UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Aplicatie chat

Echipa: Cei trei muschetari

Inginerie Software

Studentii care au realizat acest proiect:

Mihut Cristian Mihai
Curea Andrei
Sur Patrick

Grupa: 30236

Facultatea de Automatica
si Calculatoare

10 ianuarie 2026

Cuprins

1 Descrierea proiectului

Aceasta aplicatie reprezinta o platforma de tip chat, inspirata de aplicatii populare precum WhatsApp. Ne-am propus sa introducem cat mai multe functionalitati in aplicatia noastra. Utilizatorii pot crea conturi, se pot autentifica si pot sa inceapa o conversatie cu un alt utilizator. In cadrul conversatiilor, acestia pot trimite mesaje text, imagini si pot initia apeluri audio sau video in timp real.

Aplicatia este construita pe o arhitectura client–server moderna, folosind comunicare in timp real prin WebSockets.

2 Limbaje si tehnologii utilizate

2.1 Frontend

- Limbaje: HTML, CSS, JavaScript
- Framework: React
- Deploy: Vercel

2.2 Backend

- Limbaj: JavaScript
- Runtime: Node.js
- Framework: Express
- Comunicare real-time: Socket.IO
- Deploy: Render

2.3 Baza de date

- Sistem: PostgreSQL
- Provider: NeonDB

3 Cerinte functionale

- Autentificare si inregistrare utilizatori
- Vizualizare utilizatori online
- Initiere conversatie intre doi utilizatori
- Trimitere mesaje text
- Trimitere imagini
- Initiere apel audio si video
- Deconectare sigura a utilizatorului

3.1 Diagrama Use-Case

Diagrama Use-Case evidentiaza interactiunea utilizatorului cu sistemul: autentificare, pornire chat, trimitere mesaje si initiere apeluri.

