MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE INGINERIE

CENTRUL UNIVERSITAR NORD DIN BAIA MARE

PROIECTAREA APLICATILOR WEB

PROIECT – WEB CHAT

Student: **Bancoș Andrei Marius**

Specializare: **Calculatoare – Anul 4**

2025

CUPRINS

Capitolul 1. Introducere

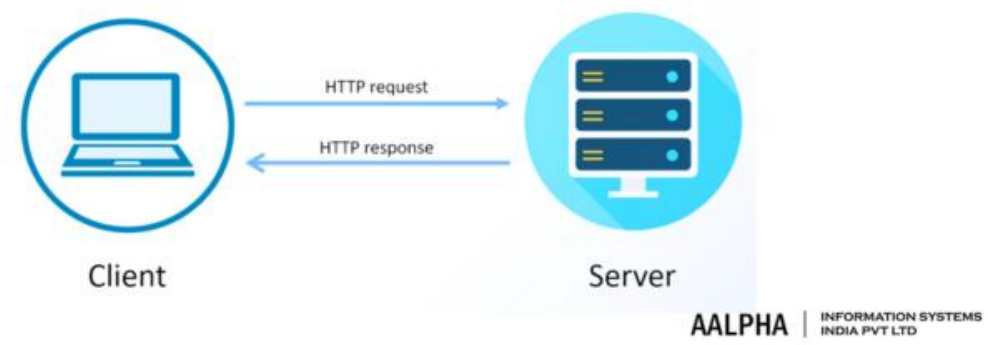
În cadrul acestui proiect, ne propunem să dezvoltăm o aplicație de web chat care va facilita comunicarea eficientă între utilizatori. Aplicația va oferi o interfață intuitivă, care va permite mesagerie instantanee. Cu un accent deosebit pe securitate și confidențialitate, vom implementa măsuri riguroase pentru a proteja datele utilizatorilor. Scopul este de a crea un mediu de comunicare prietenos și accesibil, adaptat nevoilor diverse ale utilizatorilor, care să încurajeze interacțiunea și colaborarea.

Capitolul 2. Arhitectură și tehnologi

În acest capitol o sa discutăm despre arhitectura aplicației și tehnologiile utilizate pentru realizarea acesteia. Proiectul este compus din 2 componente care formează aplicația front-end și back-end.

2.1 Arhitectura Client-Server

Arhitectura client-server este un model fundamental în dezvoltarea aplicațiilor web moderne, care separă responsabilitățile între partea de client (front-end) și cea de server (back-end). În cadrul proiectului nostru, utilizăm Angular pentru front-end și Spring Boot pentru back-end, creând astfel o aplicație web performantă și scalabilă.



Figură 1 Arhitectura Client-Server (Pawar, 2024)

2.2 Front-end

Pentru dezvoltarea front-end-ului aplicației de web chat, vom utiliza Angular ca tehnologie principală. Angular oferă un cadru robust pentru crearea de aplicații web dinamice, permițând o experiență de utilizator fluidă și interactivă.

Tehnologii:

- **HTML5:** Folosit pentru structura paginilor web, asigurând o semantică clară și accesibilitate.
- **CSS3:** Utilizat pentru stilizarea aplicației, cu suport pentru responsive design, pentru a optimiza experiența pe diferite dispozitive.
- **TypeScript:** Limbajul de programare utilizat de Angular, care adaugă tipuri statice și facilități avansate de dezvoltare.
- **RxJS:** Bibliotecă pentru gestionarea programării reactive, esențială pentru manipularea fluxurilor de date și evenimente.
- **Angular Material:** Bibliotecă de componente UI care oferă elemente de design moderne și consistente, contribuind la crearea unei interfețe prietenoase.
- **NgRx:** Bibliotecă pentru gestionarea stării aplicației, care facilitează gestionarea eficientă a datelor și interacțiunilor în aplicație.
- **Tailwindcss:** Framework de styling utilitar care permite personalizarea rapidă a designului, oferind clase utilitare flexibile pentru o dezvoltare eficientă a interfeței.

