|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ  DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI |  |

PROIECT DE DIPLOMĂ

Janina Constantina Cocei

COORDONATOR ȘTIINȚIFIC

Ș. L. Dr. Ing. Cerbulescu Cătălin

Iulie 2025

CRAIOVA

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  FACULTATEA DE AUTOMATICĂ, CALCULATOARE ȘI ELECTRONICĂ  DEPARTAMENTUL DE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI |  |

CampusConnect. Aplicație web pentru chat

Janina Constantina Cocei

COORDONATOR ȘTIINȚIFIC

Ș. L. Dr. Ing. Cerbulescu Cătălin

Iulie 2025

CRAIOVA

*„Sub mâna celui învățat, uneltele devin punți spre zări de cunoaștere.”*

Autor necunoscut

**DECLARAȚIE DE ORIGINALITATE**

Subsemnatul Janina Constantina Cocei, student la specializarea CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI din cadrul Facultății de Automatică, Calculatoare și Electronică a Universității din Craiova, certific prin prezenta că am luat la cunoştinţă de cele prezentate mai jos şi că îmi asum, în acest context, originalitatea proiectului meu de licenţă:

* cu titlul CampusConnect. Aplicație web pentru chat,
* coordonată de Ș. L. Dr. Ing. Cerbulescu Cătălin
* prezentată în sesiunea IULIE 2025.

La elaborarea proiectului de licenţă, se consideră plagiat una dintre următoarele acţiuni:

* reproducerea exactă a cuvintelor unui alt autor, dintr-o altă lucrare, în limba română sau prin traducere dintr-o altă limbă, dacă se omit ghilimele şi referinţa precisă,
* redarea cu alte cuvinte, reformularea prin cuvinte proprii sau rezumarea ideilor din alte lucrări, dacă nu se indică sursa bibliografică,
* prezentarea unor date experimentale obţinute sau a unor aplicaţii realizate de alţi autori fără menţionarea corectă a acestor surse,
* însuşirea totală sau parţială a unei lucrări în care regulile de mai sus sunt respectate, dar care are alt autor.

Pentru evitarea acestor situaţii neplăcute se recomandă:

* plasarea între ghilimele a citatelor directe şi indicarea referinţei într-o listă corespunzătoare la sfărşitul lucrării,
* indicarea în text a reformulării unei idei, opinii sau teorii şi corespunzător în lista de referinţe a sursei originale de la care s-a făcut preluarea,
* precizarea sursei de la care s-au preluat date experimentale, descrieri tehnice, figuri, imagini, statistici, tabele et caetera,
* precizarea referinţelor poate fi omisă dacă se folosesc informaţii sau teorii arhicunoscute, a căror paternitate este unanim cunoscută și acceptată.

Data, Semnătura candidatului,

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de Calculatoare și Tehnologia Informației | Aprobat la data de  …………………  Şef de departament,  Prof. dr. ing.  Marius BREZOVAN/  Comin IONETE/  Dorian COJOCARU |

**PROIECTUL DE DIPLOMĂ**

|  |  |
| --- | --- |
| Numele și prenumele studentului/-ei: | Cocei Janina Constantina |
| Enunțul temei: | CampusConnect. Aplicatie web pentru chat |
| Datele de pornire: | Pentru dezvoltarea aplicației CampusConnect, am avut ca punct de plecare:   * analiza unor aplicații de comunicare deja existente (precum WhatsApp, Discord sau Teams), pentru a identifica funcționalitățile esențiale într-un mediu colaborativ; * interesul crescut pentru digitalizarea interacțiunii în spațiul academic; * cunoștințele tehnice dobândite în timpul facultăți. |
| Conținutul proiectului: | [*Descrierea succintă a conținutului fiecărui capitol al lucrării*] |
| Material grafic obligatoriu: | Diagrame, scheme, capturi de ecran. |
| Consultații: | Periodice |
| Conducătorul științific  (titlul, nume și prenume, semnătura): | Ș. L. Dr. Ing. Cerbulescu Cătălin |
| Data eliberării temei: | 15.10.2024 |
| Termenul estimat de predare a proiectului: | 01.06.2025 |
| Data predării proiectului de către student și semnătura acestuia: |  |

|  |  |  |
| --- | --- | --- |
|  | UNIVERSITATEA DIN CRAIOVA  Facultatea de Automatică, Calculatoare şi Electronică  Departamentul de [Calculatoare și Tehnologia Informației / Automatică și Electronică / Mecatronică și Robotică] |  |

