DISTRIBUTED SYSTEMS

Project

Integrated Energy Management System

Grupa:30243

Nita Iosif-Gabriel

Introducere

Am dezvoltat un Sistem de Management al Energiei care consta intr-un frontend si doua microservicii concepute pentru a gestiona utilizatorii si dispozitivele lor inteligente de masurare a energiei asociate. Sistemul poate fi accesat de catre doi tipuri de utilizatori dupa un proces de autentificare:

- Administrator (Admin)
- Clienti (User)

Administratorul poate efectua operatiuni CRUD (Create-Read-Update-Delete) asupra conturilor de utilizatori (definite prin ID, nume, rol: admin/client), dispozitivelor inteligente de masurare a energiei (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 locatii diferite).

Arhitectura conceptuala a sistemului distribuit poate fi reprezentata astfel:

Frontend: Acesta este componenta vizuala a sistemului, care permite utilizatorilor s interactioneze cu acesta. Se bazeaza pe framework-ul Angular, si ofera interfete de utilizator pentru administratori si clienti pentru a efectua autentificarea si operatiunile de gestionare a conturilor si dispozitivelor inteligente.

Microservicii:

a. Microserviciu de gestionare a utilizatorilor: Acest microserviciu se ocupa de operatiile CRUD legate de utilizatori, inclusiv crearea, citirea, actualizarea si stergerea conturilor de utilizatori. Este dezvoltat folosind tehnologia REST, Java Spring.

b. Microserviciu de gestionare a dispozitivelor: Acest microserviciu se concentreaza pe gestionarea dispozitivelor inteligente, inclusiv definirea lor, actualizarea detaliilor si stergerea lor. Acesta, de asemenea, foloseste tehnologii REST pentru a comunica cu frontend-ul si celalalt microserviciu

c. Microserviciu de gestionare a consumarii de energie: Acest microserviciu se concentreaza pe gestionarea consumarii de energie transmitarea mesajelor/afisarea de grafice importante pentru gestionarea resurelor.

Serviciul dedicat gestionarii consumului de energie stabileste o comunicare eficienta cu serviciul de administrare al dispozitivelor prin intermediul unei cozi de mesaje (RabbitMQ).

Comunicare intre microservicii: Aceste microservicii sunt sincronizate cu ajutorul tehnologiilor REST cand este creat un user se trimite prima oara requestul la microserviciul de gestionare a dispozitivelor si se adauga in baza de date, si dupa se adauga in baza de date a userilor. Avand in vedere acest fapt microserviciile sunt sincronizate.

Baza de date: Sistemul distribuit utilizeaza o baze de date pentru a stoca si gestiona informatiile legate de utilizatori, dispozitive si asocierea dintre utilizatori si dispozitive.

Serviciu de autentificare: Acesta asigura autentificarea utilizatorilor in sistem, permitand accesul

doar utilizatorilor autorizati in functie de rol (admin sau user). Acesta este implementat folosind tehnologii de autentificare standard, precum JWT (JSON Web Tokens).

Baza de date a sistemului nostru distribuit este organizata in mod detaliat pentru a gestiona atat utilizatorii, cat si dispozitivele inteligente. Iata o descriere mai detaliata a tabelelor si a modului in care acestea sunt gestionate:

Tabela "User":

- Aceasta tabela face parte din microserviciul care gestioneaza utilizatorii.
- Aici sunt stocati toti utilizatorii, fie ei de tip USER sau ADMIN.
- Fiecare utilizator are un identificator unic (ID), un nume de utilizator (username) si o parola.
- Aceasta tabela reprezinta baza pentru autentificarea si gestionarea utilizatorilor in sistem.