2.3 Back-end

Pentru dezvoltarea back-end-ului aplicației de web chat, am ales Spring Boot ca tehnologie principală. Spring Boot facilitează crearea de aplicații Java autonome și productibile, reducând complexitatea configurării și accelerând procesul de dezvoltare.

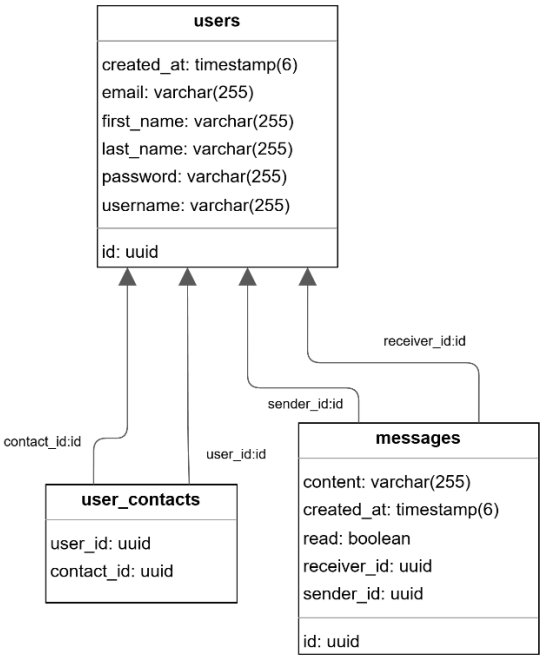
Tehnologii:

- **Spring Boot Starter Web:** Această componentă oferă toate funcționalitățile necesare pentru dezvoltarea aplicațiilor web, inclusiv configurarea automată a serverului web, facilitând astfel expunerea de API-uri RESTful.
- **Spring Boot Starter Test:** Permite dezvoltatorilor să realizeze teste unitare și de integrare, asigurând calitatea codului prin instrumente dedicate de testare.
- PostgreSQL: O bază de date relațională robustă, utilizată în mediul de producție, care asigură stocarea eficientă a datelor aplicației.
- **Spring Data JPA:** Această extensie a Spring simplifică gestionarea interacțiunilor cu baza de date, oferind un set de API-uri de nivel înalt pentru operațiunile CRUD și mapparea entităților.
- **Spring Boot Starter Actuator:** Oferă endpoint-uri utile pentru monitorizarea și gestionarea aplicației, permițând vizualizarea stării aplicației și a metricilor de performanță.

- **MapStruct:** Un instrument eficient pentru mapparea între obiecte, care simplifică conversiile de date prin generarea automată a codului, reducând astfel sarcinile manuale.
- **Spring Boot Starter Validation:** Această componentă integrează suportul pentru validarea datelor, asigurând că informațiile primite de la utilizatori respectă regulile și constrângerile definite.
- **Spring Boot Starter Security:** Oferă funcționalități de securitate esențiale, inclusiv autentificare și autorizare, protejând astfel resursele aplicației.
- **Spring Security Test:** Sprijină testarea funcționalităților de securitate, permițând verificarea eficienței măsurilor de protecție implementate.
- **Spring Boot DevTools:** Facilitează dezvoltarea rapidă prin reîncărcarea automată a aplicației, economisind timp în timpul procesului de dezvoltare.
- **OAuth2 Resource Server:** Oferă suport pentru gestionarea resurselor securizate prin protocolul OAuth2, esențial pentru aplicațiile care necesită autentificare externă.
- **OAuth2 Client:** Permite integrarea cu furnizorii de identitate OAuth2, facilitând autentificarea utilizatorilor prin diverse servicii externe.
- **Springdoc OpenAPI:** Această bibliotecă generează documentația API conform standardului OpenAPI, oferind o interfață UI intuitivă pentru explorarea serviciilor disponibile.