**REFERATUL CONDUCĂTORULUI ȘTIINȚIFIC**

|  |  |
| --- | --- |
| Numele și prenumele candidatului/-ei: |  |
| Specializarea: | [*Denumirea oficială a specializării absolvite de candidat*] |
| Titlul proiectului: | [*Titlul lucrării*] |
| Locația în care s-a realizat practica de documentare (se bifează una sau mai multe din opțiunile din dreapta): | În facultate □ |
| În producție □ |
| În cercetare □ |
| Altă locație: [*se detaliază*] |

În urma analizei lucrării candidatului au fost constatate următoarele:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nivelul documentării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Tipul proiectului | | Cercetare  □ | Proiectare  □ | Realizare practică □ | Altul  [*se detaliază*] |
| Aparatul matematic utilizat | | Simplu  □ | Mediu  □ | Complex □ | Absent  □ |
| Utilitate | | Contract de cercetare □ | Cercetare internă □ | Utilare  □ | Altul  [*se detaliază*] |
| Redactarea lucrării | | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Partea grafică, desene | | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Realizarea practică | Contribuția autorului | Insuficientă  □ | Satisfăcătoare □ | Mare  □ | Foarte mare  □ |
| Complexitatea  temei | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Analiza cerințelor | Insuficient  □ | Satisfăcător □ | Bine  □ | Foarte bine  □ |
| Arhitectura | Simplă  □ | Medie  □ | Mare  □ | Complexă  □ |
| Întocmirea specificațiilor funcționale | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Implementarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Testarea | Insuficientă  □ | Satisfăcătoare □ | Bună  □ | Foarte bună  □ |
| Funcționarea | Da  □ | Parțială  □ | Nu  □ | |
| Rezultate experimentale | | Experiment propriu  □ | | Preluare din bibliografie  □ | |
| Bibliografie | | Cărți | Reviste | Articole | Referințe web |
| Comentarii  și  observații | |  | | | |

În concluzie, se propune:

|  |  |
| --- | --- |
| ADMITEREA PROIECTULUI  □ | RESPINGEREA PROIECTULUI  □ |

Data, Semnătura conducătorului științific,

**REZUMATUL PROIECTULUI**

În această secțiune sunt sumarizate elementele principale ale proiectului. Rezumatul proiectului are menirea de a da potențialilor cititori o imagine succintă a temei abordate și a motivației alegerii acesteia, a metodologiilor de cercetare și dezvoltare alese, precum și a tehnologiilor utilizate, a problemelor întâlnite pe parcursul realizării acesteia și modul de soluționare al acestora. Autorul trebuie să puncteze în mod clar rezultatele obținute prin contribuția personală, dar și lecțiile învățate pe parcursul realizării proiectului.

***Termenii cheie***: aplicație web, comunicare academică, digitalizare, platformă educațională, React, Spring Boot, mesagerie în timp real, partajare de fișiere.

**MULȚUMIRI**

Doresc să îmi exprim întreaga mea recunoștință față de conducătorul științific, Ș. L. Dr. Ing. Cerbulescu Cătălin, pentru îndrumarea competentă, răbdarea și sprijinul oferit pe parcursul elaborării acestei lucrări de licență. Profesionalismul și sfaturile domniei sale au avut un rol esențial în structurarea și finalizarea acestei etape importante din parcursul meu academic.

Mulțumiri deosebite adresez și cadrelor didactice din cadrul Facultății de Automatică, Calculatoare și Electronică, pentru formarea și cunoștințele transmise de-a lungul anilor de studiu, contribuind la dezvoltarea mea intelectuală și profesională.

De asemenea, doresc să mulțumesc familiei mele, pentru sprijinul necondiționat, încurajare și încrederea pe care mi-au oferit-o constant. Fără susținerea lor morală și emoțională, finalizarea acestei lucrări nu ar fi fost posibilă.

