UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Energy Management System

Sisteme distribuite

Autor: Cojocaru Vicentiu
Grupa: 30242

Facultatea de Automatica si Calculatoare

11 Ianuarie 2024

Cuprins

1 Obiectivul temei

Obiectivul temei este implementarea unui sistem de management al energiei care gestioneaza utilizatorii si dispozitivele lor de masurare a energiei inteligente. SIstemul poate fi accesat de doua tipuri de utilizatori dupa un proces de autentificare: administrator si clienti. Administratorul poate efectua operatiuni CRUD (Create-Read-Update-Delete) asupra conturilor de utilizator (definite prin ID, nume, rol: admin/client), dispozitivelor de masurare a energiei inteligente (definite prin ID, descriere, adresa, consum maxim de energie pe ora) si asupra asignarii utilizatorilor la dispozitive (fiecare utilizator poate detine unul sau mai multe dispozitive inteligente in diferite locatii).

În plus față de aceste funcționalități inițiale, aplicația a fost extinsă cu un microserviciu de monitorizare a consumului de energie. Acest serviciu permite fiecărui client să vizualizeze consumul în timp real al dispozitivelor sale și să primească notificări în cazul în care se depășește limita maximă de consum orar. A fost integrată o bază de date dedicată monitorizării pentru a gestiona informațiile referitoare la consumul de energie.

Sistemul este extins cu o componentă nouă. Un microserviciu de chat facilitează comunicarea între utilizatori și administrator. Utilizatorii accesează o casetă de chat în aplicație, trimit și primesc mesaje către administrator în mod asincron. Administratorul gestionează simultan mai multe conversații cu utilizatorii și primește notificări când un utilizator trimite sau citește un mesaj.

Astfel, arhitectura sistemului constă într-un frontend, două microservicii pentru gestionarea utilizatorilor și a dispozitivelor, un microserviciu adițional pentru monitorizarea consumului de energie, un microserviciu de chat și trei baze de date distincte pentru utilizatori, dispozitive și monitorizare.

2 Analiza problemei, modelare, scenarii, cazuri de utilizare

2.1 Analiza problemei si modelarea

Soluția propusă implică crearea unui sistem complex de management al energiei, inclusiv un frontend interactiv și două microservicii independente. Utilizatorii se vor autentifica în sistem, iar în funcție de rolul lor (administrator sau client), vor fi redirecționați către interfețe specifice. Interfața destinată administratorilor va permite efectuarea operațiilor CRUD pentru gestionarea conturilor de utilizatori, dispozitivelor și asignărilor utilizator-dispozitiv. În același timp, clienții vor avea acces la informații detaliate despre dispozitivele lor de măsurare a energiei. Pentru a asigura securitatea datelor și a funcționalităților, se va implementa un sistem de autentificare pentru a preveni accesul neautorizat la funcționalitățile rezervate administratorilor. Toate aceste funcționalități vor fi implementate în cadrul unui mediu de containerizare Docker, asigurând portabilitate și scalabilitate.

În această soluție propusă, am ales tehnologii potrivite pentru fiecare componentă a sistemului. Pentru partea de frontend, vom utiliza Angular, un cadru de dezvoltare popular și robust, cunoscut pentru crearea interfețelor de utilizator interactive și dinamice. Alegerea Angular-ului va asigura o experiență utilizator fluidă și intuitivă.

Pentru partea de backend, vom folosi Spring Boot, un cadru de dezvoltare Java pentru construirea aplicațiilor Java enterprise. Spring Boot facilitează dezvoltarea rapidă și ușoară a aplicațiilor Java, oferind instrumente puternice pentru gestionarea dependențelor, securitate, gestionarea cererilor HTTP și multe altele.

Pentru stocarea datelor, vom utiliza PostgreSQL, un sistem de gestionare a bazelor de date relaționale (RDBMS) foarte fiabil și robust. PostgreSQL oferă performanță și extensibilitate, ceea ce îl face potrivit pentru gestionarea datelor într-o aplicație complexă de management al energiei.