2.4 Baza de date

Aplicația utilizează PostgreSQL, un sistem de management al bazelor de date relaționale open-source, cunoscut pentru performanța, scalabilitatea și conformitatea sa cu standardele SQL.



Figură 2 Diagramă baza de date

Descrierea tabelelor din baza de date:

1. Tabelul users

Acest tabel stochează informațiile despre utilizatori, inclusiv:

- **id:** Identificator unic al utilizatorului.
- **created_at:** Data și ora la care a fost creat contul.
- **email:** Adresa de email a utilizatorului.
- **first_name:** Prenumele utilizatorului.
- **last_name:** Numele de familie al utilizatorului.
- **password**: Parola utilizatorului, stocată sub formă criptată.
- **username**: Alias folosit în aplicație.

Reprezintă baza pentru gestionarea utilizatorilor.

2. Tabelul user_contacts

Acest tabel gestionează relațiile dintre utilizatori, stocând:

- **user_id:** Referință către un utilizator.
- **contact_id:** Referință către alt utilizator aflat în lista de contacte.

Este folosit pentru a reprezenta lista de contacte a fiecărui utilizator.

3. Tabelul messages

Acest tabel înregistrează mesajele trimise între utilizatori, conținând:

- **id:** Identificator unic al mesajului.
- **content:** Textul mesajului.
- **created_at:** Data și ora la care mesajul a fost trimis.
- **read:** Indicator dacă mesajul a fost citit sau nu.
- **receiver_id:** Referință către utilizatorul care a primit mesajul.
- **sender_id:** Referință către utilizatorul care a trimis mesajul.

Acesta păstrează un istoric complet al conversațiilor din aplicație.

Legături între tabele

Tabelul users este conectat la **user_contacts** pentru gestionarea relațiilor dintre utilizatori și contactele lor.

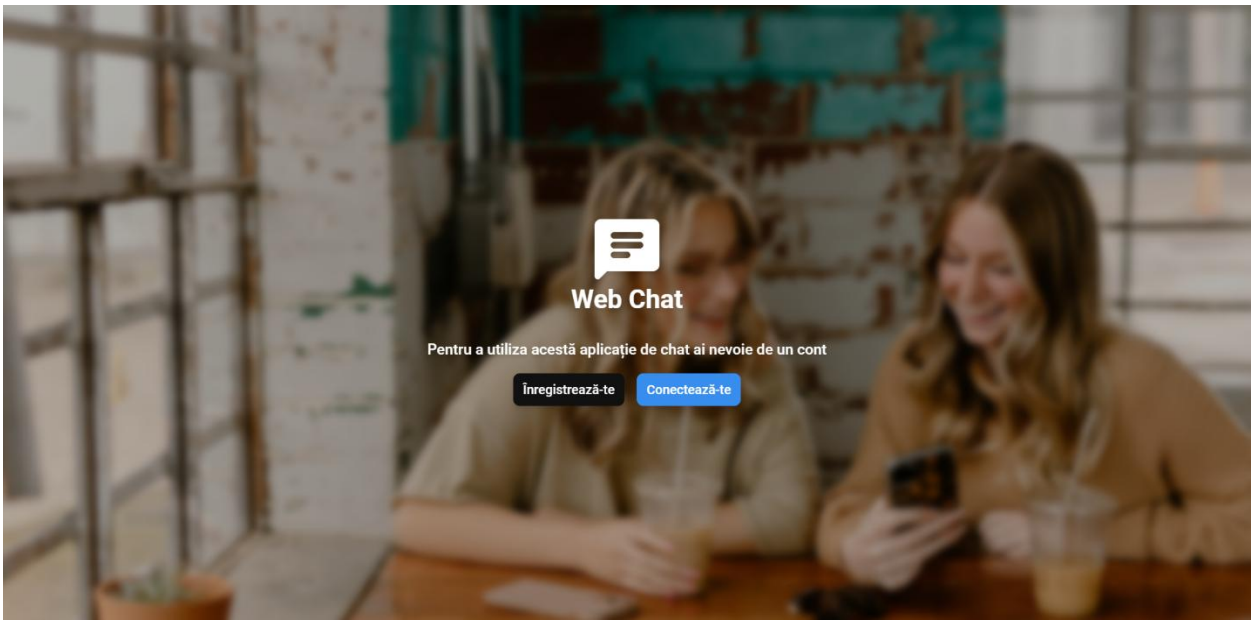
Tabelul users este conectat la **messages** pentru identificarea expeditorului și a destinatarului fiecărui mesaj.