Tuturor celor care, direct sau indirect, au contribuit la realizarea acestei lucrări, le adresez sincere mulțumiri.

**CUPRINSUL**

[1 Introducere 1](#_Toc27175)

[1.1 Scopul 1](#_Toc32577)

[1.2 Motivația 1](#_Toc32370)

[2 TEHNOLOGII ȘI FRAMEWORK-URI UTILIZATE 2](#_Toc5713)

[2.1 ReactJS 2](#_Toc23438)

[2.2 Material UI (MUI) 2](#_Toc15797)

[2.3 Redux 3](#_Toc14424)

[2.4 React Router DOM 4](#_Toc285)

[2.5 Axios 5](#_Toc26870)

[2.6 Framer Motion 5](#_Toc26209)

[2.7 SockJS & STOMP.js 6](#_Toc7402)

[2.7.1 SockJS 6](#_Toc17381)

[2.7.2 STOMP.js 6](#_Toc10852)

[2.7.3 Scenarii de integrare în aplicații web 7](#_Toc29443)

[2.8 Recharts 7](#_Toc14719)

[2.9 Yup & React Hook Form 7](#_Toc1778)

[2.10 Microlink 8](#_Toc30223)

[2.11 Spring Boot 8](#_Toc13255)

[2.12 Spring Web 9](#_Toc25891)

[2.13 Spring Data JPA 9](#_Toc11895)

[2.14 Spring Security + JWT 9](#_Toc9074)

[2.15 Spring WebSocket 10](#_Toc22029)

[2.16 PostgreSQL 11](#_Toc15429)

[2.17 Lombok 11](#_Toc14194)

[2.18 Jakarta Validation 12](#_Toc1183)

[3 INSTRUMENTE software folosite 13](#_Toc18541)

[3.1 WebStorm 13](#_Toc1200)

[3.2 Redux DevTools 13](#_Toc23507)

[3.3 IntelliJ IDEA 14](#_Toc2518)

[3.4 Postman 14](#_Toc8184)

[3.5 PgAdmin 15](#_Toc3517)

[4 PROIECTAREA ȘI ARHITECTURA APLICAȚIEI 16](#_Toc16864)

[4.1 Prezentarea generala 16](#_Toc17238)

[4.2 Descrierea funcționalităților sistemului 16](#_Toc25281)

[4.3 Modelarea interacțiunii utilizatorilor prin diagrame 17](#_Toc24774)

[4.3.1 Introducere în modelarea UML 17](#_Toc15064)

[4.3.2 Caz de utilizare: Procesul de autentificare 18](#_Toc13683)

[4.3.3 Caz de utilizare: Rolul studentului 19](#_Toc23453)

[4.3.4 Caz de utilizare: Rolul tutorelui 20](#_Toc21981)

[4.3.5 Caz de utilizare: Funcții specifice administartorului 21](#_Toc21718)

[4.4 Structura bazei de date 21](#_Toc13033)

[4.4.1 Modelul conceptual și relațional 22](#_Toc23334)

[4.4.2 Tehnologii utilizate pentru stocare 22](#_Toc24766)

[4.4.3 Maparea obiect-relatională (ORM) 22](#_Toc17401)

[4.4.4 Considerații legate de performanță și consistență 23](#_Toc12656)

[4.4.5 Referințe bibliografice 24](#_Toc2566)

[5 IMPLEMENTAREA APLICAȚIEI 25](#_Toc10755)

[4.5 Inițializarea bazei de date 25](#_Toc26306)

[4.5.1 Crearea datelor inițiale prin seedere 25](#_Toc19601)

[4.5.2 Popularea dinamică la înregistrare 25](#_Toc12402)

[4.6 Autentificarea și gestionarea conturilor 26](#_Toc18583)

[5 Concluzii 6](#_Toc27069)

[6 Bibliografie 6](#_Toc5876)

[7 Referințe web 9](#_Toc12141)

[A. Codul sursă 10](#_Toc20068)

[B. Site-ul web al proiectului 11](#_Toc14473)

[C. CD / DVD 12](#_Toc25710)