În ceea ce privește autentificarea și autorizarea, vom implementa un sistem de autentificare bazat pe tokenuri JWT (JSON Web Tokens). Acest mecanism de autentificare va permite generarea unui token JWT după ce un utilizator se autentifică cu succes. Acest token JWT va fi apoi inclus în antetele cererilor HTTP și va fi verificat la nivelul serverului pentru a autoriza accesul la resursele protejate. Folosind JWT, putem asigura securitatea autentificării și autorizării utilizatorilor într-un mod eficient și scalabil.

Această arhitectură, combinând Angular pentru frontend, Spring Boot pentru backend și PostgreSQL pentru stocarea datelor, împreună cu autentificarea bazată pe tokenuri JWT, va asigura dezvoltarea unui sistem de management al energiei sigur, performant și ușor de întreținut.

În cadrul soluției propuse, se adaugă un microserviciu pentru monitorizarea consumului de energie și se integrează RabbitMQ pentru transmiterea notificărilor în cazul depășirii consumului.

Pentru monitorizarea consumului de energie în timp real, se implementează un microserviciu dedicat care va procesa datele trimise de simulatoarele dispozitivelor inteligente de măsurare a energiei. Acest microserviciu va utiliza RabbitMQ ca middleware pentru a permite transmiterea eficientă și gestionarea fluxurilor de date. Datele trimise de dispozitivele inteligente vor fi formatate sub formă de tuple JSON și trimise către RabbitMQ pentru a fi prelucrate de microserviciu.

Microserviciul consumator (consumer) din cadrul acestui microserviciu va procesa fiecare mesaj primit de la RabbitMQ, iar în cazul depășirii consumului orar maxim, va genera notificări. Aceste notificări vor fi transmise asincron, folosind WebSocket, către aplicația clientului asociat dispozitivului respectiv. În același timp, valorile de energie pe oră vor fi salvate în baza de date de monitorizare, pentru a păstra un istoric al consumului.

Integrarea cu RabbitMQ permite transmiterea eficientă a notificărilor către aplicațiile client, permițându-le să primească în timp real informații despre depășirile limitelor de consum orar. Astfel, se asigură o funcționalitate crucială pentru monitorizarea și gestionarea consumului de energie în cadrul sistemului propus.

Această actualizare detaliază adăugarea microserviciului de monitorizare a consumului de energie și integrarea cu RabbitMQ pentru transmiterea notificărilor către aplicațiile client în cazul depășirii limitelor de consum.

În cadrul soluției propuse, se adaugă și un microserviciu dedicat pentru funcționalitatea de chat, implementat utilizând tehnologia WebSockets. Acest serviciu va permite comunicarea în timp real între utilizatori și administrator, oferind o interfață de chat interactivă în aplicație. Fiecare mesaj transmis va fi gestionat de backend și transmis prin WebSockets către destinatarul corespunzător.

Prin intermediul WebSockets, fiecare mesaj trimis de un utilizator este direcționat către destinatarul corespunzător, respectându-se confidențialitatea și siguranța conversațiilor. Astfel, interacțiunile între utilizatori și administrator vor fi realizate în timp real, oferind o experiență de comunicare fluidă și imediată.

Prin integrarea funcționalității de chat cu WebSockets în cadrul sistemului, se adaugă o nouă dimensiune de interactivitate și comunicare, oferind utilizatorilor și administratorilor posibilitatea de a comunica în timp real, completând astfel funcționalitatea sistemului de management al energiei.

Aceasta actualizare adaugă și detaliază implementarea funcționalității de chat utilizând WebSockets, asigurând o comunicare în timp real între utilizatori și administrator, iar fiecare mesaj este gestionat și salvat pentru referințe ulterioare și auditare a conversațiilor.