3. Pagini și componente

În acest capitol, vom descrie în detaliu paginile și componentele aplicației, oferind o perspectivă clară asupra funcționalităților disponibile și a modului în care acestea pot fi utilizate pentru a crea o experiență interactivă și intuitivă. Fiecare pagină a fost proiectată cu atenție pentru a răspunde nevoilor utilizatorilor, asigurând acces rapid la toate funcțiile aplicației.

3.1 Pagină principală

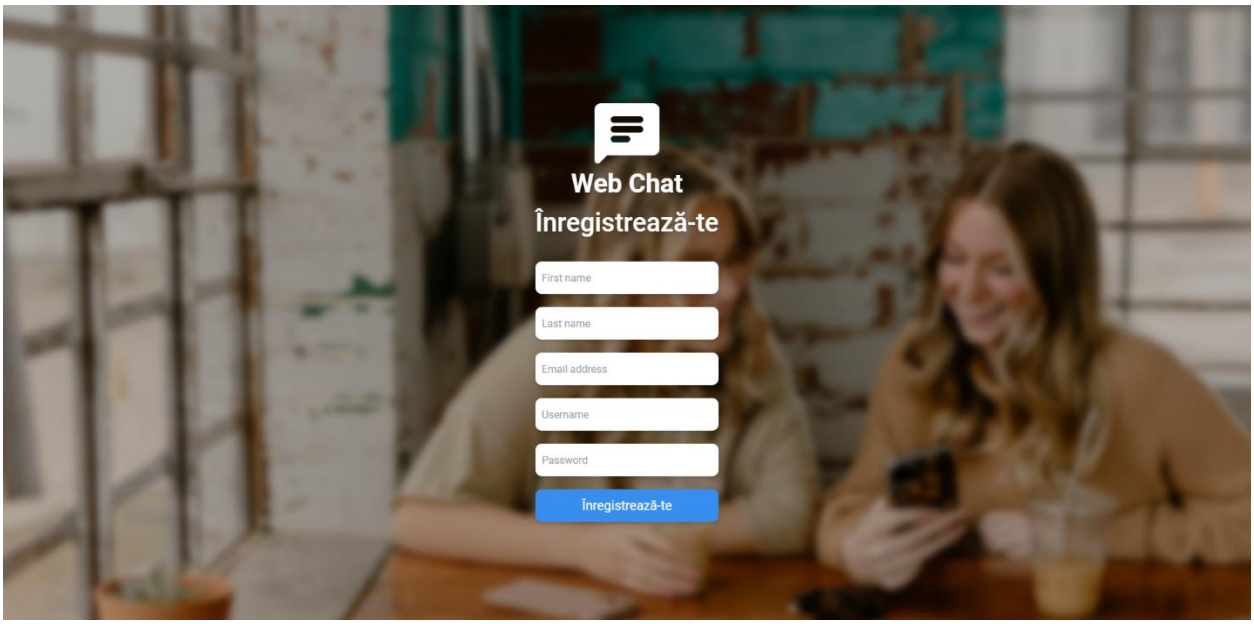
Aceasta este pagina principală a aplicației, concepută pentru a ghida utilizatorii într-un mod simplu și intuitiv către crearea unui cont nou sau conectarea la un cont existent. Pagina oferă o interfață prietenoasă și clară, asigurând o primă interacțiune plăcută cu aplicația.