Tabelele "MapperDeviceUser" si "Device":

- Aceste tabele fac parte din microserviciul dedicat gestionarii dispozitivelor.
- Tabela "Device" este destinata sa stocheze toate informatiile legate de dispozitive.
- Aceasta include un identificator unic (ID), o adresa, o descriere, consumul maxim pe ora al fiecarui dispozitiv si ID-ul utilizatorului care detine dispozitivul.
- Tabela "MapperDeviceUser" este folosita pentru a sincroniza datele intre cele doua microservicii.
- Atunci cand se adauga sau se sterge un utilizator, valorile corespunzatoare din ambele tabele ("User" si "MapperDeviceUser") sunt actualizate pentru a mentine consistenta datelor.
- Important de mentionat este faptul ca putem crea un utilizator fara sa ii asignam un dispozitiv, dar nu putem crea un dispozitiv fara sa ii asignam un utilizator. Acest lucru asigura o legatura intre proprietarul dispozitivelor si dispozitive.
- Aceasta structura de baza a bazei de date si modul de sincronizare a microserviciilor asigura o gestionare eficienta a utilizatorilor si a dispozitivelor in cadrul sistemului distribuit. Ea permite, de asemenea, flexibilitate in ceea ce priveste crearea si gestionarea utilizatorilor si dispozitivelor, mentinand in acelasi timp o structura coerenta a datelor.

Tabela “EnergyConsumption”

- Această tabelă face parte din microserviciul dedicat gestionării consumului de energie.
- Tabela este destinată înregistrării consumului pentru fiecare dispozitiv, inclusiv identificatorul dispozitivului (device id), identificatorul de timp (timestamp id) și valoarea măsurată a consumului.

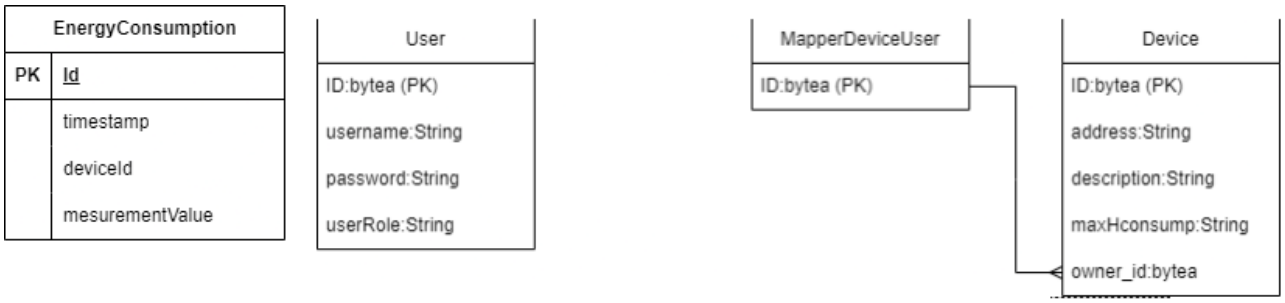


Figure 1 Schema bazei de date

Arhitecturea 3-Layer

Un model comun in dezvoltarea aplicatiilor, in special in contextul arhitecturii Model-View-Controller (MVC), presupune imparyirea aplicatiei in urmatoarele componente:

Model: Modelul reprezinta datele aplicatiei si logica de afaceri asociata. Acesta este responsabil pentru gestionarea datelor si ofera functionalitati pentru a citi, scrie si manipula aceste date. De obicei, modelul interactioneaza direct cu baza de date sau cu alte surse de date. Intr-un model pot fi incluse obiecte, clase sau structuri care definesc structura datelor.

Repository (Repo): Repository este o componenta responsabila pentru gestionarea accesului la date. Ea actioneaza ca un strat intermediar intre model si sursa de date (de exemplu, baza de date). Repository ofera metode pentru a citi si scrie datele in model si ascunde detalii specifice sursei de date. Acest strat de abstractizare face parte din principiul "Dependency Inversion", care ajuta la decuplarea modelului de implementarea reala a accesului la date.

Service: Serviciul este o componenta care contine logica de afaceri a aplicatiei. El foloseste modelele si repository-urile pentru a implementa functionalitatea aplicatiei. Serviciile ofera o interfata simplificata pentru controlere si gestionarea operatiilor complexe care implica mai multe obiecte de model sau operatiuni de date.