2.2 Cerinte

Cerintele functionale aferente proiectului sunt:

- Utilizatorii se autentifică și sunt redirecționați către pagina corespunzătoare rolului lor.
- **Rolul de Administrator/Manager:**

  Operații CRUD asupra utilizatorilor
  Operații CRUD asupra dispozitivelor
  Crearea asignării utilizator-dispozitiv
  Un administrator poate vizualiza pe pagina sa grafice cu privire la consumul fiecarui dispozitiv propriu, in functie de data selectata din calendar.
  Un administrator primeste o notificare atunci cand un dispozitiv depaseste limita maxima de consum orara.
- **Rolul de Utilizator/Client:**

  Un client poate vizualiza pe pagina sa toate dispozitivele.
  Un client poate vizualiza pe pagina sa grafice cu privire la consumul fiecarui dispozitiv, in functie de data selectata din calendar.
  Un client primeste o notificare atunci cand un dispozitiv depaseste limita maxima de consum orara.
- **Utilizatorii corespunzători unui rol nu vor putea accesa paginile corespunzătoare altor roluri (de exemplu, prin autentificare și apoi copierea și lipirea URL-ului de administrator în browser).**
- **Componenta Chat:**

  Se afișează o casetă de chat pentru utilizatori.
  Mesajele sunt trimise asincron către administrator împreună cu identificatorul utilizatorului.
  Conversații bidirecționale între utilizator și administrator sunt posibile în timpul unei sesiuni de chat.
  Administratorul poate interacționa cu mai mulți utilizatori simultan.
  Utilizatorul primește notificări când administratorul citește mesajul și invers.
  Notificări sunt afișate utilizatorului în timp ce administratorul sau utilizatorul opus tastează un mesaj.

Cerintele non-functionale aferente proiectului sunt:

- **Microservicii:**

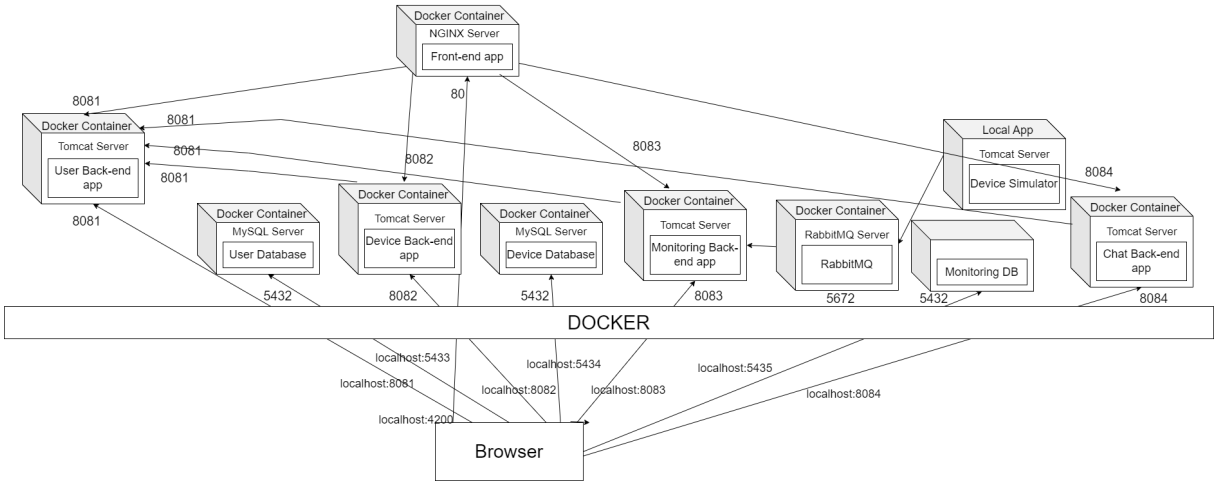
  Microserviciul de Gestionare a Utilizatorilor
  Microserviciul de Gestionare a Dispozitivelor
  Microserviciul de Monitorizare a Consumului
  Microserviciul de Chat
- **Securitate: Utilizarea autentificarii pentru a restricționa accesul utilizatorilor la paginile administratorului (cookie-uri, sesiuni, etc.).**