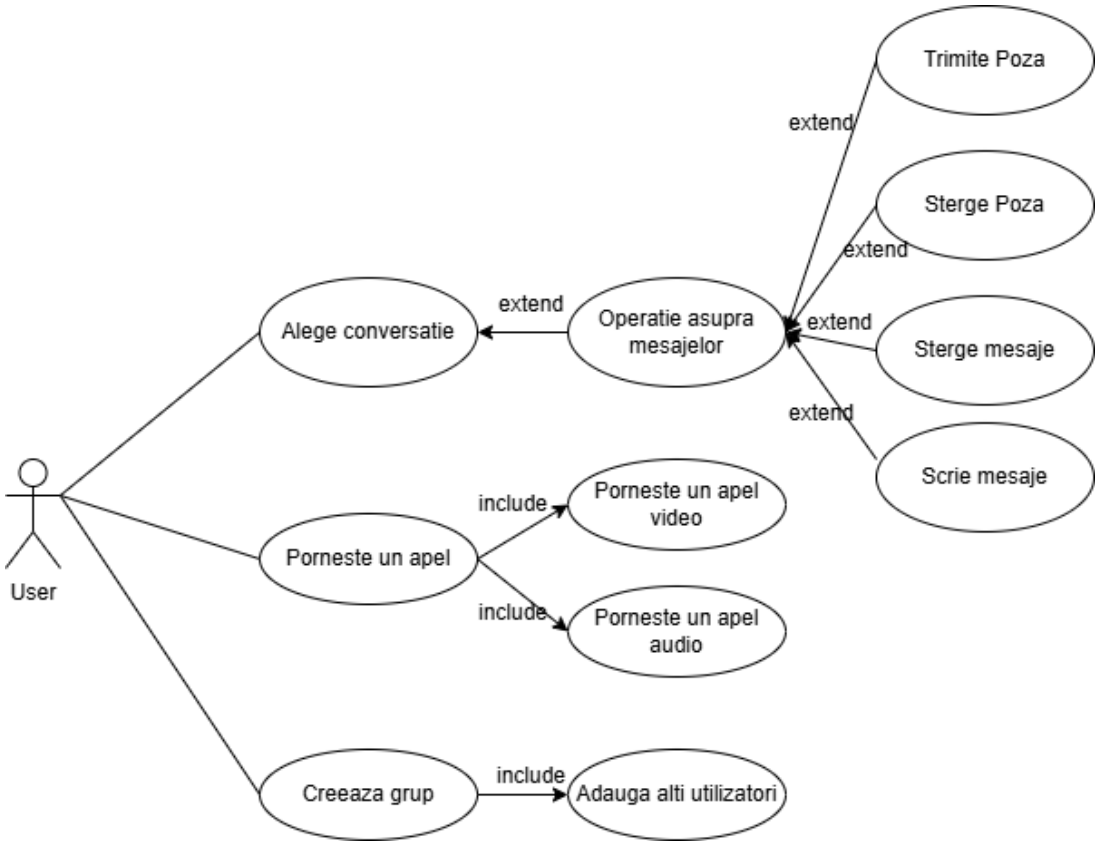


Figura 1: Diagrama Use-Case a aplicatiei de chat

4 Cerinte non-functionale

- Aplicatia trebuie sa raspunda in timp real
- Securitate prin autentificare JWT
- Scalabilitate pentru mai multi utilizatori simultan
- Interfata intuitiva
- Disponibilitate ridicata a serviciilor

5 Diagrame UML

Diagrame realizate de Cristi

Aceste diagrame UML au fost realizate de Cristi si descriu principalele fluxuri functionale ale aplicatiei de chat, de la interactiunea in timp real dintre utilizatori pana la gestionarea starilor mesajelor.

5.1 Diagrama de Comunicare – Lista utilizatori online

Aceasta diagrama de comunicare ilustreaza modul in care utilizatorii sunt gestionati pe partea de server in momentul conectarii la aplicatie. Sunt prezentate interactiunile dintre componentele principale responsabile de autentificare, inregistrarea sesiunilor active si notificarea clientilor existenti despre utilizatorii online.

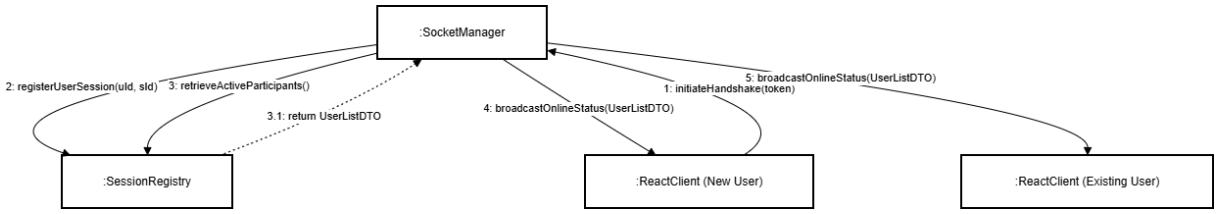


Figura 2: Diagrama de comunicare – Lista utilizatori online

5.2 Diagrama de Secventa – Preluarea istoricului conversatiei

Diagrama de secventa prezinta pasii parcursi atunci cand un utilizator selecteaza o conversatie existenta. Clientul solicita istoricul mesajelor de la server, acesta interogheaza baza de date, iar rezultatul este returnat sub forma unui obiect JSON care este ulterior afisat in interfata grafica.