Controller: Controllerul este responsabil pentru gestionarea cererilor HTTP primite de la client si pentru directionarea acestor cereri catre serviciile adecvate. Controllerul primeste date de la client prin intermediul rutei si a cererii HTTP si returneaza raspunsuri adecvate, cum ar fi pagini web, date JSON sau alte tipuri de raspunsuri. El actioneaza ca un intermediar intre utilizator si servicii.

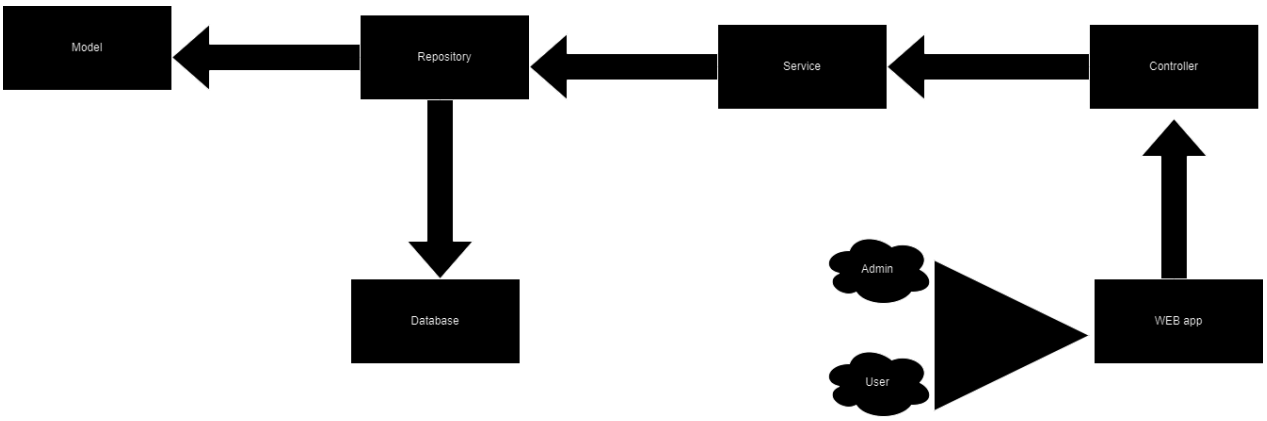


Figure 2 Web application architecture

In cadrul dezvoltarii frontend-ului aplicatiei, am ales sa folosesc framework-ul AngularJS pentru a crea o interfata utilizator moderna si reactiva. Acest framework a permis organizarea componentelor principale intr-un mod coerent si eficient, oferind utilizatorilor o experienta placuta si intuitiva. Iata o descriere mai detaliata a principalelor componente ale frontend-ului si a modului in care acestea sunt structurate:

- Pagina de Logare (Login-page): Aceasta este pagina de pornire unde utilizatorii Isi introduc datele de autentificare.
- Pagina de Utilizator (User-Page): Aici utilizatorii pot vizualiza lista dispozitivelor si se pot deconecta.
- Pagina de Administrator (Admin-Page): Pentru administratori, aceasta pagina ofera functionalitati avansate, inclusiv gestionarea dispozitivelor si a utilizatorilor.
- Componente Intermediare: Fiecare pagina poate include componente mici pentru afisarea datelor sau efectuarea operatiilor CRUD.
- Serviciu (Service): Serviciul gestioneaza apelurile catre backend si manipuleaza datele.
- Rutare (Routing): Sistemul de rutare asigura navigarea Intre pagini si aplicarea regulilor de securitate In functie de autentificare si rolul utilizatorului.

Aceasta structura eficienta permite o experienta de utilizare placuta si gestiunea eficienta a utilizatorilor si dispozitivelor In cadrul aplicatiei. AngularJS ofera un cadru puternic pentru dezvoltarea aplicatiilor web interactive si complexe.

