UTCN-POLL

Cincă Constantin
Drăghici Alexandru
Mihețan Nicoleta
Sărăndan Vlad

Universitatea Tehnică din Cluj-Napoca
Facultatea de Automatică și Calculatoare

2026

Contents

1 Introducere

1.1 Problema Abordată

În peisajul academic modern, comunicarea dintre studenți și administrația facultății, precum și între studenți înșiși, este adesea fragmentată. Informațiile esențiale sunt disipate pe multiple platforme neoficiale precum grupuri de Facebook, WhatsApp sau pagini de Instagram. Această fragmentare duce la o serie de probleme:

- **Lipsa unei surse unice de adevăr:** Este dificil pentru studenți să găsească informații oficiale și verificate.
- **Comunicare ineficientă:** Mesajele importante se pot pierde în zgomotul de pe rețelele sociale.
- **Dependența de intermediari:** Adesea, o singură persoană (administratorul unui grup) devine un punct central de blocaj pentru diseminarea informațiilor.
- **Lipsa de implicare a comunității:** Nu există o platformă centralizată care să încurajeze interacțiunea și angajamentul studenților pe teme academice și extracurriculare.

1.2 Soluția: UTCN-POLL

Proiectul **UTCN-POLL** a fost conceput ca o soluție directă la aceste probleme, propunând o platformă web dedicată, care centralizează comunicarea și adaugă elemente de interactivitate. Prin restricționarea accesului la utilizatorii cu adrese de e-mail instituționale (`@student.utcluj.ro`, `@campus.utcluj.ro`), platforma creează un mediu sigur și relevant pentru comunitatea academică.

Funcționalitățile cheie abordează direct problemele identificate:

- **Feed Social (Blog):** Oferă un spațiu centralizat pentru discuții, anunțuri și întrebări, eliminând necesitatea de a căuta informații pe multiple platforme.
- **Sondaje (Polls) cu Sistem de Pariuri:** Introduce un element de gamification care stimulează implicarea activă a utilizatorilor. Sondajele pot fi folosite de administrație pentru a culege feedback valoros de la studenți într-un mod interactiv.
- **Sistem de Jetoane (Tokens):** Creează o economie virtuală în cadrul platformei, recompensând utilizatorii pentru activitatea lor și permițându-le să participe la sondaje.

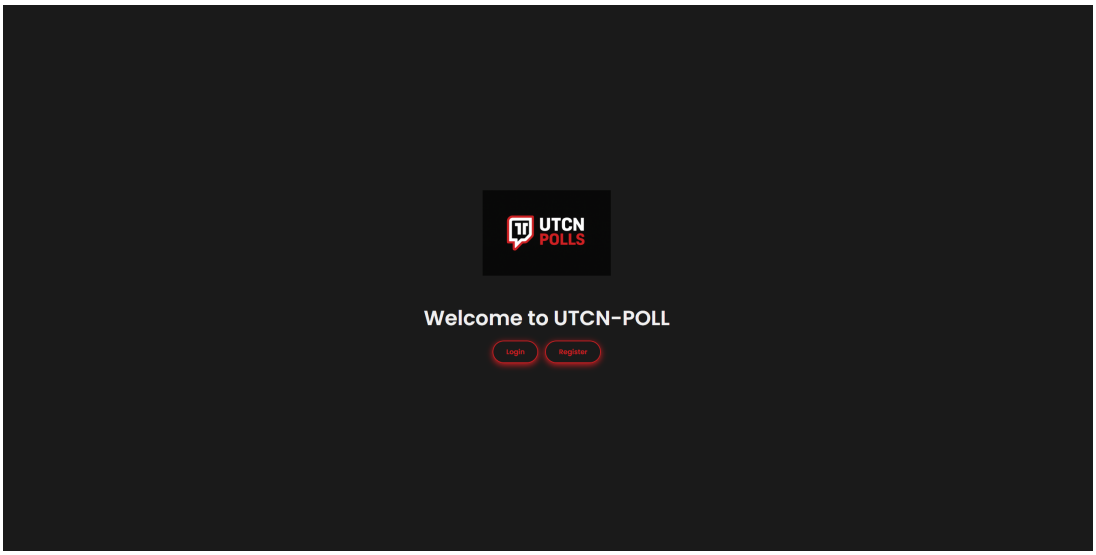

Figure 1: Pagina principală a aplicației (Homepage)

2 Arhitectura Tehnică Detaliată

2.1 Backend (Spring Boot)

2.1.1 Structura Pachetelor și Straturile Arhitecturale

Backend-ul respectă o arhitectură stratificată clasică, promovând principiul separării responsabilităților (Separation of Concerns).