3 Proiectare

3.1 Diagrama de deployment a aplicatiei

Pe host rulează runtime-ul Docker care va găzdui cinci containere, unul pentru fiecare aplicație:

- Containerul Docker pentru aplicația frontend rulează un server NGINX și face maparea portului local 80 la portul 4200 al host-ului.
- Containerul Docker pentru aplicația backend pentru User rulează un server TOMCAT și face maparea portului local 8081 la portul 8081 al host-ului.
- Containerul Docker pentru serverul de baze de date pentru User rulează un server POSTGRESQL și face maparea portului local 5432 la portul 5433 al host-ului.
- Containerul Docker pentru aplicația backend pentru Device rulează un server TOMCAT și face maparea portului local 8082 la portul 8082 al host-ului.
- Containerul Docker pentru serverul de baze de date pentru Device rulează un server POSTGRESQL și face maparea portului local 5432 la portul 5434 al host-ului.
- Containerul Docker pentru aplicația backend pentru Monitorizare rulează un server TOMCAT și face maparea portului local 8083 la portul 8083 al host-ului.
- Containerul Docker pentru serverul de baze de date pentru monitorizare rulează un server POSTGRESQL și face maparea portului local 5432 la portul 5435 al host-ului.
- Containerul Docker pentru aplicația backend pentru Chat rulează un server TOMCAT și face maparea portului local 8084 la portul 8084 al host-ului.



Aceasta înseamnă că de pe host putem accesa serverele din containere astfel:

- Aplicatia Frontend: utilizand localhost:4200
- Aplicatia Backend-User: utilizand localhost:8081
- Serverul de Baza de Date-User: utilizand localhost:5433
- Aplicatia Backend-Device: utilizand localhost:8082
- Serverul de Baza de Date-Device: utilizand localhost:5434
- Aplicatia Backend-Monitorizare: utilizand localhost:8083
- Serverul de Baza de Date-Monitorizare: utilizand localhost:5435
- Aplicatia Backend-Chat: utilizand localhost:8084

4 Implementare

4.1 Microserviciul de Gestionare a Utilizatorilor

Microserviciul de gestionare a utilizatorilor este responsabil pentru stocarea și administrarea datelor utilizatorilor în sistem. Fiecare utilizator este identificat printr-un ID unic și are un nume asociat. De asemenea, fiecare înregistrare a utilizatorului conține informații despre rolul acestuia în aplicație, fie "admin", fie "client".

În cadrul acestui microserviciu, modelul de date (User) definește structura fiecărui utilizator, incluzând ID-ul, numele și rolul. Repository-ul (UserRepository) furnizează metode pentru a efectua operații de bază asupra utilizatorilor, cum ar fi adăugarea, actualizarea, ștergerea și obținerea detaliilor unui utilizator.

Serviciul (UserService) conține logica de afaceri necesară pentru validarea și manipularea datelor utilizatorilor. Acesta se ocupă de cererile primite din partea controlerului și interacționează cu repository-ul pentru a accesa și modifica datele utilizatorilor.

4.2 Microserviciul de Gestionare a Dispozitivelor de Măsurare a Energiei

Microserviciul de gestionare a dispozitivelor de măsurare a energiei are rolul de a gestiona informațiile legate de aceste dispozitive. Fiecare dispozitiv este caracterizat printr-un ID unic și are asociate o descriere, o adresă și o valoare care reprezintă consumul maxim de energie pe oră.