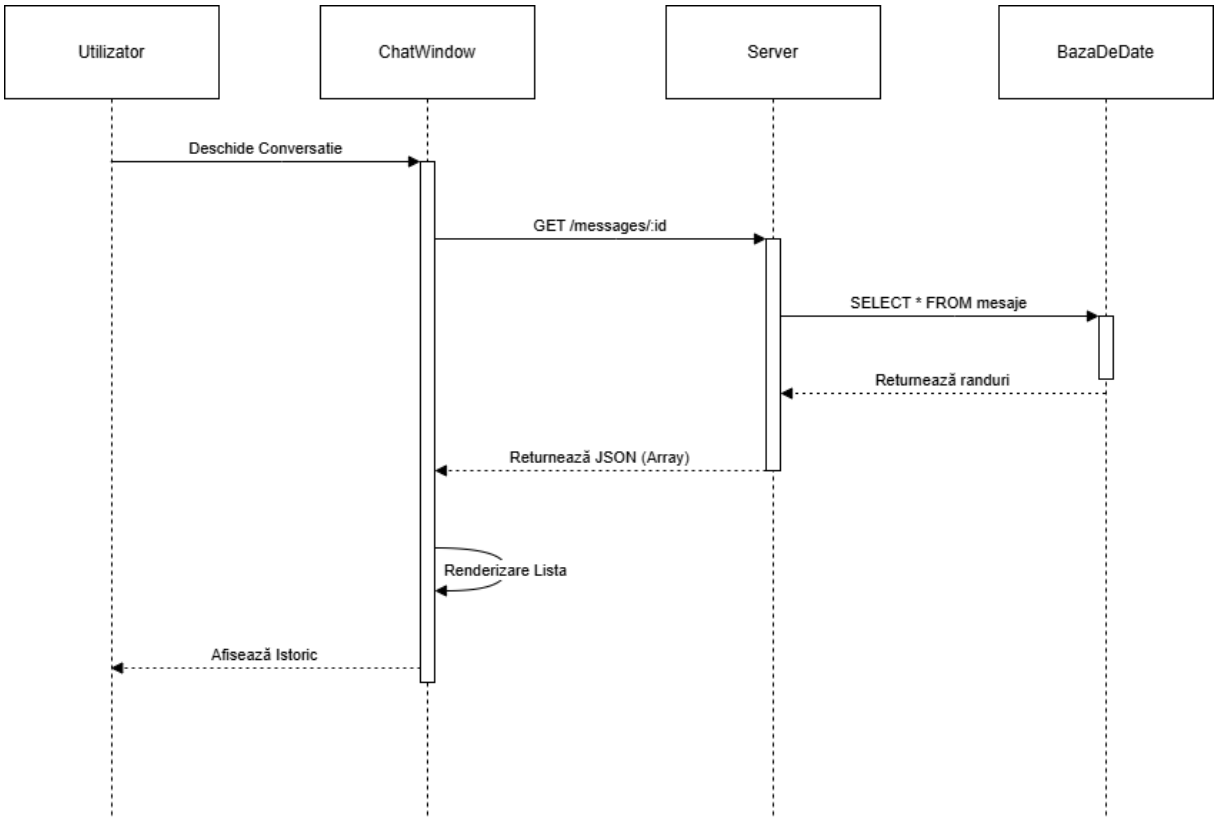
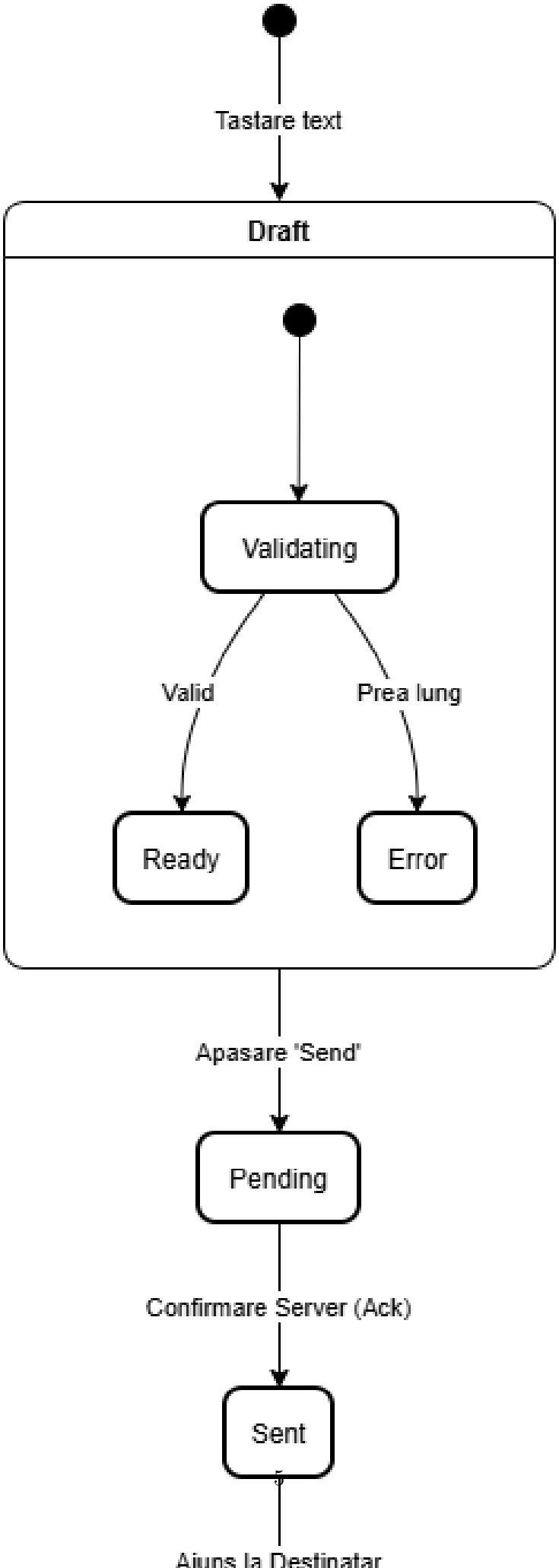