- `models`: Definește structura datelor. Acestea sunt clase POJO (Plain Old Java Object) adnotate cu JPA pentru a reprezenta entitățile din baza de date.
- `repository`: Reprezintă stratul de acces la date (Data Access Layer). Interfețele, care extind `JpaRepository`, oferă metode CRUD (Create, Read, Update, Delete) și permit definirea de interogări custom.
- `service`: Conține logica de business. Acest strat orchestrează operațiunile, interacționând cu multiple repository-uri și implementând regulile aplicației (ex: validarea unui vot, calcularea recompenselor).
- `controller`: Stratul de expunere (Presentation Layer) care definește API-ul REST. Controlează fluxul de date între client și server, primește cereri HTTP, le deleagă serviciilor corespunzătoare și formatează răspunsurile.
- `security`: Un pachet transversal care gestionează autentificarea și autorizarea, bazat pe Spring Security și JWT.

2.1.2 Modele de Date (Entități)

- **User:** Entitatea centrală, stocând datele de bază.
  - `points`: Reprezintă moneda virtuală a platformei ("tokens").
  - `lastSpinDate`: Stochează data ultimului "daily spin" pentru a preveni abuzul.
  - `user_type`: Coloana discriminator care, în funcție de valoare ("ADMIN", "MEMBER"), determină clasa concretă instanțiată de Hibernate.
- **Poll:** Reprezintă un sondaj.
  - `endDate`: Marchează momentul la care sondajul se închide automat.
  - `resolved`: Un flag boolean ce indică dacă un administrator a finalizat sondajul și a distribuit premiile.
  - `winningOptionId`: Stochează ID-ul opțiunii câștigătoare după rezolvare.
  - Relația `@ManyToOne` cu `Admin` arată cine a creat sondajul.
- **Vote:** O opțiune într-un sondaj.
  - Relația `@ManyToOne` cu `Poll` o leagă de sondajul părinte.
  - `listUsers`: O colecție de ID-uri de utilizatori care au votat pentru această opțiune.
  - `totalBets`: Suma totală a jetoanelor pariate pe această opțiune.

- **UserBet:** O entitate esențială care leagă un utilizator de un vot specific și de suma pariată. Aceasta este crucială pentru a calcula corect recompensele proporționale la rezolvarea sondajului. Conține `userId`, `voteId`, și `betAmount`.
- **BlogPost** și **Comments:** Structuri clasice pentru un sistem de blog, cu relații ierarhice.

2.1.3 Logica de Business (Service Layer)

Exemplu: Logica de Rezolvare a unui Sondaj (`PollService.resolvePoll`) Acesta este unul dintre cele mai complexe procese de business:

1. **Validări Inițiale:** Serviciul verifică dacă sondajul există și dacă a expirat.
2. **Identificarea Opțiunii Câștigătoare:** Se preia opțiunea câștigătoare pe baza ID-ului furnizat de administrator.
3. **Calcularea Fondurilor (Pools):** Se calculează două sume totale: `winnerPool` (suma jetoanelor pariate pe opțiunea câștigătoare) și `loserPool` (suma jetoanelor de pe toate celelalte opțiuni).
4. **Distribuirea Recompenselor:** Se iterează prin lista de utilizatori care au ales opțiunea corectă. Pentru fiecare utilizator câștigător:
   (a) Se caută în tabela `user_bets` pariul exact plasat de acel utilizator pe acea opțiune.
   (b) Se calculează procentul pe care pariul său îl reprezintă din `winnerPool`.
   (c) Câștigul său este calculat ca: *pariul_initial + (procentul_detinut * loserPool)*.
   (d) Punctele câștigate sunt adăugate în contul utilizatorului.
5. **Finalizarea:** Sondajul este marcat ca `resolved = true` și `winningOptionId` este setat, pentru a nu mai putea fi rezolvat a doua oară.

2.1.4 Securitate: Autentificare cu JWT

1. **Login:** Utilizatorul trimite email și parolă la `/auth/login`. `AuthService` validează credențialele. Dacă sunt corecte, `JwtUtil` generează un token JWT care conține email-ul utilizatorului (`subject`) și o dată de expirare.
2. **Stocare Client:** Token-ul este stocat pe client (în `localStorage`).
3. **Cereri Autentificate:** Pentru fiecare cerere către un API protejat (ex: `/api/posts`), clientul adaugă header-ul: `Authorization: Bearer <token>`.
4. **Validare Server:** `JwtAuthenticationFilter` interceptează cererea, extrage token-ul, îl validează folosind `JwtUtil`. Dacă token-ul este valid, extrage email-ul și setează un obiect `Authentication` în `SecurityContextHolder`, făcând utilizatorul "logat" pentru durata acelei cereri.

