Aplicație Chat & Games

13 ianuarie 2026

Cuprins

Capitolul 1

Introducere

1.1 Scopul Documentației

Acest document reprezintă ghidul tehnic complet pentru aplicația web "Chat & Games". Scopul său este de a oferi o înțelegere detaliată a arhitecturii, componentelor, tehnologiilor și fluxurilor de date ale sistemului. Documentația se adresează dezvoltatorilor software, arhitecților de sistem și personalului tehnic responsabil cu întreținerea și dezvoltarea viitoare a aplicației.

1.2 Prezentare Generală a Aplicației

"Chat & Games" este o platformă web interactivă, în timp real, care combină funcționalități de comunicare socială cu jocuri multi-player. Aplicația oferă utilizatorilor o experiență integrată, permițându-le să interacționeze și să se joace în același mediu.

Principalele funcționalități ale platformei includ:

- **Sistem de Autentificare:** Utilizatorii își pot crea conturi noi și se pot autentifica folosind credențiale securizate.
- **Chat Global:** O cameră de chat publică unde toți utilizatorii conectați pot comunica în timp real.
- **Camere de Chat (Rooms):** Utilizatorii pot crea sau se pot alătura unor camere de chat tematice, private sau publice, pentru discuții de grup.
- **Chat Privat:** Funcționalitate de mesagerie directă între doi utilizatori.
- **Joc de Poker:** O implementare a jocului Texas Hold'em Poker, unde utilizatorii se pot alătura unor mese, pot juca cu fise virtuale și pot interacționa prin acțiuni specifice jocului (call, raise, fold etc.).
- **Joc de Hangman (Spânzurătoarea):** Un joc clasic pentru doi jucători, în care un utilizator setează un cuvânt, iar celălalt încearcă să-l ghicească literă cu literă.

1.3 Tehnologii Utilizate

Aplicația este construită pe un stack tehnologic modern, separând clar logica de backend de interfața de frontend, pentru a asigura scalabilitate, mentenabilitate și o experiență de utilizare fluidă.

1.3.1 Backend (Spring Boot)

Serverul aplicației este dezvoltat folosind ecosistemul **Spring Boot**, o platformă robustă bazată pe limbajul Java, recunoscută pentru performanță și securitate.

- **Spring Web (MVC):** Utilizat pentru a expune endpoint-uri RESTful API, necesare pentru acțiuni precum înregistrarea, autentificarea și gestionarea inițială a jocurilor.
- **Spring WebSocket:** Asigură comunicarea bidirecțională, în timp real, între client și server. Aceasta este tehnologia cheie pentru funcționalitățile de chat și pentru sincronizarea stării jocurilor.
- **Spring Security:** Gestionează procesele de autentificare și autorizare. Securitatea este implementată folosind JSON Web Tokens (JWT), asigurând că fiecare cerere către resurse protejate este validată corespunzător.
- **Spring Data JPA & Hibernate:** Utilizate pentru persistența datelor. Modelele de date (Utilizatori, Jocuri etc.) sunt mapate către o bază de date relațională (ex: PostgreSQL), iar interacțiunile sunt gestionate prin repository-uri JPA.

1.3.2 Frontend (React)

Interfața cu utilizatorul este o aplicație de tip **Single Page Application (SPA)**, construită cu biblioteca **React.js**.

- **React.js:** Biblioteca principală pentru construirea componentelor UI reutilizabile și gestionarea stării aplicației.
- **React Router:** Utilizat pentru navigația client-side, permițând o experiență de utilizare rapidă, fără reîncărcarea paginii.
- **Bootstrap:** Cadrul CSS utilizat pentru stilizarea componentelor, asigurând un design modern și responsiv.
- **WebSocket API (Client):** Browser-ul utilizează API-ul nativ WebSocket pentru a stabili și menține conexiunea cu serverul Spring Boot.

Capitolul 2

Arhitectura Sistemului

2.1 Arhitectura de Nivel Înalt

Aplicația "Chat & Games" adoptă o arhitectură client-server clasică, decuplată, formată din trei straturi principale:

1. **Stratul de Prezentare (Client):** Reprezentat de aplicația React care rulează în browser-ul utilizatorului. Acesta este responsabil pentru randarea interfeței grafice, gestionarea interacțiunilor utilizatorului și comunicarea cu serverul.
2. **Stratul de Logică a Aplicației (Server):** Reprezentat de aplicația Spring Boot. Acesta conține logica de business, gestionează starea jocurilor, procesează mesajele de chat, se ocupă de securitate și persistă datele.
3. **Stratul de Date (Baza de Date):** O bază de date relațională (ex: PostgreSQL) unde sunt stocate informațiile persistente, precum conturile de utilizator, istoricul jocurilor sau setările.