Figura 3: Diagrama de secventa – Preluarea istoricului conversatiei

5.3 Diagrama de Stare – Starea unui mesaj

Aceasta diagrama de stare descrie ciclul de viata al unui mesaj in aplicatia de chat. Sunt evidentiate starile prin care trece un mesaj, de la momentul redactarii si validarii, pana la trimiterea, livrarea si citirea acestuia de catre destinatar.


Tastare text
Draft
Validating
Valid
Prea lung
Ready
Error
Apasare 'Send'
Pending
Confirmare Server (Ack)
Sent
Ajuns la Destinatar

Diagrame realizate de Patrick

Aceste diagrame UML au fost realizate de Patrick si descriu fluxurile legate de autentificare, deconectare si gestionarea starilor sesiunii utilizatorului in aplicatia de chat.

5.4 Diagrama de Activitate – Autentificare utilizator

Aceasta diagrama de activitate prezinta fluxul complet de autentificare si inregistrare al utilizatorului. Sunt evidentiate deciziile dintre modul login si modul register, tratarea erorilor, precum si tranzitia catre interfata principala a aplicatiei in cazul autentificarii reusite.

Figura 5: Diagrama de activitate – Autentificare utilizator

5.5 Diagrama de Secventa – Deconectare utilizator

Diagrama de secventa descrie procesul de deconectare sigura a utilizatorului din aplicatie. Sunt ilustrate interactiunile dintre interfata grafica, serviciul de socket, stocarea locala si starea aplicatiei, asigurand inchiderea corecta a conexiunii si curatarea datelor de autentificare.