Figură 3 Pagină principală

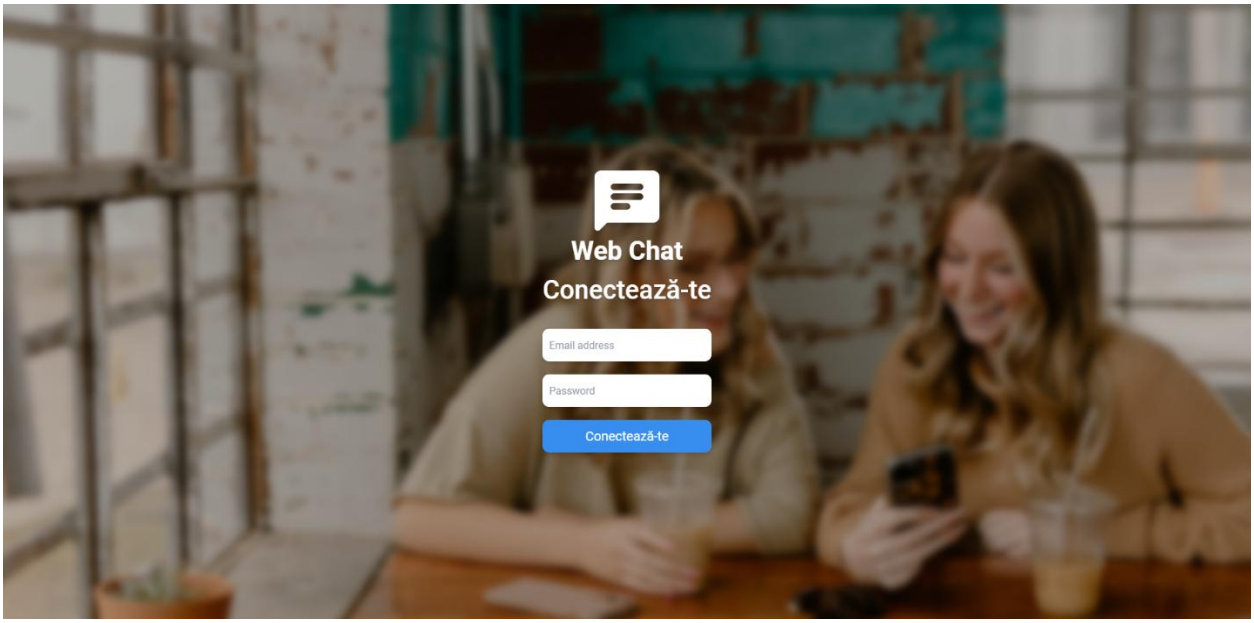
3.2 Pagină de înregistrare și conectare

Pagina de înregistrare permite utilizatorilor să își creeze un cont rapid utilizând niște informații de baza.



