

Stecko Daiana

Grupa: 30242

DISTRIBUTED SYSTEMS

Assignment 3

Chat System for Client Support

**Curpins**

**1.**Tema

**2.**Functionalitate

**3.**Instrumente utilizate

**4.**Design-ul Arhitectural

Diagrama de cazuri

Diagrama de deployment

Diagrama de clase

**5.**Diagrama de model

**6.**Dezvoltari ulterioare

**7.**Concluzie

**8.**Bibliografie

**1.Tema**

Cerința pentru acest proiect a fost să realizăm o aplicație de tipul client-server, in care sa implementam un chat. Cu ajutorul acestui chat persoanele vor putea comunica intre el. Adică se vor putea trimite mesaje de la administrator la client, dar si invers. Mesajele vor fi trimise asincron către administrator, odată ce ii apare un mesaj, acesta va putea trimite un răspuns la acel mesaj. Mesajele pot fi trimise înainte si înapoi intre client si administrator in timpul sesiunii de chat.

Administratorul va putea discuta cu mai mulți clienți in același timp.

De asemenea, se mai cere si o sesiune de notificări, adică o notificare este afișata atunci când celălalt utilizator citește mesajul.

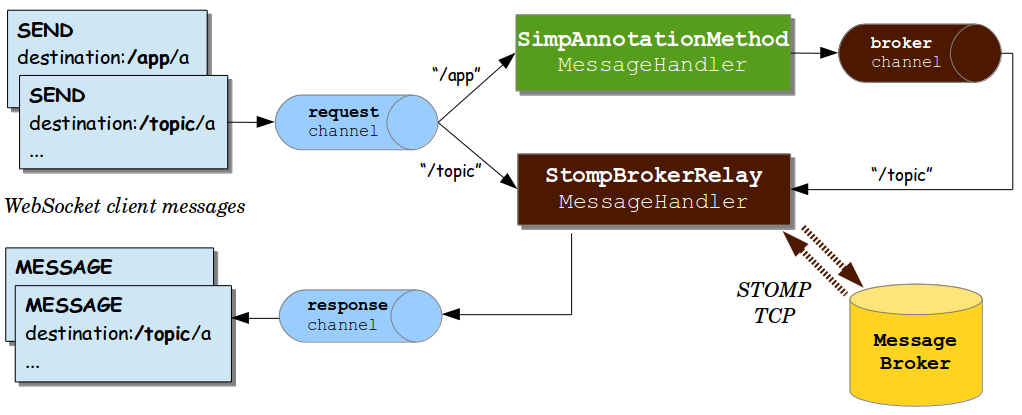
A mai fost necesara si adăugarea unei afișări pentru utilizator in momentul in care tastează cealaltă persoana acesta sa fie anunțat de aceasta acțiune.

**2.Functionalitate**

Aceasta aplicație presupune dezvoltarea unei aplicații ce conține o parte de chat. Pentru început utilizatorii vor putea acesta opțiune de chat din bara de navigare. După care își vor introduce numele si se vor conecta la chat. Persoanele vor fi conectate fiecare pe un browser separat. Acest chat are funcționalitate de un grup chat, însă are si posibilitatea de a face un chat privat, doar intre doua persoane.

Datele nu sunt salvate in baza de date, iar atunci când vor da reload se vor șterge.

Pentru realizarea simulării din aceasta temă, am deschis u browser, unde m-am conectat, după care am mai deschis încă o fereastra pentru a putea trimite mesaje intre mai multe persoane. Pe prima fereastra îmi va apărea si persoana care a intrat a doua. Am un buton de chat grup, unde pot trimite mesaje si pot primi mesajele de la celelalte persoane de pe celelalte browser, Deasemena, exista posibilitatea sa selectez o persoana din lista ce ii înregistrată si sa-i trimit un mesaj in privat, iar aceasta îmi va putea răspunde tot in privat.



**3.Instrumente utilizate**

La realizarea backend-ul m-am folosit de Spring Boot-Intellij IDEA. Am mai lucrat in trecut cu acest program, nu a fost ceva foarte diferit. Pentru frontend am utilizat VisualStudio Code cu Reactjs. Nu am mai utilizat foarte mult in tecut, a fost ceva mai nou, din acest motiv m-am confruntat cu diferite probleme, neștiind inițial cum funcționează. Insă in final am reusit să mă familearizez si lucrurile au decurs bine.

Am folosit ca baza de date postgres.

Pentru implementare am folosit stompjs, care este o biblioteca ce ofera STOMP prin websocket pentru browsere web sau alte medii bazate pe JavaScript si am integrat conexiunea in tema trecuta pentru a putea avea acces la baza de date si la toate funcționalitățile de data trecuta si mai exact de aplicatia de front si de back.

Pe partea de backend m-am folosit de websocketuri pentru a trimite pe front si pentru legături. In ceea ce privește partea de front, după cum am precizat si mai sus, m-am folosit de stomp pentru socket.