2.2 Frontend (React)

2.2.1 Managementul Stării Globale și Autentificarea

UserContext.jsx: Este inima managementului de sesiune în frontend.

- **Inițializare:** La încărcarea aplicației, un hook `useEffect` este declanșat. Acesta citește token-ul din `localStorage`.
- **Decodare și Validare:** Dacă token-ul există, este decodat cu `jwt-decode` pentru a-i verifica data de expirare. Dacă a expirat, se apelează `logout()`.
- **Hidratarea Datelor:** Dacă token-ul este valid, contextul face o cerere la backend la `/api/users/email/{email}` pentru a prelua obiectul complet al utilizatorului (cu puncte, rol, etc.). Acest proces se numește "hidratare".
- **Furnizarea Datelor:** Contextul expune datele utilizatorului (`user`), token-ul (`token`), starea de încărcare (`loading`) și funcțiile de `login` și `logout` către întreaga aplicație.

2.2.2 Componente Complexe și Stare Locală

PostCard.jsx Această componentă demonstrează o bună gestiune a stării locale și interacțiunea cu API-ul.

- **Stări Locale:** Folosește `useState` pentru a gestiona vizibilitatea secțiunii de comentarii (`isCommentSectionVisible`), starea de editare (`isEditing`), conținutul editat (`editedBody`), etc.
- **Actualizări Optimiste:** În funcția `handleLike`, se aplică o tehnică de "actualizare optimistă". Interfața este actualizată instantaneu (contorul de like-uri se schimbă, butonul își schimbă starea) *înainte* ca răspunsul de la server să sosească. Dacă API-ul returnează o eroare, UI-ul este readus la starea inițială. Acest lucru oferă o experiență de utilizare mult mai fluidă.
- **Delegarea Evenimentelor:** Componenta primește funcții ca props (ex: `onDeletePost`, `onUpdatePost`) de la componenta părinte (`Feed.jsx`). Când o acțiune este efectuată, `PostCard` apelează aceste funcții pentru a notifica părintele să actualizeze lista globală de postări.