Figură 4 Pagină de înregistrare

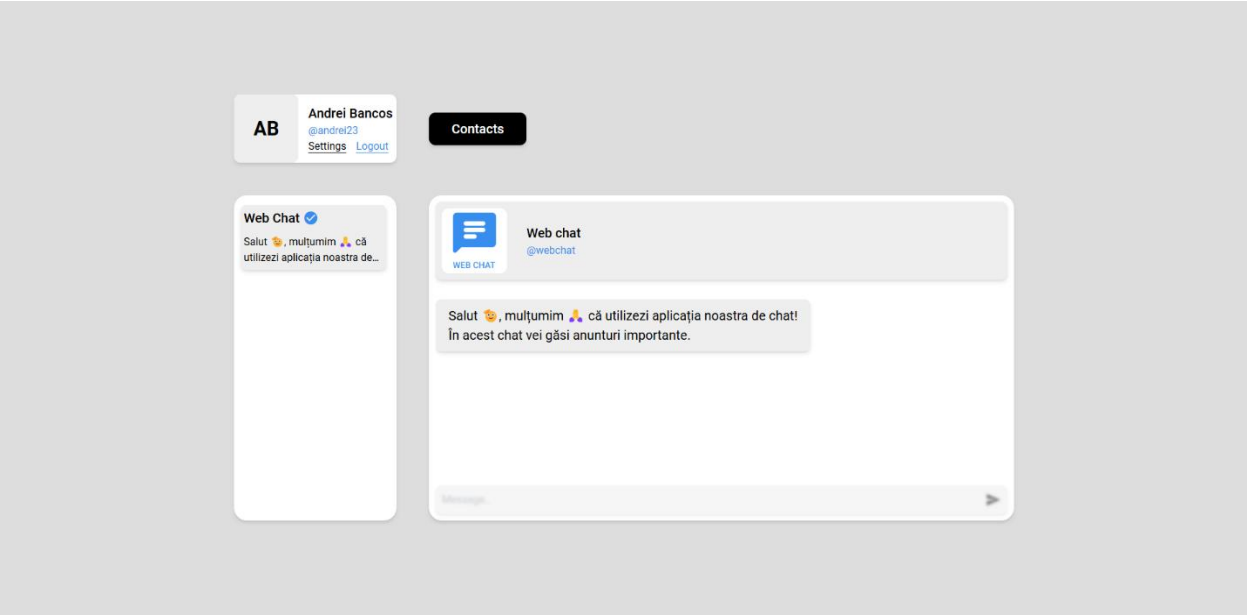
Pagina de conectare permite utilizatorilor accesul în aplicație.



Figură 5 Pagină de conectare

3.3 Pagină de chat (mesaje)

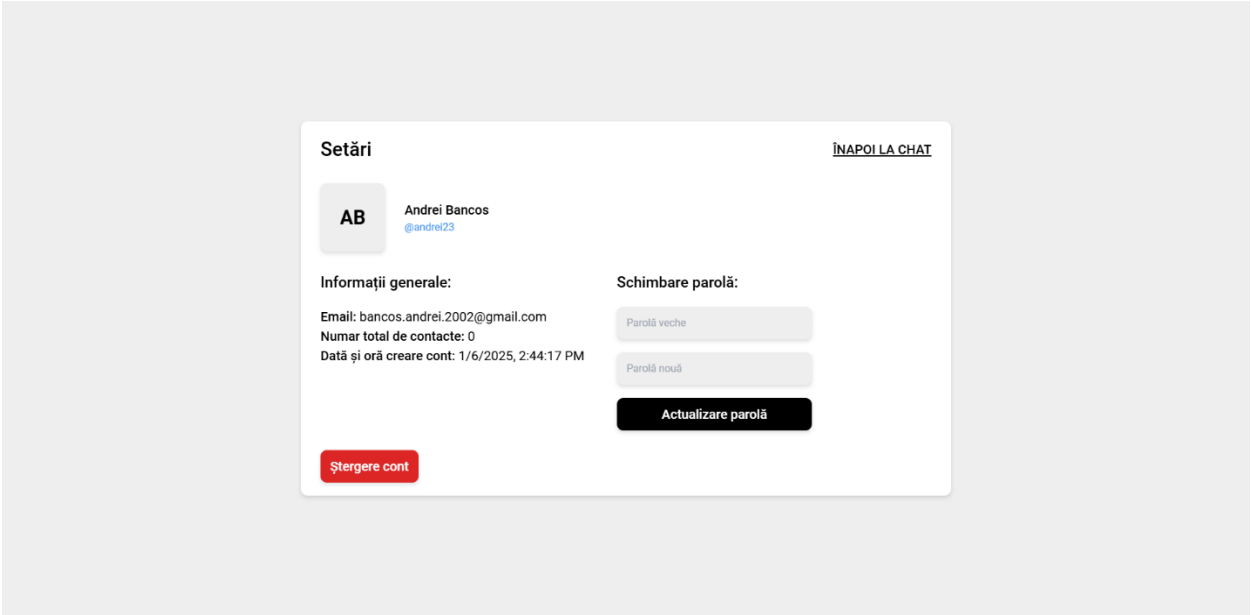
Aceasta este pagina de chat a aplicației, destinată comunicării 1 la 1 între utilizatori. Aici poți vizualiza lista de conversații existente, selecta un contact și trimite mesaje în timp real. Interfața este simplă și intuitivă, oferind acces la setări și funcționalitatea de deconectare. Pagina asigură o experiență fluidă și eficientă pentru utilizatori, fiind punctul central al interacțiunii din aplicație.



