Documentatie ChatApp

Întocmit de: **Torz Iulian Marius**
Pentru disciplina: Proiectarea Aplicatiilor Web

Data predării: 13/01/2025

Cuprins

Capitolul 1. Introducere

1.1. Tema proiectului

Acest proiect reprezintă un sistem de mesagerie online, dezvoltat utilizând Spring Boot pentru partea de backend și un front-end simplu bazat pe tehnologii web tradiționale (HTML, CSS și JavaScript) implementat prin Thymeleaf. Scopul proiectului este de a oferi o platformă intuitivă și securizată, prin care utilizatorii pot comunica între ei în timp real, având în același timp acces la o autentificare sigură și sesiuni persistente.

1.2. Obiective

a) Autentificare și autorizare - sistemul permite utilizatorilor să se înregistreze, să se autentifice și să își acceseze conversațiile într-un mod securizat, respectând cele mai bune practici de protecție a datelor. Stocarea parolei in baza de date se face prin cryptarea cu algoritmul de encriptare Blowfish folosind de BCrypt
b) Persistența Mesajelor - mesajele trimise între utilizatori sunt stocate într-o bază de date relationala, oferind acces la istoricul conversațiilor
c) Gestionarea sesiunilor - sistemul implementează sesiuni persistente prin utilizarea cookie-urilor, astfel încât utilizatorii să rămână conectați chiar dacă închid aplicația sau browserul.
d) Interfață simplă - aplicația oferă o interfață ușor de utilizat, proiectată pentru o experiență minimă de configurare și navigare intuitivă.

1.3. MoSCoW list

1.3.1. Must

- Conversatii 1-1
- Conversatii de grup (mai mult de 2 utilizatori)
- Discutie generala (toti utilizatorii)

1.3.2. Should

- Afisare lista utilizatori
- Suport pentru fisiere/poze/documente

1.3.3. Could

- Sistem de reactionare la mesaje
- Apeluri video/audio

1.3.4. Won't

- Mesaje vocale
- Editare mesaje
- Customizare/avatare profile

Capitolul 2. Tehnologi utilizate

În ceea ce privește tehnologia, proiectul este construit pe baza arhitecturii MVC (Model-View-Controller), asigurând o separare clară între diferitele părți ale aplicației. Partea de backend este dezvoltată folosind framework-ul Spring Boot, ales pentru flexibilitatea și simplitatea sa, precum și pentru integrarea nativă cu alte componente Spring, cum ar fi Spring Data JPA și Spring Security.

Spring Data JPA facilitează interacțiunea cu baza de date relațională, în acest caz MySQL, prin intermediul unei soluții ORM, și anume Hibernate. Acesta asigură o manipulare elegantă a datelor și oferă posibilitatea mapării obiectelor Java la tabelele bazei de date fără a necesita scrierea excesivă de SQL. Partea de securitate este gestionată de Spring Security, care oferă o soluție robustă pentru autentificare și autorizare. Totodată, parolele utilizatorilor sunt criptate folosind algoritmul BCrypt, ceea ce asigură că acestea sunt stocate într-un mod sigur și că datele confidențiale sunt protejate. Un utilizator care nu este autentificat nu poate sa acceseze nici o pagina inafara de cea de register si login, asigurand un grad ridicat de protectie impotriva utilizatorilor cu intentii rele.

Pentru gestionarea dependintelor si al build-urilor proiectului s-a folosit maven, prin care s-a importat librarii precum lombok pentru generarea claselor POJO (plain old java object) si eficientizarea crearii setterilor , getterilor si a constructorilor.

Pe partea de front-end, interfața utilizatorului este construită cu ajutorul HTML și CSS, oferind un aspect simplu, dar eficient, care facilitează navigarea utilizatorilor. JavaScript este utilizat pentru a aduce funcționalitate dinamică aplicației, permițând gestionarea datelor în timp real și interacțiunea directă cu API-urile backend.

Comunicarea in timp real intre utilizatori a fost facilitata de catre Websockets, care reprezintă o tehnologie de comunicație în timp real care permite o conexiune bidirecțională persistentă între utilizatori. Eficienta sta in livrarea instanta a datelor si a mesajelor conexiunea ramanand deschisa pana la inchiderea conversatiei, prin Websockets se pot gestiona multiple conversatii in acelasi timp permitand scalabilitate ridicata. Pe partea de front-end Websockets-urile sunt implementate prin STOMP (Simple (or Streaming) Text Oriented Messaging Protocol) un protocol de mesagerie, fiind descris ca TCP pentru Web si se bazeaza pe comenzi precum CONNECT, SEND, SUBSCRIBE.

Baza de data aleasa a fost MySQL, fiind o baza de data relationala aduca familiaritate si stabilitate proiectului. Conexiunea cu baza de date se face prin aplication.properties si este gestionata de catre spring.

Sesiunea web este asigurata printr-un cookie care este salvat in browserul utilizatorului, acesta asigura autentificarea instantanee chiar daca utilizatorul opreste dispozitivul, inchide browserul sau aplicatia se opreste.