3 Diagrame UML

3.1 Diagrame Use-Case

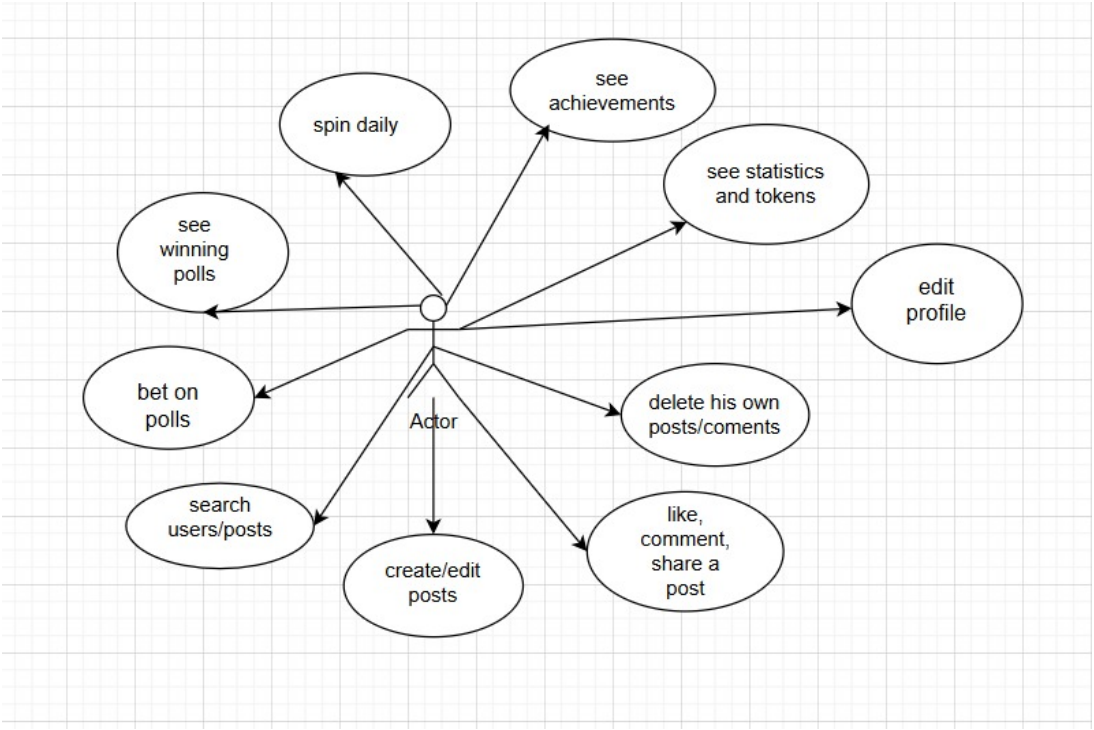

Figure 2: Diagrama Use-Case pentru un utilizator standard (student)

