tip LoginUserDTO din corpul cererii. Acest obiect conține informațiile de autentificare ale utilizatorului, cum ar fi numele de utilizator și parola.
* Utilizând serviciul loginService, se încearcă autentificarea utilizatorului pe baza informațiilor primite. În caz de succes, se generează un token JWT utilizând JwtService.
* Răspunsul HTTP este o entitate de tip LoginResponse care conține detalii despre utilizator și token-ul generat. Răspunsul are statusul HttpStatus.OK

Microserviciul de Gestionare a Dispozitivelor furnizează o colecție de puncte terminale proiectate pentru gestionarea dispozitivelor și pentru a facilita sincronizarea cu Microserviciul de Gestionare a Utilizatorilor care menține datele utilizatorilor, constând în principal din ID-uri de utilizator și adrese de email. Acestea sunt caracteristicile cheie ale microserviciului:

1. **Operațiuni CRUD pentru Dispozitive**: Microserviciul oferă puncte terminale care permit operațiuni Create, Read, Update și Delete (CRUD) asupra tabelului de dispozitive. Aceste operații permit administrarea înregistrărilor dispozitivelor și sunt accesibile exclusiv utilizatorilor cu rolul de administrator.
2. **Sincronizare cu Tabelul de Utilizatori:** Pentru a menține consistența datelor între servicii, microserviciul include puncte terminale pentru sincronizarea datelor cu un alt serviciu responsabil pentru gestionarea informațiilor despre utilizatori. Tabelul de utilizatori din acest serviciu conține în principal ID-uri de utilizator și adrese de email.
3. **Atribuire Utilizator-Dispozitiv:** În plus, microserviciul facilitează atribuirea dispozitivelor utilizatorilor, oferind posibilitatea atât de a adăuga dispozitive utilizatorilor, cât și de a le elimina. Aceste operații sunt, de asemenea, rezervate utilizatorilor cu privilegii administrative.

**Microserviciul de Monitorizare și Comunicare** își propune să supravegheze și să administreze procesele de comunicare. Funcționalitatea sa cuprinde o componentă de consum de mesaje responsabilă pentru preluarea măsurătorilor de la un simulator, stocarea acestor informații într-o bază de date și declanșarea notificărilor în cazul în care consumul mediu depășește consumul maxim orar predefinit. În plus, microserviciul interacționează cu microserviciul de dispozitive pentru a obține informații despre dispozitive. Acesta oferă un endpoint dedicat prin intermediul căruia datele sunt puse la dispoziție pentru reprezentare grafică.

**Microserviciul de Chat** facilitează comunicarea între utilizatori și administratorul sistemului, permițându-le să pună întrebări și să primească răspunsuri. Funcționalitatea sa principală este gestionarea conversațiilor între utilizatori și administrator, oferind o platformă de comunicare eficientă în cadrul Sistemului de Management Energetic. La baza functionalitatii chatului se afla Serviciul de Chat, si WebsocketControllerul.

* **Clasa Chat:** se ocupa de manipularea si transmiterea mesajelor. Folosește SimpMessagingTemplate pentru a converti și trimite mesajul către destinația specificată. Metoda **sendMessage** este responsabilă pentru transmiterea mesajelor obișnuite către un anumit destinatar (reciever) prin intermediul WebSocket. Mesajele trimise sunt obiecte de tipul Message, care conțin informații precum textul mesajului, expeditorul, destinatarul și timpul trimiterii. Metoda **sendTypingMessage** este responsabilă pentru transmiterea mesajelor de tip typing către destinatar . Mesajele trimise sunt obiecte de tipul TypingMessage, care conțin fielduri precum textul mesajului, expeditorul, si o valoare booleana care indica daca utilizatorul tasteaza. Metoda **sendSeenNotification** este responsabilă pentru transmiterea mesajelor de tip seen către destinatar . Mesajele trimise sunt obiecte de tipul SeenMessage, care conțin fielduri precum textul mesajului, expeditorul, si o valoare booleana care indica daca s-a citit mesajul.
* **Clasa WebsocketController**: gestionează comunicarea prin WebSocket în cadrul aplicației. Acesta are trei metode, **handleChatMessage, handleTypingNotification, handleSeenNotification.** Fiecare metoda gestioneaza un tip diferit de mesaj, mapate la diferite cai. Ele apeleaza metodele din clasa Chat.

# Proiectarea bazei de date

**Microserviciul de Gestionare a Utilizatorilor:**

În baza de date pentru Microserviciul de Gestionare a Utilizatorilor există o singură tabelă numită "AppUser". Această tabelă include următoarele coloane:

* **UserId**: Un identificator unic pentru fiecare utilizator.
* **Username**: Numele utilizatorului.
* **Password:** Parola utilizatorului
* **Role:** Rolul utilizatorului. Acesta poate fi Client sau Admin.

**Microserviciul de Gestionare a Dispozitivelor:**

Baza de date pentru Microserviciul de Gestionare a Dispozitivelor cuprinde doua tabele:

Tabelul AppUser:

* **UserId**: Un identificator unic pentru fiecare utilizator.
* **Username:** Adresa de email a utilizatorului.

Tabelul Device:

* **Id**: Un identificator unic pentru fiecare dispozitiv.
* **Description**: O descriere a dispozitivului.
* **Address**: Locația fizică a dispozitivului.
* **Consumption**: Cantitatea maximă de energie pe care dispozitivul o poate consuma într-o oră.
* **UserID:** id-ul userului la care este asignat dispozitivul.

Există o relație one-to-many între tabela de AppUser și Device, care permite ca utilizatorul să aibe mai multe deviceuri. Un device se poate asigna doar la un utilizator.

**Microserviciul de Monitorizare și Comunicare**:

Baza de date pentru Microserviciul de Monitorizare și Comunicare cuprinde doua tabele:

Tabelul Device:

* **Id:** Un identificator unic pentru fiecare dispozitiv.
* **Consumption**: Cantitatea maximă de energie pe care dispozitivul o poate consuma într-o oră.
* **UserID:** id-ul userului la care este asignat dispozitivul.

Tabelul Measurement:

* **Id:** Un identificator unic pentru fiecare măsurare
* **Hour:** ora la care s-a făcut măsurarea
* **Measurement\_date:** data la care s-a făcut măsurarea
* **DeviceID:** id-ul deviceului care a măsurat valorile

**User Management Microservice:**

A screenshot of a computer

Description automatically generated

**Device Management Microservice:**

A screenshot of a computer

Description automatically generated

**Monitoring and Communication Microservice:**

A screenshot of a computer

Description automatically generated

# UML Deployment Diagram

Am folosit Docker pentru procesul de implementare, unde am creat cinci containere: unul pentru interfața frontend, câte unul pentru fiecare dintre serviciile backend și încă doua pentru serverul SQL.

A screenshot of a computer

Description automatically generated

# Considerații privind construirea și execuția

Am creat un Dockerfile pentru fiecare microserviciu backend și frontend, apoi am folosit comanda „docker build -t image\_name” pentru a genera o imagine pentru fiecare dintre ele. În plus, am dezvoltat un fișier Docker Compose pentru a defini imaginile, containerele și porturile pentru fiecare tip de serviciu. În cele din urmă, am inițiat containerele executând comanda „docker-compose up -d”.