



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

SISTEME DISTRIBUITE

Assignment 3

Web sockets and security

Bumbu Razvan-Vasile

Grupa 30644



CUPRINS

1. Cerinta
2. Tehnologii folosite
3. Solutia aleasa
4. Arhitectura sistemului
5. Diagrama de deployment
6. Concluzii
7. Bibliografie



1. Cerinta

Dezvoltati un microserviciu de chat si un component de autorizare pentru Sistemul de Management Energetic. Componenta de autorizare ar trebui sa ofere acces securizat al utilizatorilor la microserviciile sistemului. Microserviciul de chat ar trebui să permită comunicarea între utilizatori si administratorul sistemului, permitandu-le sa puna intrebari si sa primeasca raspunsuri.

2. Tehnologii folosite

Pentru a realiza partea de backend a proiectului am folosit Java Spring Boot, care este un cadru de dezvoltare open-source pentru Java, special conceput pentru crearea rapida si simpla a aplicatiilor Java. Acesta se bazeaza pe platforma Spring si ofera dezvoltatorilor o modalitate eficienta de a crea aplicatii robuste, scalabile si usor de gestionat. Cu Spring Boot, sunt furnizate seturi preconfigurate de dependente si solutii comune, eliminand nevoia de a configura manual multe aspecte ale aplicatiei. Acest lucru permite dezvoltatorilor sa se concentreze asupra dezvoltarii functionalitatii principale a aplicatiei, fara a fi nevoie sa petreaca timp prea mult pe configurari si setari. Spring Boot faciliteaza dezvoltarea aplicatiilor Java moderne si este folosit pe scara larga in industrie.

Pentru partea de frontend a proiectului am folosit React, care este o biblioteca JavaScript, dezvoltata si mentinuta de Facebook, open-source, utilizata pentru dezvoltarea interfetelor de utilizator interactive si eficiente. Cunoscuta pentru reactivitatea sa, React permite dezvoltatorilor sa creeze aplicatii web dinamice, prin construirea componentelor reutilizabile. Acesta utilizeaza o abordare de programare declarativa, care face gestionarea starii si actualizarea DOM-ului mai eficienta, oferind o experienta de utilizare mai fluida. React se integreaza bine cu alte tehnologii si este adesea utilizat impreuna cu biblioteci precum Redux sau React Router pentru a construi aplicatii web complexe. Datorita popularitatii sale si a sprijinului comunitatii, React ramane o alegere



populara pentru dezvoltarea frontend.

Bazele de date ale aplicatiei sunt mentinute folosind PostgreSQL. Cunoscut si sub numele de Postgres, este un sistem de gestionare a bazelor de date open-source. Acesta ofera o performanta excelenta, extensibilitate si suport pentru un numar mare de tipuri de date si functii. Postgres este utilizat pe scara larga in aplicatii web, enterprise si proiecte de date, datorita fiabilitatii, securitatii si flexibilitatii sale.

Partea de deploy a aplicatiei a fost realizata folosind Docker, o platforma de containerizare open-source care simplifica dezvoltarea, distributia si gestionarea aplicatiilor. Prin utilizarea containerelor, Docker permite ambalarea aplicatiilor si dependentelor lor intr-un mediu izolat si portabil, asigurand astfel consistenta si usurinta in implementare pe diverse platforme. Aceasta optimizeaza resursele si accelereaza procesele de dezvoltare si implementare a aplicatiilor. Docker este esential in dezvoltarea DevOps si in gestionarea aplicatiilor cloud-native.

WebSockets reprezinta o tehnologie de comunicare bidirectionala in timp real intre client si server pe internet. Protocolul WebSocket ofera o conexiune persistenta si eficienta, eliminand necesitatea cererilor repetitive. Acesta permite schimbul instantaneu de date intre aplicatii web si server, facilitand actualizari in timp real si interactivitate fara a depinde de cereri HTTP periodice. Prin intermediul unui canal de comunicare continuu, WebSockets optimizeaza performanta si reduce latenta, imbunatatind semnificativ experienta utilizatorului in aplicatiile web dinamice si colaborative.

JSON Web Tokens (JWT) reprezinta o metoda sigura de autentificare si transmitere a informatiilor intre parti in format JSON. Aceste tokene compacte si autentificabile furnizeaza o solutie eficienta pentru securitatea aplicatiilor web si serviciilor API. JWT contin informatii criptate si semnate digital, permitand verificarea integritatii si a sursei datelor. Ele sunt utilizate pentru autentificarea utilizatorilor si transmiterea autorizatiilor intre clienti si servere. Prin utilizarea algoritmilor de semnare, JWT asigura confidentialitatea si autenticitatea



informatiilor, facilitand astfel securitatea robusta in mediul digital.

3. Solutia aleasa

Pentru dezvoltarea aplicatiei, am preluat aplicatia dezvoltata la tema anterioara, careia i-am adaugat un microserviciu nou:

- **Message-microservice**

- Acesta este microserviciul care se ocupa cu consumarea evenimentelor aflate in cozile gestionate de RabbitMQ.
- Acest serviciu are o baza proprie de date unde se salveaza mesaje trimise de utilizatori atat intre ei, cat si pe grupuri. De asemenea, se salveaza si informatiile despre grupuri.
- Microserviciul comunica cu frontend-ul aplicatiei folosind Web Sockets.