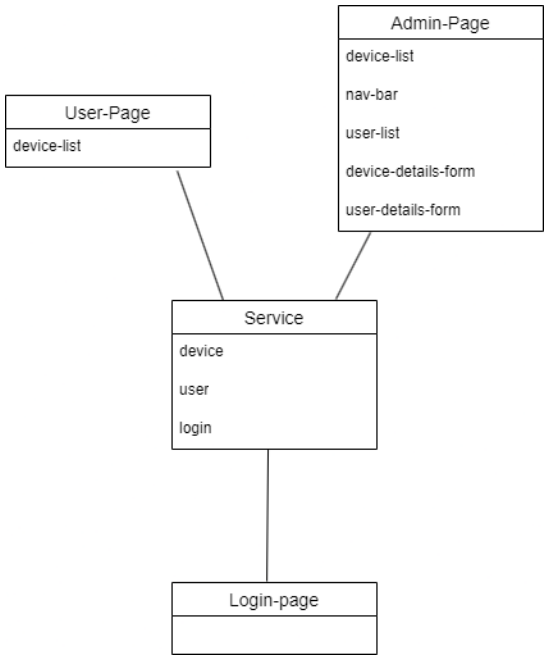


Figure 3 Main components Frontend

Am folosit platforma Docker care ruleaza containere permite impachetarea distribuirea si rularea aplicatiilor in servicii container software isolate. Un container este un pachet care contine aplicatia, bibliotecile, dependentele si configuratiile, toate intr-un mediu izolat.

Caracteristicile principale Docker:

- Izolare
- Portabilitate
- Eficienta
- Gestionarea versiunilor
- Orcherstrare

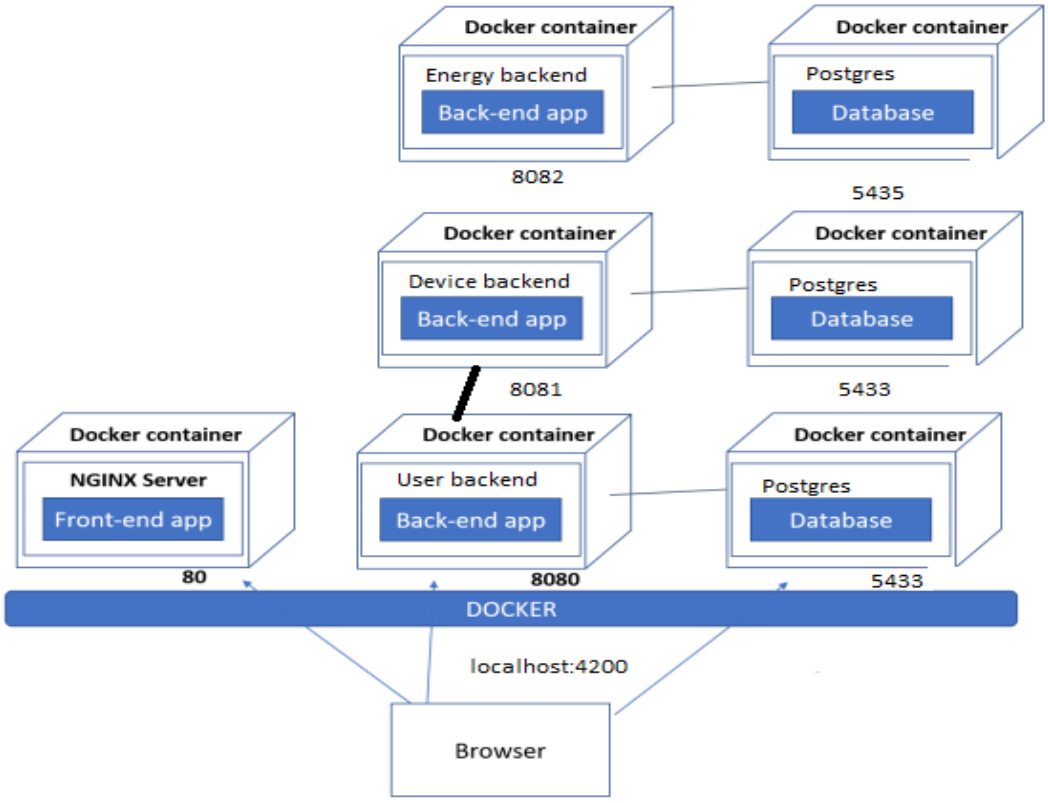
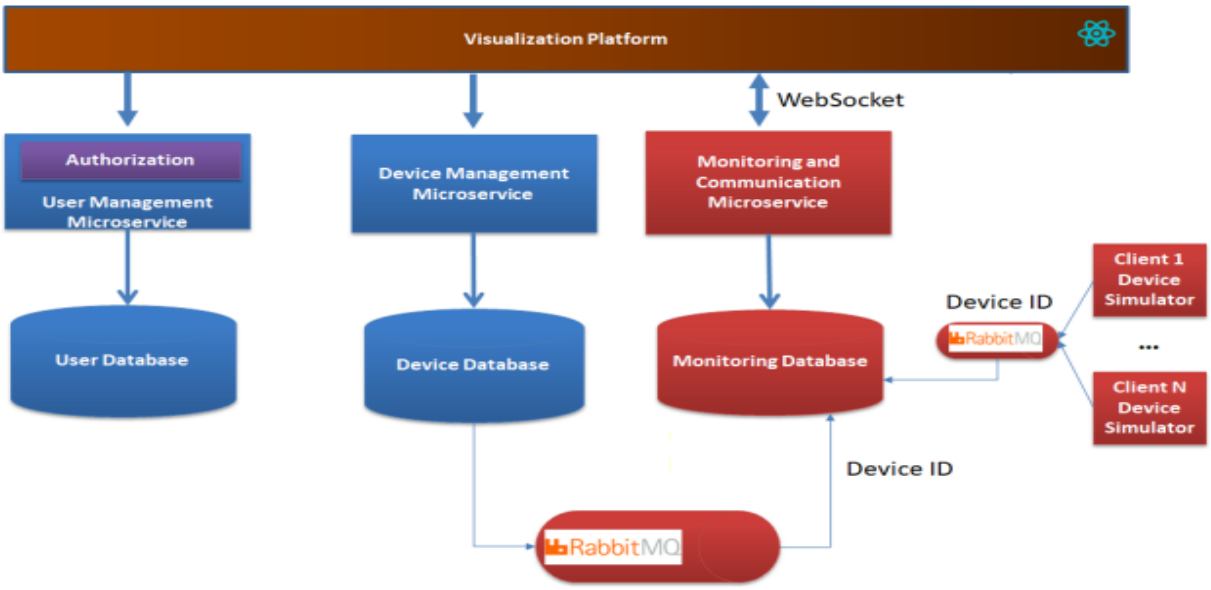


Figure 4 Schema containerelor Docker



Vom discuta despre partea cu rosu.
Comunicarea dintre frontend si microserviciul de gestionare al energiei(monitoring and communication) a fost realizata prin WebSockets.
WebSockets reprezinta un protocol de comunicare bidirectionala in timp real, implementat peste un singur canal de comunicare TCP. Acesta ofera posibilitatea unei conexiuni persistente intre un client si un server, faciliteaza schimbul rapid de date in ambele directii. Cand se conecteaza clientul se deschide un websocket si automat se transmit toate dispozitivele acestuia pentru a verifica consumul. Daca consumul depaseste valoarea maxima automat se va trimite o notificare.
Datele trimise la microserviciul de gestionare al energiei(monitoring and communication) sunt transmise printr-o coada de tip RabbitMQ care trimite date asincron. Si acest microserviciu mai comunica cu microserviciul de gestionare a dispozitivelor pentru operatiile de stergere.