Comunicarea între client și server se realizează prin două canale principale:

- **HTTP/S (REST API):** Pentru acțiuni stateless, cum ar fi autentificarea sau crearea unei noi mese de joc.
- **WebSockets (WSS):** Pentru comunicare stateful și în timp real, esențială pentru chat și actualizarea stării jocurilor în desfășurare.

Figura 2.1: Diagrama arhitecturală de nivel înalt.

Notă: Figura 2.1 este un placeholder. O diagramă reală ar trebui să ilustreze clientul (React), serverul (Spring Boot) și baza de date, împreună cu fluxurile de comunicare (REST și WebSocket).

2.2 Componentele Backend (Spring Boot)

Backend-ul este structurat modular pentru a separa responsabilitățile.

2.2.1 Controlere REST (@RestController)

Aceste clase gestionează cererile HTTP și expun endpoint-urile API.

- `AuthController`: Responsabil pentru endpoint-urile `/api/auth/register`, `/api/auth/login` și `/api/auth/logout`.
- `PokerGameController`: Gestionează crearea și alăturarea la mesele de poker (`/api/poker/create` `/api/poker/join`).
- `HangmanGameController`: Responsabil pentru crearea și alăturarea la jocurile de Hangman.

2.2.2 Gestionarea Conexiunilor WebSocket (@WebSocketHandler)

O componentă centrală, `GameWebSocketHandler`, gestionează ciclul de viață al conexiunilor WebSocket.

- **afterConnectionEstablished:** Se execută la conectarea unui nou client. Aici se validează token-ul JWT și se adaugă sesiunea clientului într-o colecție de clienți activi.
- **handleTextMessage:** Procesează mesajele primite de la clienți. Un mecanism de dispecerizare analizează tipul mesajului (ex: `poker_action`, `sendRoomMessage`) și invocă serviciul corespunzător.
- **afterConnectionClosed:** Se execută la deconectarea unui client, eliminându-l din jocurile active și din lista de utilizatori online.

2.2.3 Servicii de Business (@Service)

Aceste clase conțin logica principală a aplicației și nu depind direct de protocolul de comunicare (HTTP sau WebSocket).

- `UserService`: Gestionează logica legată de utilizatori (creare, autentificare).
- `JwtService`: Generează și validează token-uri JWT.
- `PokerGameService`: Menține starea tuturor jocurilor de poker active (în memorie), procesează acțiunile jucătorilor (check, raise, fold) și determină câștigătorii.
- `HangmanGameService`: Gestionează logica pentru jocurile de Hangman.
- `ChatService`: Gestionează camerele de chat și distribuie mesajele către destinatarii corecți.

2.2.4 Stratul de Persistență (@Repository)

Interfețele care extind `JpaRepository` (ex: `UserRepository`) asigură o abstractizare peste operațiunile cu baza de date, permițând operații CRUD (Create, Read, Update, Delete) simple și eficiente.

2.3 Componentele Frontend (React)

Frontend-ul este construit pe bază de componente, fiecare având o responsabilitate specifică.

- **Componente de Autentificare:** `Login.jsx`, `Register.jsx` - formulare pentru autentificare și înregistrare.
- **Componente de Layout:** `Header.jsx` - conține meniul principal de navigație, lista de utilizatori activi și butonul de logout.
- **Componente de Chat:** `GlobalChat.jsx`, `RoomChat.jsx`, `PrivateChat.jsx` - implementează interfețele pentru diferitele tipuri de chat.
- **Componente de Joc:**
  - `PokerLobby.jsx` și `PokerTable.jsx`: Afișează lista de mese de poker disponibile și, respectiv, interfața unei mese de joc în desfășurare.
  - `HangmanLobby.jsx` și `HangmanGame.jsx`: Componente similare pentru jocul Hangman.
- **Serviciu WebSocket:** Un modul JavaScript (nu o componentă React) responsabil pentru gestionarea conexiunii WebSocket: conectare, trimitere de mesaje și primirea de evenimente de la server, pe care le distribuie apoi componentelor relevante.