Figură 6 Pagină de chat

3.4 Pagină de setări

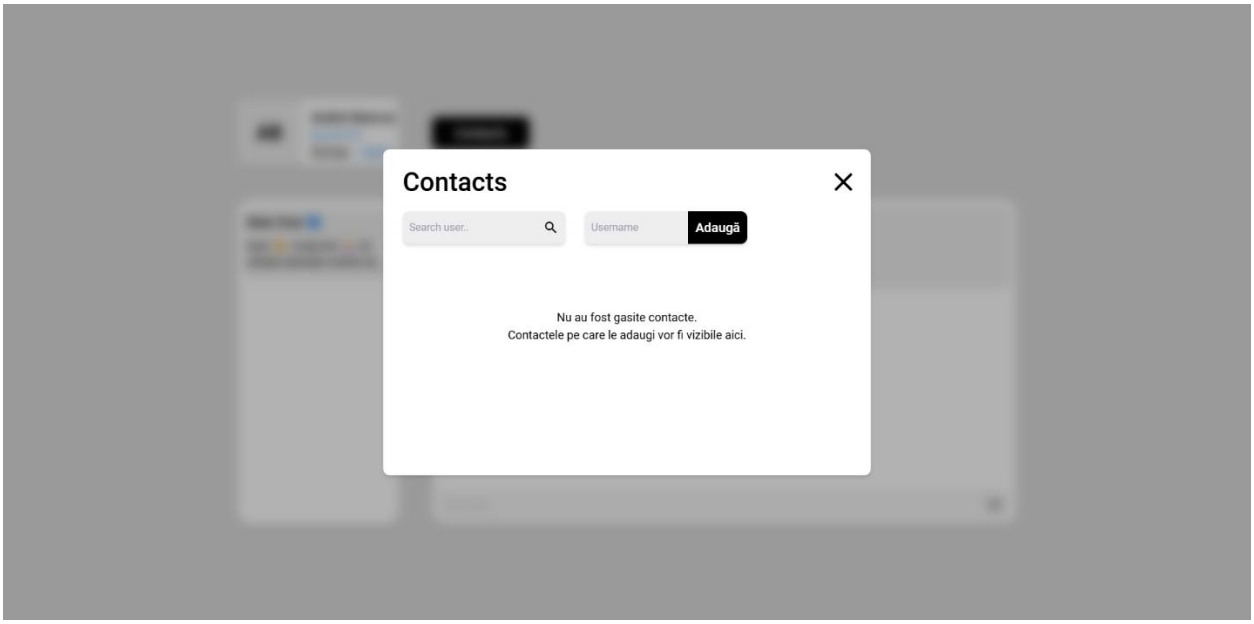
Pagina de setări permite utilizatorilor să gestioneze informațiile personale și să efectueze modificări legate de contul lor. În această pagină, utilizatorii pot vizualiza detalii generale, precum emailul, numele de utilizator, numărul total de contacte și data creării contului. De asemenea, există opțiunea de schimbare a parolei prin introducerea parolei vechi și setarea uneia noi. Tot aici, utilizatorii pot șterge definitiv contul printr-un buton dedicat. Interfața include și o opțiune pentru a reveni rapid la pagina de chat, asigurând o navigare ușoară și funcționalitate completă.