[Index 13](#_Toc22848)

**LISTA FIGURILOR**

[Figură 1 - Fluxul procesului de autentificare 18](#_Toc12798)

[Figură 2 - Fluxul de utilizare pentru student 19](#_Toc3681)

[Figură 3 - Fluxul de utilizare pentru tutore 20](#_Toc14470)

[Figură 4 - Fluxul de utilizare pentru Admin 21](#_Toc31284)

[Figură 5 - Baza de date 23](#_Toc32128)

**LISTA TABELELOR**

**Error! No table of figures entries found.**

# Introducere

## Scopul

Scopul principal al acestei lucrări a fost dezvoltarea unei aplicații menite să sprijine îmbunătățirea comunicării în mediul academic, în special între studenți și între studenți și tutori. CampusConnect a fost concepută ca o platformă internă, dedicată comunității universitare pentru a promova schimbul de mesaje, fotografii, link-uri și documente într-un mod eficient, organizat și accesibil.

Dincolo de componenta practică, proiectul a reprezentat și o oportunitate valoroasă de a îmbunătății și aplica cunoștințele dobândite pe parcursul studiilor, în cadrul specializării Calculatoare, în limba română, la Facultatea de Automatică, Calculatoare și Electronică din Craiova. Procesul de dezvoltare a presupus implicare activă în toate etapele: de la analiză și proiectare, până la implementare și testare.

Redactarea acestei lucrări a avut ca scop documentarea detaliată a întregului parcurs tehnic – de la alegerile arhitecturale și tehnologice până la provocările întâmpinate pe parcurs. Această lucrare reflectă aplicarea concretă a cunoștințelor teoretice, subliniind dorința de a le extinde prin integrarea soluțiilor moderne și adaptându-se la nevoile reale ale mediului academic printr-un proces de digitalizare continuă.

## Motivația

Alegerea acestei teme a fost determinată de dorința de a contribui activ la procesul de digitalizare a comunicării în mediul academic. Într-un context în care interacțiunile dintre studenți și tutori devin mai complexe și dinamice, am identificat nevoia unei soluții moderne, care să faciliteze colaborarea într-un mod eficient și accesibil.

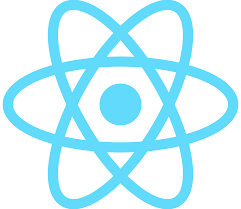
CampusConnect s-a conturat din convingerea că digitalizarea nu ar trebui să se limiteze doar la platformele educaționale formale, ci poate și ar trebui să se extindă către aspectele cotidiene ale vieții universitare. Printr-un sistem dedicat de mesagerie și partajare de resurse, aplicația își propune să simplifice și să organizeze mai bine comunicarea între membrii comunității academice.

Totodată, această temă a reprezentat pentru mine o oportunitate valoroasă de dezvoltare personală și profesională. Realizarea unui proiect complet, cu aplicabilitate concretă, mi-a oferit șansa de a aprofunda cunoștințele dobândite în timpul facultății și de a le pune în practică într-un mod relevant, tehnic și util pentru mediul universitar.

# TEHNOLOGII ȘI FRAMEWORK-URI UTILIZATE



## ReactJS

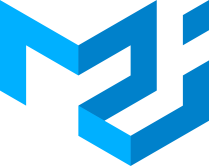
 React este o bibliotecă JavaScript care este utilizată pentru construirea interfețelor de utilizator, atât web cât și native (Inc. F. , 2023). Aplicațiile dezvoltate cu React sunt construite din componente independente, fiecare responsabilă de o anumită parte a interfeței, de la un simplu buton până la o pagină completă. (Inc. F. , 2023) Această abordare simplifică logica aplicației și facilitează testarea, mentenanța și extinderea codului.

Unul dintre principiile fundamentale ale React este Virtual DOM (Document Object Model). În loc să interacționeze direct cu DOM-ul real, React utilizează o copie virtuală, ceea ce reduce semnificativ numărul de manipulări directe ale DOM-ului. Acest lucru îmbunătățește performanța aplicațiilor, deoarece React efectuează un proces de reconciliare prin care identifică diferențele între starea anterioară și cea curentă a interfeței, actualizând doar elementele afectate.