Fiecare microserviciu este securizat cu ajutorul JWT Security. Astfel, fiecare endpoint va primi si un token care contine rolul utilizatorului criptat. In acest fel se poate verifica daca utilizatorului ii este permis sau nu sa efectueze o anumita operatie. In caz negative, acesta va fi redirectionat catre pagina de logare.

De asemenea, pentru realizarea comunicarii in timp real, dar si pentru generarea de notificari, am folosit Web Sockets.

4. Arhitectura sistemului

Arhitectura aplicatiei este una clasica. Aplicatia de React trimite HTTP requests (GET, POST, PUT, DELETE) catre backend. Acest lucru se face cu ajutorul bibliotecii axios, care faciliteaza efectuarea cererilor asincrone catre API-urile din backend. Datele sunt transmise sub forma de obiecte JSON sau parametri, iar raspunsurile sunt prelucrate pentru a actualiza starea componentelor React si a afisa datele sau rezultatele cererilor in interfata utilizatorului.



Pe partea de backend, exista 3 pachete pentru fiecare microserviciu:

- **Controller:** Responsabil pentru gestionarea cererilor HTTP, interactioneaza cu utilizatori si rutează datele catre servicii.
- **Service:** Furnizeaza logica de afaceri a aplicatiei, proceseaza cererile si gestioneaza operatiunile specifice.
- **Repository:** Gestionarea datelor si comunicarea cu baza de date, asigurand manipularea eficienta a entitatilor persistente.

Fiecare microserviciu are propria baza de date in care sunt stocate datele.

O observatie importanta este aceea ca in momentul in care se doreste efectuarea unei operatii de stergere/modificare a datelor pe unul dintre utilizatori, microserviciul care se ocupa cu operatiile pe acestia va crea un API call asupra microserviciului care se ocupa cu dispozitivele pentru a actualiza datele referitoare la maparea dintre acel utilizator si dispozitivele pe care acestea le detine.

Consumption-microservice consuma evenimente de pe cozile din RabbitMQ, in timp ce **device-microservice** produce evenimente, la fel ca si **reader-microservice**.

Message-microservice comunica cu frontend-ul aplicatiei prin Web Sockets, pentru a realiza comunicarea si generarea de notificari in timp real.

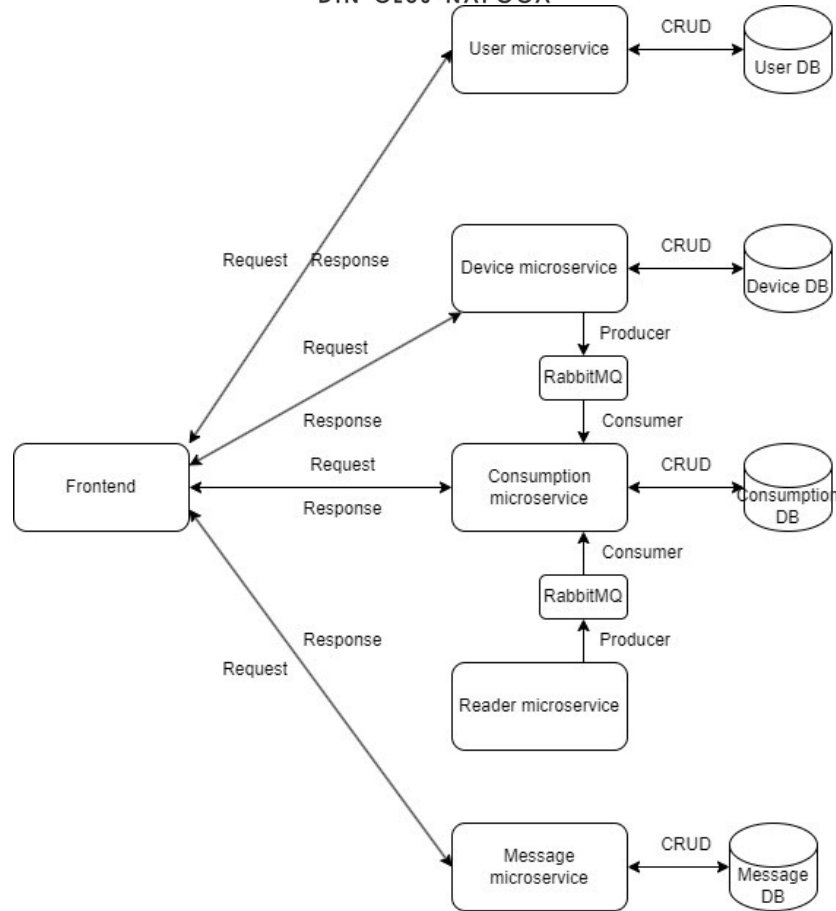


Fig1. Arhitectura aplicatiei

















































































5. Diagrama de deployment

In Docker, aplicatia contine 5 containere:

- **Energy-management** – partea de front-end a aplicatiei
- **Device-management-microservice** – microserviciul pentru dispozitive impreuna cu baza de date asociata
- **User-management-microservice** – microserviciul pentru utilizatori impreuna cu baza de date asociata
- **Consumption-microservice** – microserviciul pentru consum impreuna cu baza de date asociata
- **Rabbit-container** – serviciul RabbitMQ



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

<input type="checkbox"/>	<div></div>	consumption-microservice	Running (2/2)	0.22%	24 minutes ago				
<input type="checkbox"/>	<div></div>	application-consumption-1 6fd4aff4fed 	consumption-microservice-app Running	0.22%	8031:8030 	24 minutes ago			
<input type="checkbox"/>	<div></div>	database-consumption-1 0580bb31a99b 	postgres:13-alpine Running	0%	5435:5432 	24 minutes ago			
<input type="checkbox"/>	<div></div>	device-management-microservice	Running (2/2)	0.13%	22 minutes ago				
<input type="checkbox"/>	<div></div>	application-device-1 ea8befec46dd 	device-management-microserv Running	0.13%	8011:8010 	22 minutes ago			
<input type="checkbox"/>	<div></div>	database-device-1 47472bc2b871 	postgres:13-alpine Running	0%	5434:5432 	22 minutes ago			
<input type="checkbox"/>	<div></div>	enery-management	Running (1/1)	0.04%	22 minutes ago				
<input type="checkbox"/>	<div></div>	client-1 d3304a3a96fd 	enery-management-client Running	0.04%	3003:3000 	22 minutes ago			
<input type="checkbox"/>	<div></div>	message-microservice	Running (2/2)	0.15%	4 minutes ago				
<input type="checkbox"/>	<div></div>	application-message-1 5e73e8b4626c 	message-microservice-applica Running	0.15%	8041:8040 	4 minutes ago			
<input type="checkbox"/>	<div></div>	database-message-1 9bd7e74d9288 	postgres:13-alpine Running	0%	5436:5432 	4 minutes ago			
<input type="checkbox"/>	<div></div>	rabbit-container d97306bc6e1d 	rabbitmq:3-management Running	0.57%	15672:15672  Show all ports (2)	2 hours ago			
<input type="checkbox"/>	<div></div>	user-management-microservice	Running (2/2)	0.18%	26 minutes ago				
<input type="checkbox"/>	<div></div>	application-user-1 8ee6b9c2bc96 	user-management-microservic Running	0.18%	8021:8020 	26 minutes ago			
<input type="checkbox"/>	<div></div>	database-user-1 6ca6c2979e32 	postgres:13-alpine Running	0%	5433:5432 	26 minutes ago			

Showing 15 items

Fig2. Containerele din Docker

Pentru a putea fi realizata comunicarea intre containere, acestea au trebuit sa fie puse in aceeasi retea `energy_network`. Astfel, fiecare container va avea asociata o subretea cu un IP prin intermediul caruia poate sa realizeze comunicarea cu celelalte containere din retea.

6. Concluzii

In concluzie, explorand tehnologiile Java, Spring Boot, React, Docker si PostgreSQL, am dobandit cunostinte valoroase in dezvoltarea si gestionarea aplicatiilor software. Am invatat ca Java este un limbaj versatil, Spring Boot simplifica dezvoltarea rapida, React ofera interfete interactive, Docker permite containerizarea eficienta, iar PostgreSQL gestioneaza datele cu eficienta. De asemenea, RabbitMQ este esențial pentru comunicarea eficientă în aplicații



distribuite, oferind un sistem de cozi de mesaje pentru transmiterea asincronă a datelor, facilitând scalabilitatea și fiabilitatea în dezvoltare. JSON Web Tokens (JWT) reprezintă o soluție eficientă pentru securitatea autentificării și autorizării în aplicațiile web, oferind o transmitere sigură a informațiilor între părți. Împreună cu WebSockets, această tehnologie optimizează comunicarea în timp real, facilitând schimbul securizat de date între clienți și servere în medii web dinamice și colaborative.

Aceste tehnologii reprezintă o bază solidă pentru proiecte software de succes. Am înțeles importanța adaptării la cerințele pieței și am învățat cum să construiesc, să implementez și să gestionez aplicații scalabile, asigurând securitatea și performanța acestora. Această învățare continuă este crucială într-o industrie IT în evoluție rapidă și plină de oportunități.

7. Bibliografie

- <https://dsrl.eu/courses/sd/>
- <https://luizcostatech.medium.com/how-to-dockerize-spring-boot-react-apps-1a4aea1acc44>
- <https://tecadmin.net/dockerize-react-app/>
- <https://www.youtube.com/watch?v=XDlGwyVfSMA>
- <https://www.youtube.com/watch?v=9Zv-hbxlRTc&list=WL&index=28&t=2185s>
- <https://www.youtube.com/watch?v=0--LI3WHMTQ>
- <https://www.youtube.com/watch?v=KxqIJblhzfi>