Capitolul 3. Prezentare

3.1. Structura

Arhitectura proiectului este gândită în jurul separării funcționalităților în trei componente esențiale: modelul, controlerul și vizualizarea. Modelul reprezintă partea de backend care interacționează cu baza de date și definește structurile de date esențiale, precum entitățile de utilizator și mesaj. Partea de controler gestionează logica aplicației, expune endpoint-urile REST și răspunde cererilor HTTP venite de la utilizatori. Vizualizarea, sau partea de interfață, este responsabilă pentru afișarea datelor și interacțiunea utilizatorilor cu aplicația.

În ceea ce privește fluxul de date, acesta începe cu utilizatorul, care trimite o cerere către server, fie pentru autentificare, fie pentru trimiterea unui mesaj. Cererea este procesată de controler, care verifică autentificarea și autorizarea, iar în cazul unei cereri valide, interacționează cu baza de date prin intermediul JPA și Hibernate. Datele necesare sunt apoi returnate către utilizator printr-un răspuns, iar interfața este actualizată corespunzător.

DTO (Data Transfer Object) este folosit în arhitectura aplicației tale pentru a simplifica și standardiza transferul de date între diferite componente, precum controller-ele și alte servicii sau module. S-a folosit in transmiterea mesajelor si in transmiterea datelor utilizatorilor.

3.2. Functionare

În utilizarea aplicației, un utilizator începe prin a accesa pagina de înregistrare, unde poate crea un cont prin completarea unor câmpuri precum nume de utilizator și parolă. După înregistrare, utilizatorul se poate autentifica pe pagina de login, iar dacă autentificarea este reușită, acesta este redirecționat către interfața principală de chat. Interfața permite trimiterea și primirea de mesaje, care sunt stocate în baza de date pentru a asigura persistența. Din acest punct utilizatorul poate sa creeze o conversatie noua, sau sa aleaga din conversatiile sale create anterior. Daca utilizatorul selecteaza un utilizator cu care are deja o conversatie creeata , nu se fa afisa o eroare care spune ca exista deja, el va fi redirectionat in conversatia existenta. Procesul de gasire a conversatiilor este urmatorul: se cauta id-urile utilizatorilor, se numara numarul participantilor in conversatie si se cauta daca exista aceste id-uri deja intr-o conversatie.

Sistemul este capabil să gestioneze istoricul conversațiilor, iar utilizatorii pot vizualiza mesaje trimise anterior într-un format cronologic. Toate mesajele sunt asociate atât expeditorului, cât și destinatarului, iar timestamp-urile sunt salvate pentru fiecare mesaj.

Mesajele sunt trimite prin websockets, care se ocupa si de actualizarea in timp real, prin protocolul de subscribe, odata ce utilizatorul face click pe o conversatia se deschide un websocket cu subscribe, daca face click pe alta conversatie, websocket-ul initial se inchide, pentru a evita mesajele multiple, daca nu s-ar inchide websocket-ul si s-ar reveni la conversatie mesajul ar fi trimis de mai multe ori.

3.3. Arhitectura

Aplicația urmează arhitectura MVC (Model-View-Controller), oferind o separare clară a logicii aplicației, interacțiunii cu utilizatorii și gestionării datelor.

Modelul este reprezentat de structurile de date precum User si Message, este gestionat prin intermediul hibernate si JPA, si schematic poate fi reprezentat in felul din figura 1.