**4.Design-ul Arhitectural**

O arhitectură conceptuală descrie un stil arhitectural stratificat. Fiecare strat într-un stil arhitectural stratificat este un pachet de software care are o interfață bine definită și câteva dependențe bine-cunoscute cu alte straturi. Fiecare strat implementează o funcție tehnică în cadrul serviciului. De exemplu, un strat de acces la date este responsabil pentru încapsulare mijloacele tehnice pentru accesarea unei baze de date. Toate cererile de acces la date către o bază de date trec prin stratul de acces la date pentru acea bază de date.

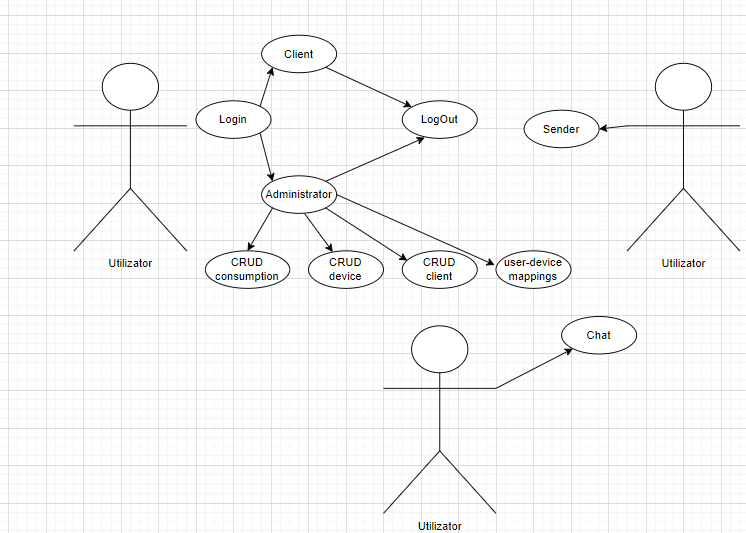
Sunt descrise ideile designului la nivel general, fiecare strat îndeplinind un anumit rol.

SockJS este o biblioteca JavaScript de browser care ofera un obiect asemanator WebSocket SockJS va ofera un API Javascript coerent, incruciat, care creeaza un canal de comunicare interdomeniu, full duplex, cu latenta scazuta, intre browser si serverul web.

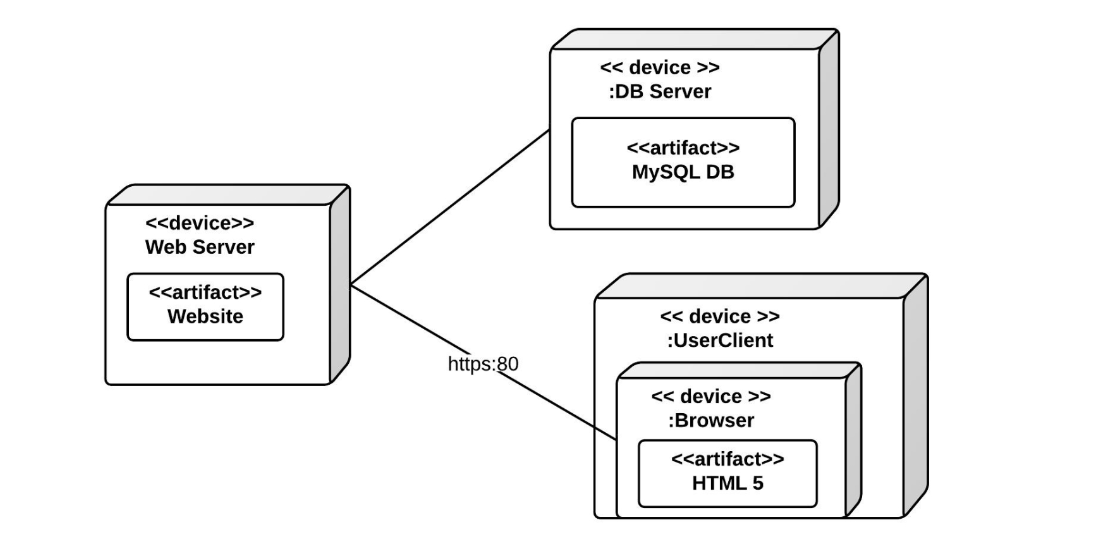
O imagine care conține text, carte de vizită, captură de ecran, miniatură