Componentele în React sunt reprezentate fie prin componente funcționale (funcții JavaScript care returnează JSX), fie prin clase (în versiunile mai vechi). În prezent, componentele funcționale sunt preferate datorită suportului pentru React Hooks, introdus în versiunea 16.8. (Abramov D. , 2019) Acesta permite gestionarea stării și a efectelor colaterale direct în componentele funcționale, fără a fi necesare clase.

Ecosistemul React este susținut de numeroase biblioteci complementare precum Redux (pentru gestionarea stării globale), React Router (pentru rutare), MUI (Material UI – pentru interfețe vizuale) etc. (Banks A. &., 2020)

## Material UI (MUI)

MUI, cunoscut anterior sub denumirea de Material-UI, este o bibliotecă de componente UI pentru React, construită pe baza principiilor de design promovate de Material Design, un sistem de design dezvoltat de Google. (Inc. G. , 2023) Scopul acestei biblioteci este de a facilita dezvoltarea interfețelor moderne, coerente și accesibile, accelerând procesul de dezvoltare frontend prin oferirea de componente predefinite, personalizabile și bine documentate. (Team M. , 2024)

MUI oferă o gamă largă de componente, de exemplu: butoane, formulare, tabele, bare de navigare, dialoguri, alerte, etc. toate implementate ca și componente React. Acestea respectă standardele de accesibilitate (ARIA) și sunt optimizate pentru performanță și responsivitate. (Larsen, 2021)

Arhitectura modulară a MUI permite importul selectiv al componentelor necesare, contribuind la optimizarea performanței aplicației prin reducerea dimensiunii bundle-ului final. De asemenea, biblioteca este compatibilă cu instrumente moderne de styling precum Emotion (biblioteca implicită de CSS in JS), dar oferă și opțiuni pentru integrarea cu alte soluții de stilizare, cum ar fi Styled Components sau Tailwind CSS. (Team M. , 2024)

MUI este utilizat pe scară largă în aplicații comerciale și academice datorită documentației extinse, comunității active și integrării excelente cu ecosistemul React. (Larsen, 2021)

## Redux

Redux este o bibliotecă JavaScript open-source utilizată în mod obișnuit pentru gestionarea stării aplicațiilor frontend, în special în cadrul ecosistemului React, dar nu exclusiv. A fost dezvoltată inițial de Dan Abramov și Andrew Clark în 2015 și se bazează conceptual pe modelul fluxului unidirecțional de date inspirat de arhitectura Flux dezvoltată de Facebook. (Abramov D. ș., 2024)

Principiul de bază al Redux este acela de a menține o singură sursă de adevăr („single source of truth”) prin intermediul unui „store” global în care este salvată întreaga stare a aplicației. Interacțiunea cu această stare se realizează prin acțiuni („actions”) care sunt procesate de funcții pure denumite „reduceri” („reducers”). Acest model determină o predictibilitate crescută a comportamentului aplicației și facilitează testarea și depanarea codului. (Ng, 2017)

Un flux simplificat în Redux implică următoarele etape:

* **Dispatcharea unei acțiuni** – un obiect JavaScript care descrie o intenție de modificare a stării.
* **Procesarea acțiunii de către reducer** – o funcție care primește starea curentă și acțiunea și returnează o nouă stare.
* **Actualizarea store-ului** – noua stare este salvată în store și componentele abonate sunt notificate.

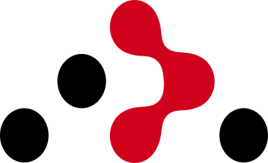
Redux încurajează o arhitectură predictibilă și scalabilă, de aceea este preferat în aplicații complexe unde starea aplicației este distribuită pe multiple componente.