Figură 7 Pagină de setări

3.5 Componenta de contacte

Componenta de contacte este dedicată gestionării listelor de contacte și permite utilizatorilor să adauge noi contacte în aplicație. Aceasta include o bară de căutare pentru găsirea rapidă a utilizatorilor după nume de utilizator, precum și un câmp în care se poate introduce direct numele de utilizator pentru a adăuga un contact. După adăugare, contactele vor fi afișate în această listă, asigurând acces rapid și organizat la conversațiile individuale. Interfața este intuitivă.



Figură 8 Componenta de contacte

4. Testare și validare

Pentru asigurarea funcționării corecte a aplicației, am realizat o testare manuală atât a front-end-ului, dezvoltat în Angular, cât și a back-end-ului, construit cu Spring Boot. Testarea a fost efectuată etapizat, verificând funcționalitatea fiecărei componente și interacțiunea dintre acestea, astfel încât să oferim utilizatorilor o experiență cât mai fluidă și fără erori.

Testarea manuală a front-end-ului (Angular)

Testarea interfeței de utilizare s-a axat pe verificarea funcționalității fiecărui element și a fluxurilor principale de utilizare. Pașii principali ai testării includ:

1. **Pagini de logare și înregistrare**:
    - Am verificat validarea câmpurilor (câmpuri goale, email invalid, parole necorespunzătoare, etc).
    - Am testat afișarea corectă a mesajelor de eroare atunci când datele introduse sunt invalide.
    - Am verificat dacă utilizatorii sunt redirecționați corect către pagina de chat după autentificare.
2. **Pagina de chat:**
    - Am testat afișarea listei de conversații și actualizarea acesteia în timp real la primirea unui mesaj.
    - Am verificat trimiterea și recepționarea mesajelor, asigurând afișarea lor corectă în interfață.
    - Am verificat funcționarea barei de căutare a contactelor și deschiderea unei conversații noi.
3. **Pagina de setări**:
    - Am testat funcționalitatea schimbării parolei, inclusiv afișarea mesajelor de succes sau eroare.
    - Am verificat afișarea corectă a informațiilor utilizatorului (email, data creării contului, numărul de contacte).
    - Am verificat procesul de ștergere a contului și redirecționarea către pagina de logare.
4. **Design responsiv**:
    - Am testat interfața pe diferite rezoluții (desktop, tabletă, mobil) pentru a mă asigura că elementele sunt afișate corect și că experiența utilizatorului rămâne optimă.

Testarea manuală a back-end-ului (Spring Boot)

Pentru partea de back-end, am verificat funcționalitatea endpoint-urilor, comunicarea corectă cu baza de date PostgreSQL și integrarea cu front-end-ul. Testele principale efectuate au fost:

1. **Autentificare și înregistrare**:
    - Am verificat endpoint-urile responsabile de logare și înregistrare pentru validarea datelor introduse și returnarea răspunsurilor corespunzătoare.
    - Am testat generarea și verificarea token-urilor de autentificare pentru sesiuni sigure.
2. **Trimiterea și recepționarea mesajelor**:
    - Am verificat endpoint-urile pentru gestionarea mesajelor (creare, citire, listare) folosind unelte precum Postman și Swagger.
    - Am testat comunicarea prin WebSocket pentru trimiterea mesajelor în timp real, asigurându-mă că expeditorul și destinatarul primesc mesajele corect.
3. **Gestionarea contactelor**:
    - Am testat funcționalitatea adăugării și listării contactelor, verificând corectitudinea relațiilor many-to-many din baza de date.
    - Am verificat răspunsurile endpoint-urilor pentru adăugarea unui contact inexistent sau dublu.
4. **Erori și gestionarea excepțiilor**:
    - Am simulat scenarii de eroare pentru a verifica gestionarea acestora și returnarea mesajelor adecvate.
5. **Performanță și stabilitate**:
    - Am monitorizat consumul de resurse al aplicației în timpul rulării.

Bibliografie

Pawar, P. (2024). *Client Server Architecture: Key Components, Types, Benefits*. Preluat de pe https://www.aalpha.net/blog/client-server-architecture/.