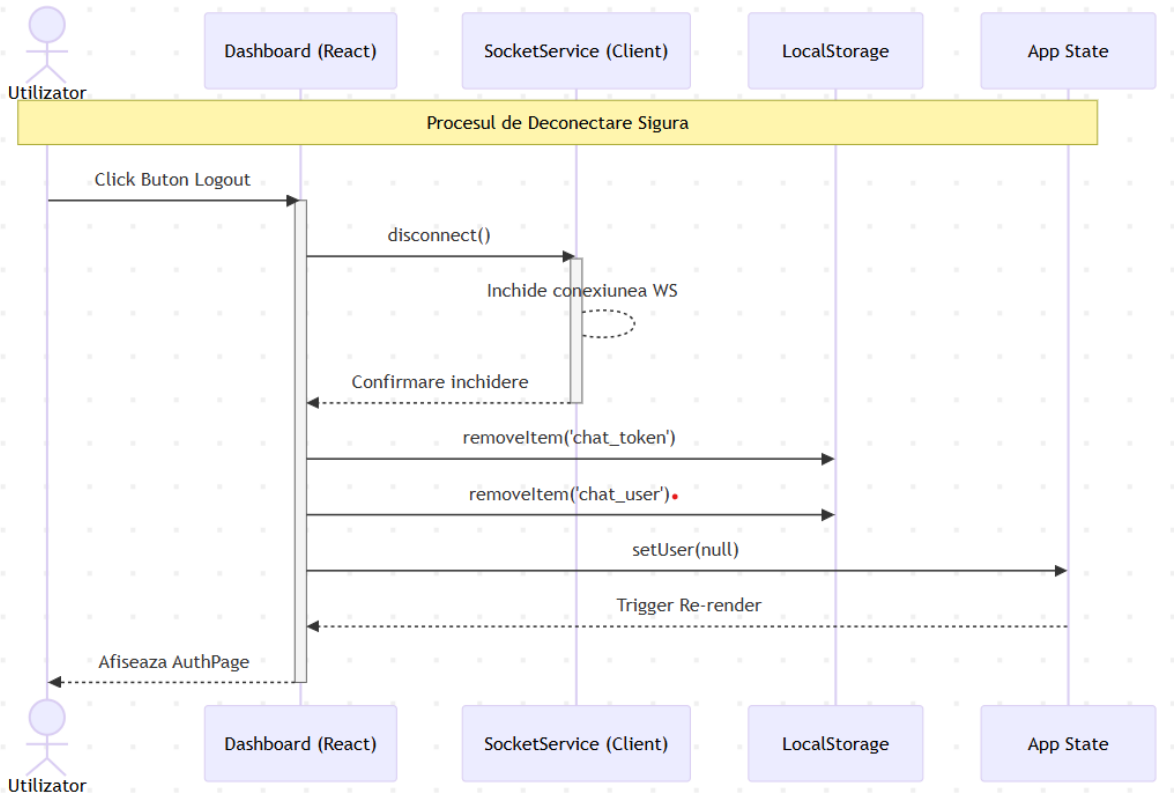


Figura 6: Diagrama de secventa – Deconectare utilizator

5.6 Diagrama de Stare – Starea sesiunii utilizatorului

Aceasta diagrama de stare ilustreaza starile prin care poate trece sesiunea unui utilizator in aplicatie, de la momentul pornirii aplicatiei, autentificare, utilizare activa, pana la deconectare sau inchiderea browserului.

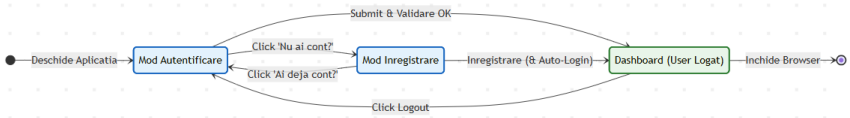


Figura 7: Diagrama de stare – Starea sesiunii utilizatorului

Diagrame realizate de Andrei

Urmatoarele diagrame UML au fost realizate de Andrei si descriu fluxurile esentiale pentru initierea conversatiilor, trimiterea mesajelor si gestionarea conexiunii utilizatorului in aplicatia de chat.

5.7 Diagrama de Comunicare – Inceperea unei conversatii

Aceasta diagrama de comunicare prezinta pasii necesari pentru initierea unei conversatii intre doi utilizatori. Sunt ilustrate interactiunile dintre client, server si mecanismele de autentificare, precum si notificarea participantilor implicati in conversatie.