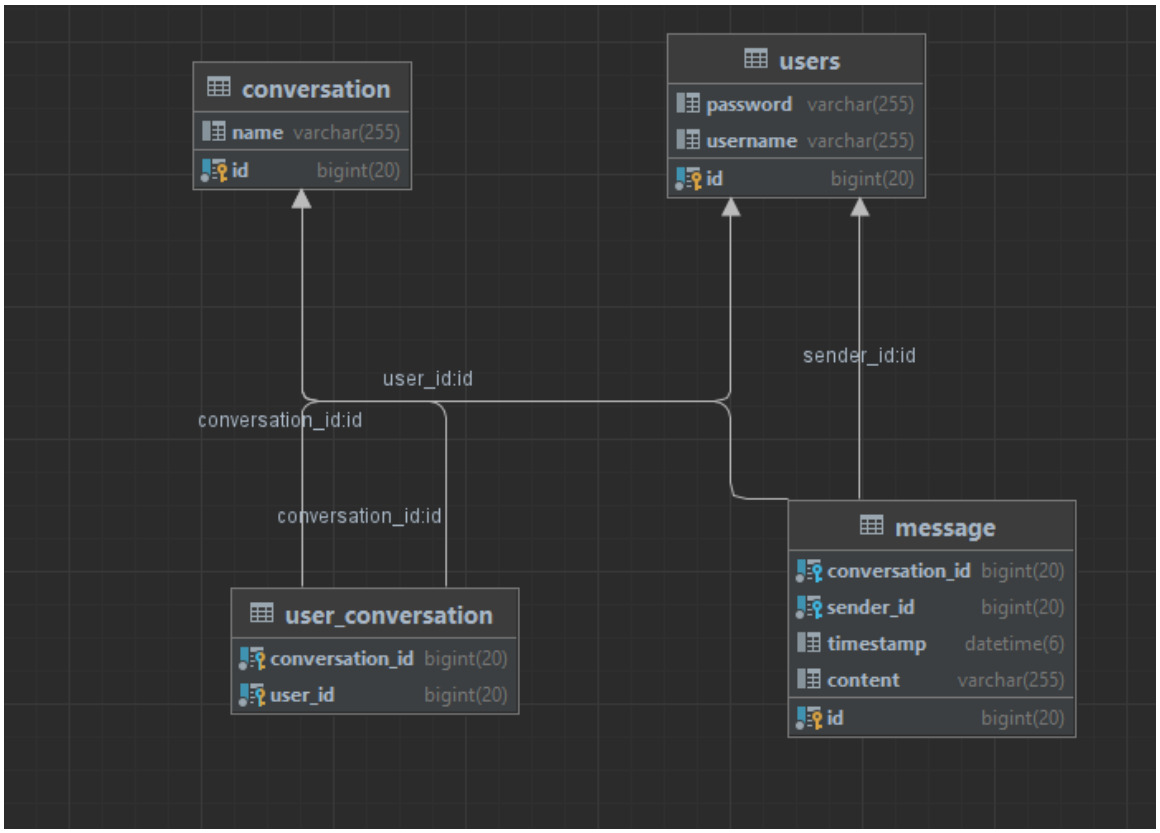

Figura 1. Modelul Datelor

View-ul consta in paginile HTML care sunt servite de catre backend prin endpoint-uri REST, cosmetizarea fiind facuta cu CSS si reactivitatea fiind oferita de Javascript.

Controller-ul este responsabil de gestionarea cererilor HTTP si conectarea utilizatorilor la logica si modelul de date. Endpoint-urile expuse sunt pentru autentificare, creearea conversatiilor, si preluarea mesajelor.

3.3.1. Endpoints

Endpoint	HTTP Method	Controller	Description
/api/conversations/{conversationId}/messages	GET	ChatController	Preia toate mesajele pentru o anumită conversație identificată de conversationId.
/api/conversations/{conversationId}/messages	POST	ChatController	Trimite un mesaj la o anumită conversație. Corpul solicitării conține conținutul mesajului, iar conversationId identifică conversația.
/api/conversations/{conversationId}	GET	ChatController	Preia detaliile unei anumite conversații prin conversationId-ul acesteia.
/api/users/by-username/{username}	GET	ChatController	Preia ID-ul utilizatorului asociat cu un anumit nume de utilizator. Returnează o stare 404 dacă utilizatorul nu este găsit.
/users/{userId}/conversations	GET	ChatController	Preia toate conversațiile pentru un anumit utilizator identificat prin ID-ul de utilizator.
/api/current-username	GET	UserController	Returnează numele de utilizator al utilizatorului autentificat curent pe baza contextului de securitate.
/api/users	GET	UserController	Preia o listă a tuturor utilizatorilor din sistem.
/conversations	POST	ChatController	Creează o conversație nouă cu un set de participanți specificat în corpul solicitării. Asigură cel puțin doi participanți și evită conversațiile duplicate prin verificarea celor existente cu aceiași participanți.
/login	GET	LoginController	Servește pagina de login

/register	GET	RegistrationController	Servește pagina de înregistrare unde utilizatorii pot crea conturi noi.
/register	POST	RegistrationController	Se ocupă de înregistrarea utilizatorilor. Validează datele indexează parola și salvează, noul utilizator în baza de date. Redirecționează către pagina de autentificare după succes sau returnează erori la pagina de înregistrare.
/conversation/{conversationId}/send-message	Websocket	WebSocketChatController	Gestionează mesajele WebSocket trimise la o anumită conversație (conversationId). Transmite mesajul tuturor abonaților subiectului WebSocket al conversației.
/topic/conversation/{conversationId}	Websocket	WebSocketChatController	Un subiect de abonament WebSocket în care clienții pot asculta mesaje noi într-o conversație.

Capitolul 4. Concluzii

În concluzie, acest proiect oferă o soluție completă pentru un sistem de mesagerie online securizat și eficient. Alegerea tehnologiilor utilizate, precum și implementarea arhitecturii MVC, au asigurat realizarea unei aplicații scalabile și ușor de extins. Funcționalitățile de securitate și gestionarea sesiunilor, combinate cu o interfață simplă, oferă utilizatorilor o experiență plăcută și sigură. Proiectul poate servi drept bază pentru adăugarea de funcționalități avansate, cum ar fi notificările în timp real prin WebSocket sau integrarea cu servicii de cloud pentru o mai bună scalabilitate.