Un avantaj important al utilizării Redux constă în facilitarea funcționalităților avansate precum:

* time-travel debugging (prin redux-devtools),
* serializarea și rehidratarea stării,
* logarea și urmărirea modificărilor de stare. (Alex Banks, 2016)

Deși a fost criticat uneori pentru boilerplate-ul necesar, apariția unor extensii precum redux-thunk, redux-saga sau redux-toolkit a simplificat considerabil scrierea și gestionarea logicii aplicațiilor moderne. (Ng, 2017)

## React Router DOM

**React Router DOM** este o bibliotecă esențială pentru gestionarea rutării în aplicațiile dezvoltate cu **React**, oferind suport pentru navigarea între pagini fără reîncărcarea completă a aplicației. Aceasta implementează funcționalități de tip Single Page Application (SPA), permițând comutarea dinamică între diferite componente sau views, pe baza adresei URL, fără pierderea stării aplicației. (Inc. R. S., 2024)

Această bibliotecă este construită pe baza arhitecturii **React Router**, dar adaptată pentru medii bazate pe DOM, cum ar fi aplicațiile web. Începând cu versiunea 6, React Router a introdus o serie de îmbunătățiri semnificative, inclusiv un API[[1]](#footnote-0) declarativ mai intuitiv, suport îmbunătățit pentru *nested routes* [[2]](#footnote-1) și utilizarea data loaders [[3]](#footnote-2) pentru preluarea datelor înainte de randarea componentelor. (Fulton, 2022)

Componentele de bază oferite de React Router DOM includ:

* BrowserRouter: componentă care folosește API-ul HTML5 pentru a păstra sincronizarea între interfață și URL.
* Routes și Route: definesc rutele aplicației și componentele asociate.
* Link: permite navigarea internă fără reîncărcare.
* useNavigate, useParams, useLocation: hook-uri pentru manipularea programatică a rutării și accesarea parametrilor din URL. (Banks R. &., 2023)

React Router DOM contribuie astfel la crearea unei experiențe de utilizare fluentă și interactivă, fiind un instrument indispensabil în dezvoltarea aplicațiilor moderne React care necesită navigare multiplă, autentificare pe bază de rute și gestionare de layout-uri dinamice.

## Axios

IMG_256**Axios** este o bibliotecă JavaScript open-source utilizată pentru efectuarea cererilor HTTP din aplicații web sau mobile. Este construită pe baza **Promise API**, ceea ce o face ideală pentru lucrul asincron, în special în cadrul aplicațiilor dezvoltate cu **React**, **Vue**, **Angular** sau alte framework-uri JavaScript moderne. (Contributors, 2024)

Axios oferă o interfață simplificată pentru trimiterea cererilor de tip GET, POST, PUT, DELETE și altele, către servere RESTful, facilitând comunicarea între client și server într-un mod scalabil. Datorită suportului nativ pentru interceptoare, Axios permite tratarea globală a erorilor, adăugarea de token-uri de autentificare în antetul cererilor, sau logarea răspunsurilor.

Funcționalități esențiale oferite de Axios includ:

* Suport pentru transformarea automată a datelor JSON.
* Configurarea implicită a antetelor HTTP.
* Compatibilitate cu API-urile browserelor moderne și cu Node.js. (Zhang, 2022)

Axios este utilizat frecvent în aplicațiile care comunică cu un backend REST, precum cele dezvoltate cu **Spring Boot**, deoarece facilitează trimiterea de date serializate, gestionarea sesiunilor sau a token-urilor JWT și manipularea răspunsurilor de la server într-un mod eficient.

## Framer Motion

**Framer Motion** este o bibliotecă de animații declarative pentru React, dezvoltată de echipa Framer. Aceasta permite dezvoltatorilor să creeze tranziții și animații fluide într-un mod intuitiv și performant, folosind sintaxa JSX familiară. Este considerată una dintre cele mai populare și moderne soluții pentru animarea interfețelor web în aplicații React, fiind potrivită atât pentru animații simple (fade, slide, scale), cât și pentru interacțiuni complexe și micro-animații. (Fitzgerald, 2021)

**Caracteristici principale:**

* **API declarativ și intuitiv**: Permite definirea animațiilor direct în JSX, fără a apela imperativele CSS sau JavaScript.
* **Performanță ridicată**: Utilizează requestAnimationFrame și layout animations optimizate pentru redare fluidă.
* **Integrare ușoară cu React**: Se integrează natural în fluxul aplicației, fără a fi nevoie de logică suplimentară complexă. (Framer, 2024)

## SockJS & STOMP.js

Pentru aplicațiile moderne care necesită actualizări în timp real, cum ar fi aplicațiile de chat, colaborare sau notificări live, utilizarea WebSocket-urilor este esențială. Totuși, din cauza limitărilor de compatibilitate cu anumite browsere sau rețele, au fost dezvoltate soluții de fallback care să mențină o conexiune bidirecțională între client și server. Printre cele mai populare soluții se numără SockJS, în combinație cu protocolul STOMP, accesat prin biblioteca STOMP.js.