Un aspect important al acestui microserviciu este sincronizarea tabelei de utilizatori cu cea din microserviciul de utilizatori. Această sincronizare a tabelelor ajută la minimizarea numărului de cereri între cele două microservicii și asigură coerența datelor în cadrul sistemului. Astfel, fiecare dispozitiv este atribuit unui utilizator existent în baza de date, iar această informație este actualizată în timp real, reflectând orice modificare în lista utilizatorilor din microserviciul dedicat utilizatorilor.

Această sincronizare eficientă între tabelele de utilizatori din ambele microservicii contribuie la optimizarea performanței sistemului, reducând complexitatea operațiunilor și minimizând numărul de cereri de rețea necesare pentru a obține și a actualiza informațiile relevante despre utilizatori și dispozitivele asociate.

În structura acestui microserviciu, modelul de date (Device) definește proprietățile fiecărui dispozitiv, inclusiv ID-ul, descrierea, adresa și consumul maxim de energie. Repository-ul (DeviceRepository) furnizează metode pentru a gestiona aceste informații, cum ar fi adăugarea, actualizarea, ștergerea și obținerea detaliilor unui dispozitiv.

Serviciul (DeviceService) conține logica necesară pentru validarea și manipularea datelor dispozitivelor. Acesta interacționează cu repository-ul pentru a realiza operațiile necesare solicitate de controler și asigură integritatea datelor în cadrul sistemului.

4.3 Microserviciul de Monitorizare a Consumului de Energie

Microserviciul de gestionare a consumului de energie operează cu două tipuri principale de date: Device și Devicedata. Device este un tip de date sincronizat cu tabela Device din microserviciul corespunzător dispozitivelor de măsurare a energiei. Acesta conține informații despre dispozitive, inclusiv ID-ul unic, descrierea, adresa și consumul maxim de energie pe oră. Sincronizarea constantă între această entitate și tabela corespunzătoare din microserviciul dispozitivelor este esențială pentru menținerea coerenței datelor în întregul sistem. Devicedata este un alt tip de date care primește informațiile citite de simulatoarele dispozitivelor. Acesta

conține datele efective despre consumul de energie, actualizate periodic de simulatoare. Având în vedere fluxul constant de date provenind de la dispozitive, acest tip de date este crucial pentru monitorizarea și stocarea valorilor de consum în timp real.

În cadrul microserviciului, se integrează RabbitMQ pentru a transmite notificări către frontend. Atunci când apar depășiri ale consumului maxim de energie pe oră pentru anumite dispozitive, microserviciul generează notificări prin RabbitMQ. Aceste notificări sunt trimise către frontend folosind WebSocket, permițând utilizatorilor să primească în timp real avertismente legate de depășirea limitelor de consum.

Astfel, microserviciul de gestionare a consumului de energie nu doar monitorizează și gestionează datele dispozitivelor și consumul de energie, dar și facilitează transmiterea eficientă a notificărilor către interfața de utilizator pentru o experiență în timp real.

4.4 Microserviciul de Chat

Microserviciul de chat reprezintă o parte esențială a interacțiunii în timp real între utilizatori și administrator. Acesta utilizează tehnologia WebSockets pentru a facilita schimbul de mesaje între utilizatori și administrator. Fiecare mesaj trimis de utilizatori este gestionat și direcționat către destinația corespunzătoare.

În cadrul arhitecturii sistemului, microserviciul de chat este integrat în aplicația de frontend, permițând utilizatorilor să comunice în timp real cu administratorul. Notificările sunt trimise prin WebSockets pentru a alerta utilizatorii atunci când un mesaj este primit sau citit de administrator.

Acest microserviciu adaugă o dimensiune interactivă și imediată sistemului, permitând comunicarea eficientă și rapidă între utilizatori și administrator.

4.5 Aplicația de Frontend

Aplicația de frontend dezvoltată în Angular oferă o interfață intuitivă și prietenoasă pentru utilizatori. Aceasta permite autentificarea și autorizarea utilizatorilor prin intermediul unei pagini de autentificare securizate. Aplicația se conectează la microserviciile de gestionare a utilizatorilor și dispozitivelor pentru a obține și actualiza datele.