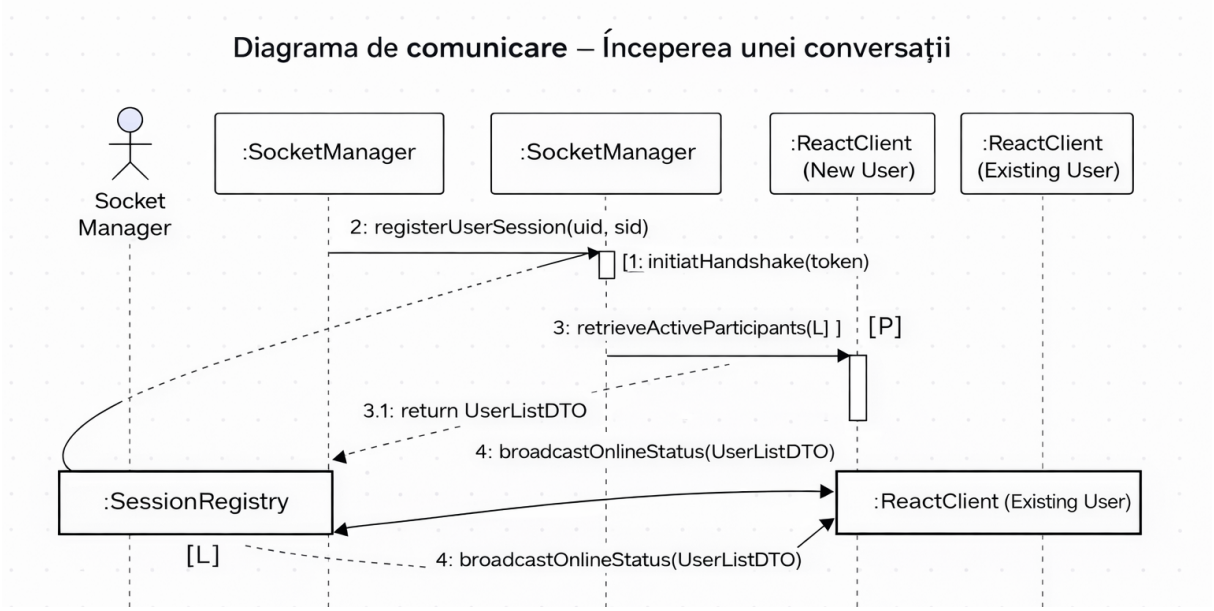


Figura 8: Diagrama de comunicare – Inceperea unei conversatii

5.8 Diagrama de Secventa – Trimiterea unui mesaj

Diagrama de secventa descrie fluxul complet al trimiterii unui mesaj text intre doi utilizatori. Mesajul este transmis catre server prin Socket.IO, este salvat in baza de date si ulterior distribuit catre destinatar in timp real.