### ****SockJS****

**SockJS** este o bibliotecă JavaScript care oferă o interfață WebSocket, dar cu fallback-uri automate către alte protocoale atunci când WebSocket-ul nu este disponibil. Aceasta asigură compatibilitate extinsă și o conexiune fiabilă între client și server, indiferent de condițiile de rețea sau browser. (SockJs, 2024)

**Caracteristici principale:**

* Oferă fallback la protocoale precum XHR-streaming, XHR-polling, EventSource etc.
* Funcționează fără întreruperi în medii cu restricții de rețea (ex: firewall-uri, proxy-uri).
* Utilizată frecvent în aplicații în care fiabilitatea conexiunii în timp real este critică.

### ****STOMP.js****

**STOMP (Simple Text Oriented Messaging Protocol)** este un protocol simplu, bazat pe text, pentru comunicarea între clienți și servere prin mesaje. **STOMP.js** este o bibliotecă JavaScript care implementează acest protocol, permițând frontend-ului să interacționeze cu un server STOMP (de exemplu, Spring WebSocket) într-un mod standardizat. (WebSocket Support, n.d.) (STOMP, n.d.)

**Funcționalități cheie:**

* Abonarea la canale (topics) și trimiterea de mesaje către destinații (queues/topics).
* Suport pentru reconectare și heartbeat-uri.
* Permite o separare clară între logica de transport (SockJS/WebSocket) și modelul de mesagerie.

### ****Scenarii de integrare în aplicații web****

Combinația **SockJS + STOMP.js** este utilizată frecvent în aplicații bazate pe **Spring Boot**, care oferă suport nativ pentru WebSocket și STOMP prin modulul spring-websocket. Această soluție permite dezvoltarea de aplicații cu funcționalități avansate de comunicare în timp real, cum ar fi:

* transmiterea de mesaje în timp real;
* notificări live;
* actualizări sincronizate între utilizatori.

## Recharts

**Recharts** este o bibliotecă de vizualizare a datelor construită special pentru React, bazată pe **D3.js**. Aceasta permite dezvoltatorilor să creeze cu ușurință grafice declarative, reutilizabile și responsive. Sintaxa sa simplificată, component-based, se aliniază cu paradigma React, oferind în același timp o flexibilitate ridicată pentru personalizarea graficelor. (Recharts, n.d.)

Caracteristici principale:

* Grafică SVG pentru o randare clară și scalabilă;
* Suport pentru o varietate de tipuri de grafice: linii, bare, pie chart, area chart, radar etc.;
* Ușor de personalizat cu ajutorul props-urilor și al stilizării dinamice

## Yup & React Hook Form

**React Hook Form** este o bibliotecă JavaScript open-source destinată gestionării formularelor în aplicațiile React, recunoscută pentru simplitatea sa, performanța ridicată și integrarea nativă cu mecanismul de *hooks* introdus în React 16.8. Aceasta permite înregistrarea ușoară a câmpurilor de formular, validarea acestora și controlul erorilor, totul cu un minim de re-randăr. (Form, 2024)

Biblioteca se remarcă printr-un API simplificat, folosind funcții precum useForm(), register(), handleSubmit() sau formState, oferind o alternativă performantă față de metodele tradiționale de gestionare a stării formularelor în React.

Pentru validare, React Hook Form poate fi integrat cu diverse biblioteci externe, printre care cea mai populară este Yup, o bibliotecă de validare schema-based pentru JavaScript. (Jquense, 2023) Yup permite definirea unor reguli de validare declarative, sub formă de *schema objects*, ceea ce contribuie la o logică clară și reutilizabilă de validare a datelor.