Descriere generată automat

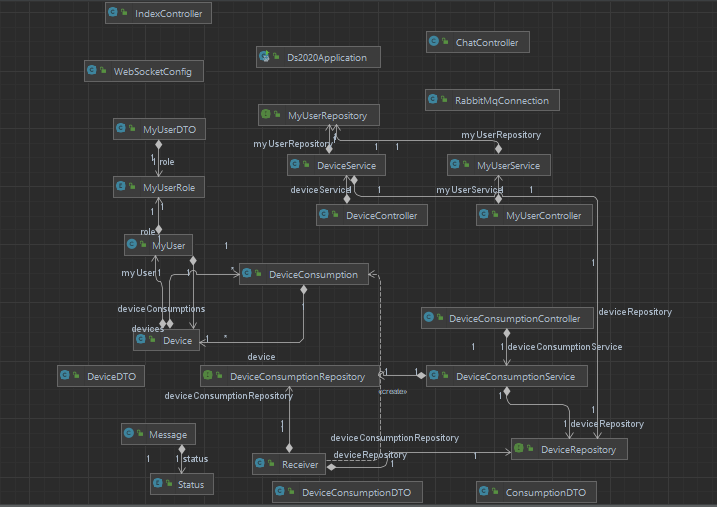
**Diagrama de cazuri**



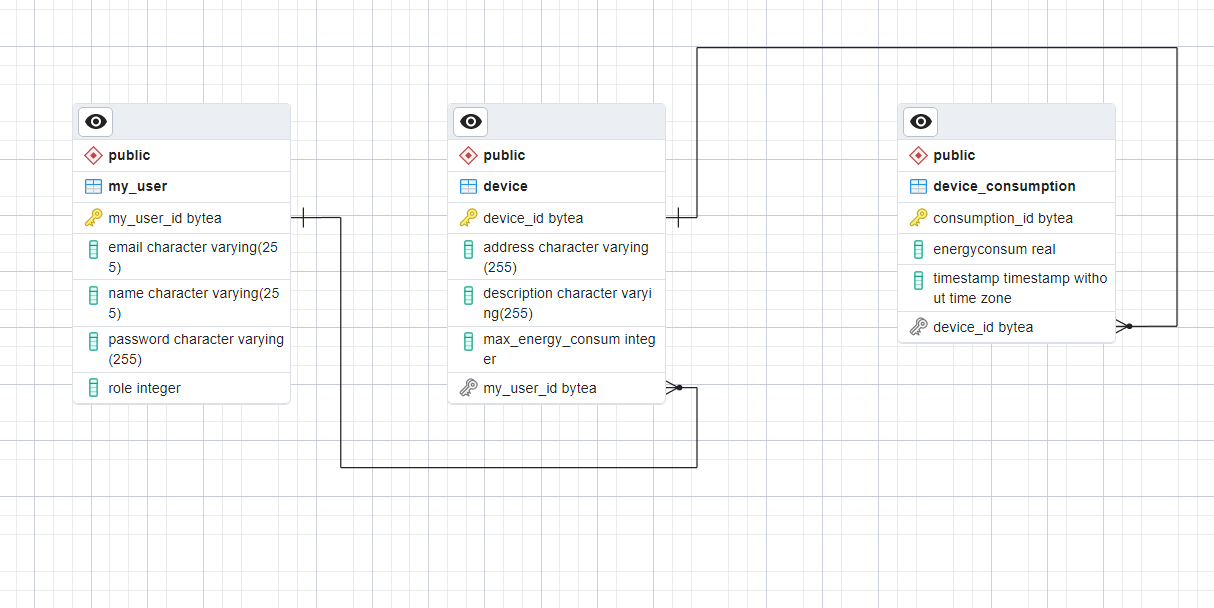
**Diagrama de deployment**

****

**Diagrama de clase**

****

**5.Diagrama de model**

****

**6.Dezvoltari ulterioare**

Cum nu am reușit să implementez toate cerințele, consider că pe viitor aș putea să mă ocup și de partea de notificare, adica unser-ul sa fie notificat in momentul in care celalat user a vazut mesajul sau. Deasemena as putea sa ma ocup si de partea in care un utilizator incepe sa scrie ceva sa se afiseze un mesaj care sa anunte acest lucru.

**7.Concluzie**

Acest proiect m-a ajutat să învăț foarte multe lucruri legate de transmiterea datelor mai ușor, până acum nu am pus foarte mult accent pe această parte. Însă odată cu proiectul acesta pot spune că am reușit să dezvolt această parte. Deci pot spune că am descoperit și odată și învățat foarte multe lucruri noi care mă vor ajuta cu siguranța pe viitor, consider că această materie ne ajută să facem mai multe lucruri deodată, ceea ce este foarte bine și în acest mod putem vedea ceea ce ne place cu adevărat.

**8.Bibliografie**

<https://umes4ever.medium.com/real-time-application-using-websocket-spring-boot-java-react-js-flutter-eb87fe95f94f>

<https://stomp-js.github.io/>

<https://www.youtube.com/watch?v=o_IjEDAuo8Y&t=4s>

<https://spring.io/guides/gs/messaging-stomp-websocket/>

**W3schools**

**StackOverFlow**