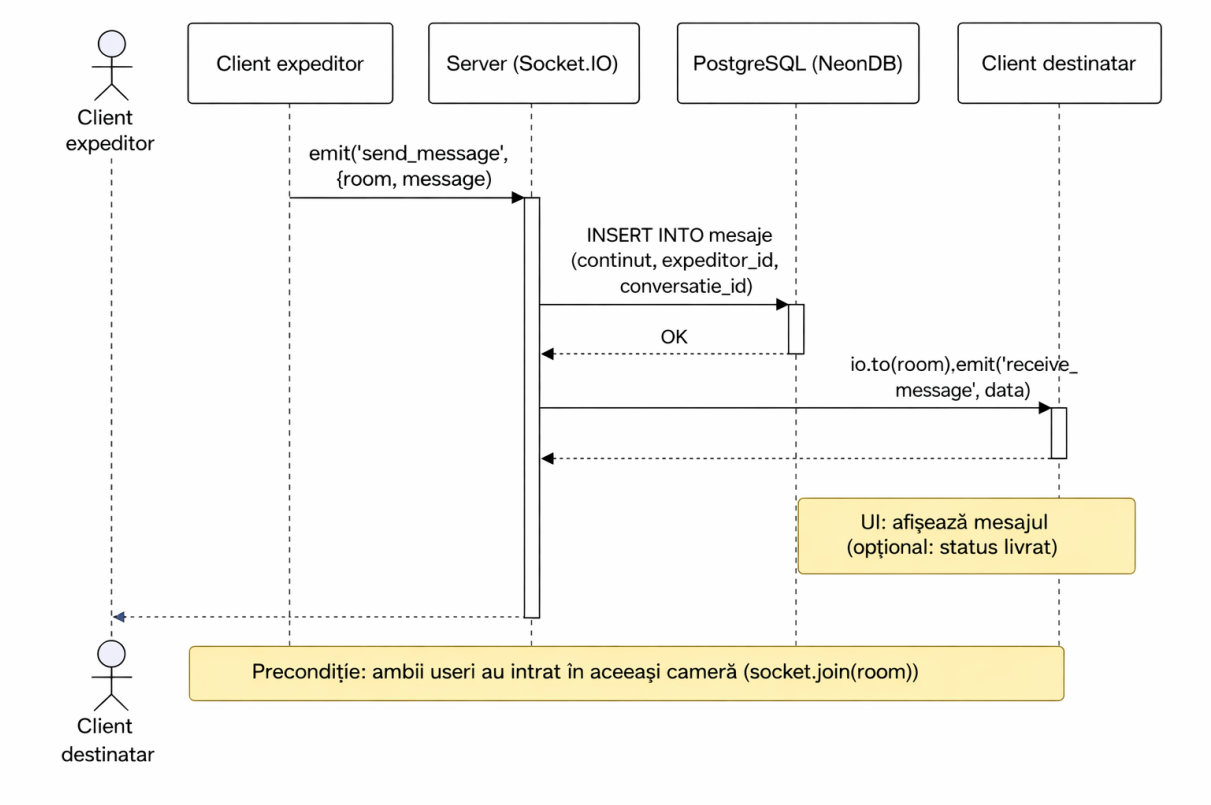


Figura 9: Diagrama de secventa – Trimiterea unui mesaj

5.9 Diagrama de Stare – Conexiunea utilizatorului

Aceasta diagrama de stare evidentiaza starile prin care trece conexiunea unui utilizator pe parcursul utilizarii aplicatiei. Sunt prezentate tranzitiile dintre starea deconectata, conectare, conectat si deconectare, inclusiv tratarea erorilor de autentificare.