În procesul de implementare a aplicației frontend în Angular, s-au folosit diverse concepte și tehnologii pentru a asigura o interfață de utilizator robustă și eficientă. Interfața de utilizator a fost împărțită în componente Angular, fiecare reprezentând o parte specifică a aplicației, precum autentificarea, gestionarea utilizatorilor și afișarea dispozitivelor.

Pentru a comunica cu microserviciile backend, aplicația utilizează servicii Angular care fac cereri HTTP pentru a obține și a trimite date către și de la server. Implementarea serviciilor se bazează pe observabilele RxJS, permițând gestionarea asincronă a datelor și evenimentelor.

De asemenea, aplicația utilizează module pentru a organiza componentele și serviciile în funcție de funcționalități. Interfața grafică este realizată utilizând șabloane și stiluri CSS, iar pentru a menține coerența și reutilizarea codului, s-au folosit directive și componente Angular. De asemenea, s-au implementat funcționalități pentru gestionarea stării aplicației, astfel încât să se asigure o interactivitate fluidă și un comportament coerent al interfeței de utilizator.

O caracteristică importantă a aplicației este capacitatea de a afișa lista dispozitivelor de măsurare a energiei asociate fiecărui utilizator, permițându-le să vizualizeze detaliile acestor dispozitive. Utilizatorii pot interacționa cu aplicația pentru a adăuga sau șterge dispozitive și pot vedea informații în timp real despre consumul de energie.

Interfața de utilizator dezvoltată în Angular integrează funcționalitatea de chat, oferind utilizatorilor posibilitatea de a comunica în timp real cu administratorul. În cadrul acestei interfețe, utilizatorii pot trimite și primi mesaje folosind caseta de chat dedicată. Pentru implementarea acestei funcționalități, Angular folosește WebSocket pentru transmiterea și gestionarea mesajelor către și de la microserviciul de chat din backend. Mesajele sunt afișate în timp real, permițând utilizatorilor să interacționeze și să primească răspunsuri imediate din partea administratorului.

5 ReadMe

5.1 Deploy

Clonează acest repository pe computerul local:

git clone https://gitlab.com/ds2023_30242_cojocaru_vicentiu/ds2023_30242_cojocaru_vicentiu

Rulați următoarea comandă pentru a porni aplicația:

docker-compose up

5.2 Accesarea aplicatiei

După ce aplicația este implementată, puteți accesa serviciile de pe mașina gazdă folosind următoarele endpoint-uri:

Aplicația Frontend: Accesați aplicația frontend la **http://localhost:4200**.

Serviciul Backend pentru Useri: Serviciul backend pentru utilizatori este accesibil la **http://localhost:8081**.

Serviciul Backend pentru Device-uri: Serviciul backend pentru dispozitive este accesibil la **http://localhost:8082**.

Serviciul Backend pentru Monitorizare: Serviciul backend pentru monitorizare este accesibil la **http://localhost:8083**.

Serviciul Backend pentru Chat: Serviciul backend pentru monitorizare este accesibil la **http://localhost:8084**.

6 Concluzii

Această experiență a marcat debutul meu în universul complex al microserviciilor și Docker, fiind o călătorie captivantă în lumea dezvoltării software modernă. În această călătorie, am avut oportunitatea să învăț și să explorez adâncimile arhitecturii microserviciilor, descoperind beneficiile modulării și scalabilității.

Prin împărțirea aplicației în microservicii individuale, am reușit să obțin o abordare agilă și flexibilă pentru dezvoltare. Acest proces a implicat gestionarea complexă a dependențelor, dar ne-a oferit, în schimb, libertatea de a actualiza și întreține fiecare componentă în mod independent, adaptându-ne rapid cerințelor în continuă schimbare.