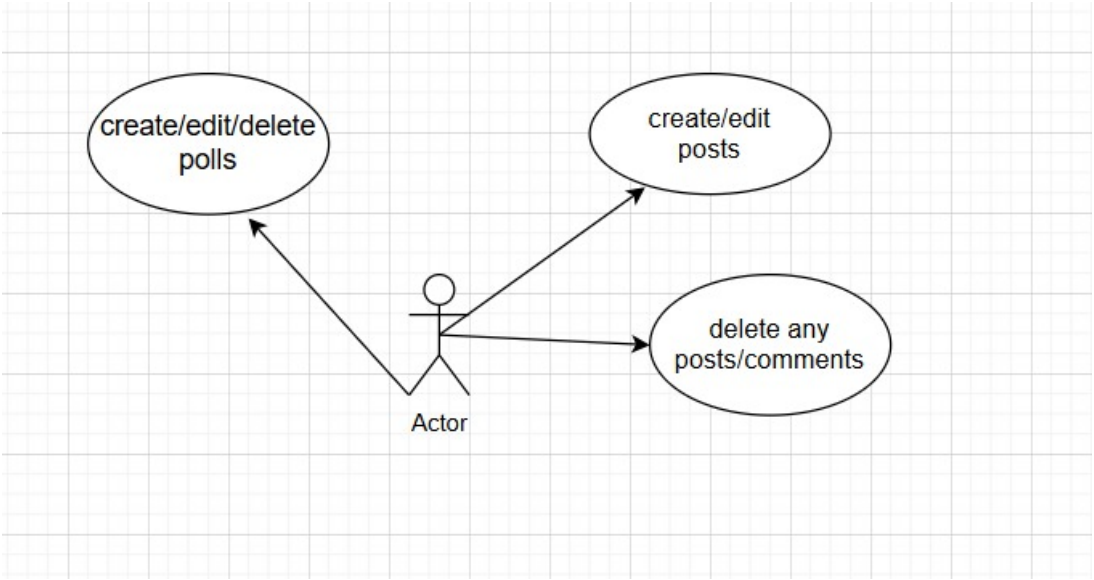

Figure 3: Diagrama Use-Case pentru un administrator

3.2 Diagrama de Pachete

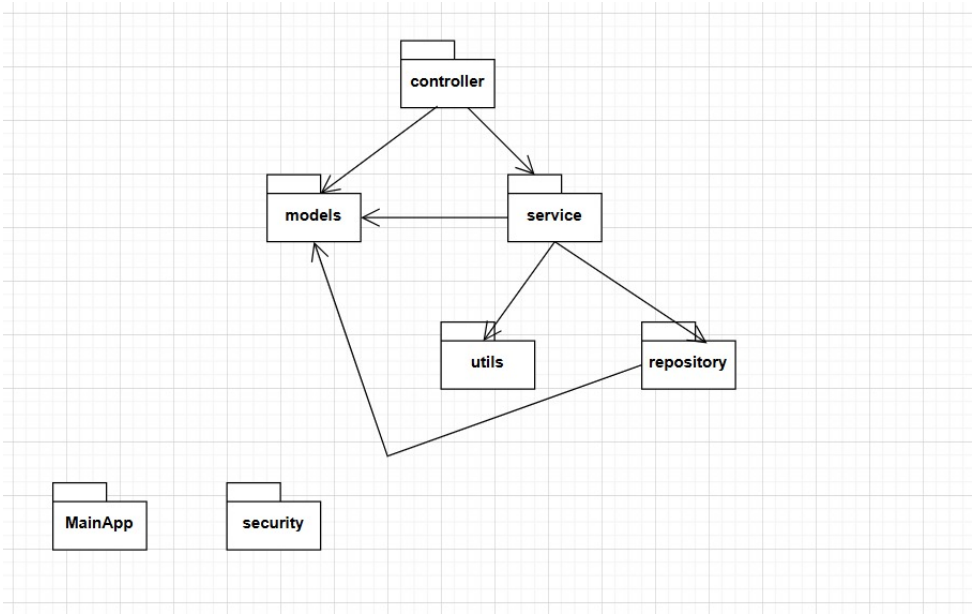

Figure 4: Diagrama de pachete

3.3 Diagrama de Clase

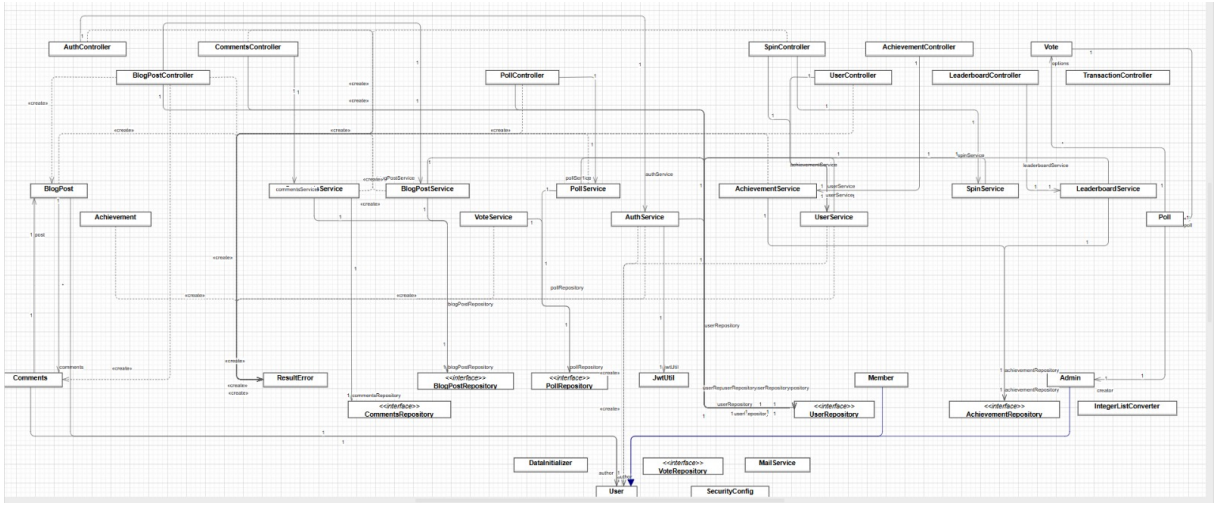

Figure 5: Diagrama de clase

4 Configurare și Rulare

4.1 Cerințe Preliminare

- Java 17 sau o versiune superioară
- Node.js și npm
- Un server de baze de date MySQL

4.2 Configurare Backend

1. Navigați în directorul `UTCN-POLL/backend`.
2. Asigurați-vă că serverul MySQL este pornit și că schema `utcn_poll` a fost creată.
3. Actualizați detaliile de conexiune la baza de date în fișierul `src/main/resources/application.prope`

```
spring.datasource.url=jdbc:mysql://localhost:3306/utcn_poll
spring.datasource.username=root
spring.datasource.password=parola
spring.jpa.hibernate.ddl-auto=update
```

Listing 1: Configurare bază de date în application.properties

Rulați comanda din rădăcina folderului `backend`:

```
./mvnw spring-boot:run
```

4.3 Configurare Frontend

1. Navigați în directorul `UTCN-POLL/my-app`.
2. Instalați dependențele:

```
npm install
```

Porniți serverul de dezvoltare:

```
npm start
```

Aplicația frontend va fi accesibilă la adresa `http://localhost:3000`.

5 Concluzii

Proiectul UTCN-POLL demonstrează o implementare robustă a unei aplicații web full-stack, utilizând tehnologii moderne și principii de design software consacrate. Prin separarea clară a logicii de backend de interfața de frontend, sistemul este modular, scalabil și ușor de întreținut. Logica de business complexă, cum ar fi sistemul de pariuri și recompense, a fost implementată cu atenție în stratul de servicii, în timp ce frontend-ul oferă o experiență de utilizare reactivă și plăcută prin managementul eficient al stării și actualizări optimiste. Documentația de față a acoperit atât viziunea de ansamblu, cât și detaliile tehnice esențiale, servind ca un ghid complet pentru înțelegerea și dezvoltarea ulterioară a platformei.