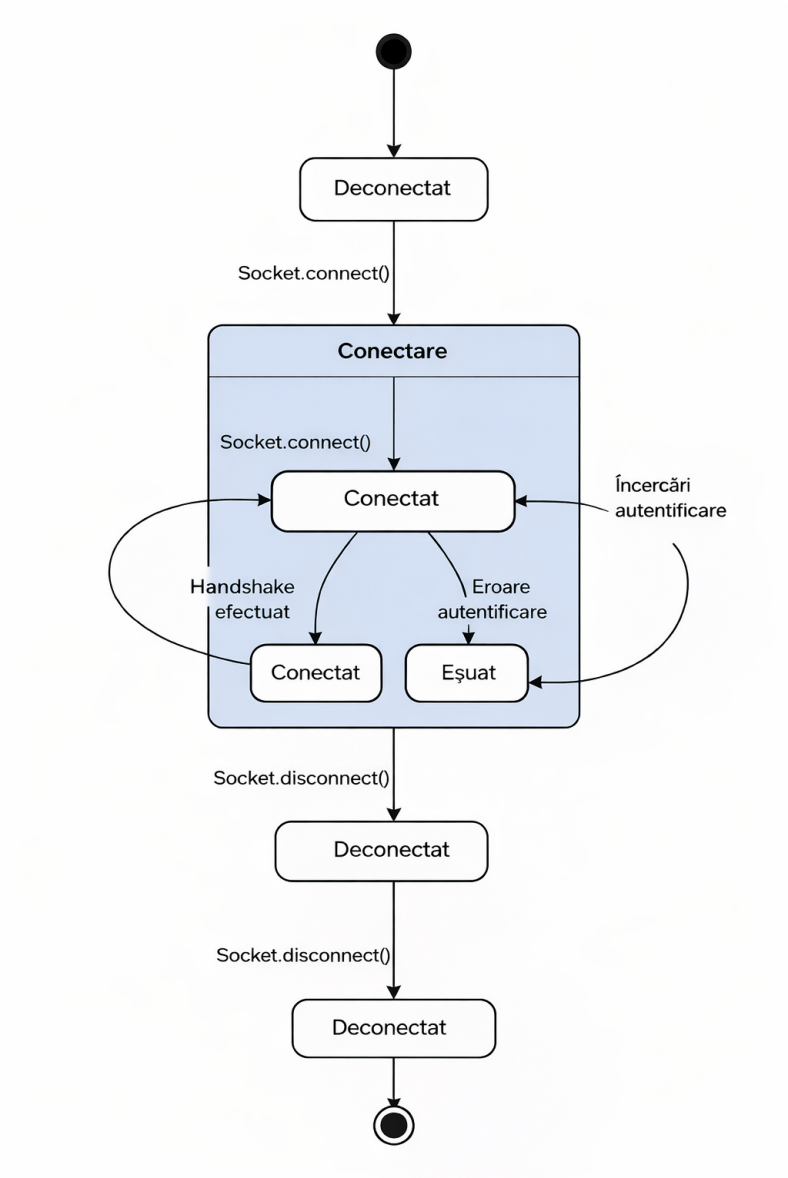


Figura 10: Diagrama de stare – Conexiunea utilizatorului

6 Design Patterns

In cadrul dezvoltarii aplicatiei de chat au fost identificate si utilizate mai multe design pattern-uri, fiecare membru al echipei contribuind la arhitectura aplicatiei prin aplicarea unui pattern specific, adaptat responsabilitatilor implementate.

6.1 Andrei – Observer

Pattern-ul **Observer** a fost utilizat pentru notificarea in timp real a utilizatorilor cu privire la modificarile aparute in aplicatie. Serverul actioneaza ca un *Subject*, iar clientii conectati reprezinta *Observers*. Prin intermediul Socket.IO, serverul emite evenimente precum primirea unui mesaj nou, actualizarea listei de utilizatori online sau initierea unui apel, iar clientii abonati reactioneaza automat la aceste schimbari.

6.2 Cristian – Chain of Responsibility

Pattern-ul **Chain of Responsibility** a fost aplicat pentru securizarea datelor si controlul fluxului de cereri catre server. Cererile HTTP sunt procesate succesiv printr-un lant de middleware-uri precum `express.json()`, `cors()`, `authMiddleware` si rutele protejate (ex: `myConversations`). Fiecare componenta din lant are responsabilitatea de a valida sau prelucra cererea inainte de a fi transmisa mai departe.

6.3 Patrick – Singleton

Pattern-ul **Singleton** a fost utilizat pentru gestionarea conexiunii cu baza de date. Prin asigurarea unei singure instante de conexiune la baza de date PostgreSQL (NeonDB), aplicatia evita crearea mai multor conexiuni inutile, asigurand consistenta datelor si o utilizare eficienta a resurselor serverului.

7 ReadMe

Aplicatia noastra este hostata online prin intermediul a doua platforme de cloud hosting:Render(backend) si Vercel(frontend).Pentru a testa aplicatia alaturi de functionalitatile ei accesati linkul:https://aplicatie-chat.vercel.app/ .