## Microlink

Microlink este un serviciu și o bibliotecă JavaScript open-source care permite generarea de previzualizări pentru link-uri printr-o singură cerere HTTP. Aceasta returnează informații precum titlul paginii, imaginea principală, descrierea, favicon-ul și alte elemente relevante, folosind standarde precum Open Graph sau Twitter Cards. Microlink este potrivită pentru aplicații moderne ce necesită îmbogățirea conținutului text cu elemente vizuale dinamice, fiind ușor de integrat atât pe backend, cât și direct în frontend. (Microlink, 2024)

## Spring Boot

Spring Boot este un framework open-source dezvoltat de Pivotal Software, parte a ecosistemului Spring, care facilitează crearea rapidă a aplicațiilor Java enterprise. Scopul principal al Spring Boot este de a simplifica procesul de dezvoltare și configurare a aplicațiilor backend, oferind o abordare opinată („opinionated”) care elimină necesitatea configurațiilor XML extinse sau setărilor complicate. (Walls, 2022)

Spring Boot oferă suport pentru o serie de caracteristici esențiale în dezvoltarea modernă de aplicații web, printre care:

* **Embedded servers** (Tomcat, Jetty, Undertow), care permit rularea aplicației fără a instala un server separat.
* **Autoconfiguration**, prin care framework-ul configurează automat componentele necesare pe baza dependențelor din proiect.
* **Starter dependencies**, care grupează automat bibliotecile necesare pentru diverse funcționalități (ex: spring-boot-starter-web, spring-boot-starter-data-jpa).

Aplicațiile Spring Boot sunt adesea utilizate pentru dezvoltarea de API-uri RESTful, datorită integrării excelente cu Spring MVC și suportului pentru serializare JSON, validare, autentificare și interacțiune cu baze de date prin JPA/Hibernate. (Long, 2021)

Datorită simplității sale și integrării cu instrumente moderne precum Spring Security, Spring Data, Spring Cloud și altele, Spring Boot a devenit una dintre cele mai utilizate tehnologii backend în industrie și în proiecte academice, oferind o bază solidă pentru aplicații scalabile și ușor de întreținut. (Long, 2021)

## Spring Web

Spring Web este un modul esențial din cadrul ecosistemului Spring Framework, destinat construirii de aplicații web robuste, scalabile și ușor de întreținut. Acesta oferă suport complet pentru dezvoltarea aplicațiilor web tradiționale și a serviciilor RESTful, prin integrarea cu module precum Spring MVC (Model-View-Controller).

Componenta principală a Spring Web este Spring MVC, un framework care urmează arhitectura de tip model-vizualizare-controler. Acesta permite separarea clară a logicii de afaceri de interfața utilizator și oferă un control fin asupra fluxului de date și a comportamentului aplicației. (Walls, 2022)

## Spring Data JPA

Spring Data JPA este un submodul din ecosistemul Spring Data, conceput pentru a simplifica interacțiunea dintre aplicațiile Java și bazele de date relaționale. Acesta oferă o abstracție de nivel înalt peste Java Persistence API (JPA), permițând dezvoltatorilor să creeze rapid și eficient straturi de persistență printr-un model declarativ bazat pe interfețe.

Unul dintre avantajele majore ale Spring Data JPA este posibilitatea de a defini interfețe de tip Repository (precum JpaRepository, CrudRepository sau PagingAndSortingRepository) fără a fi necesară implementarea explicită a metodelor CRUD (Create, Read, Update, Delete), acestea sunt generate automat la runtime de către framework. (Meier, 2022) În plus, Spring Data JPA permite:

* Crearea de interogări derivate pe baza convenției de denumire a metodelor (findByName, findByEmailAndStatus etc.).
* Utilizarea interogărilor personalizate cu **JPQL** sau **SQL nativ** prin adnotarea @Query.
* Gestionarea relațiilor dintre entități (OneToMany, ManyToOne, etc.) și a tranzacțiilor.

Prin automatizarea operațiunilor repetitive și expunerea unui API intuitiv, Spring Data JPA contribuie la reducerea cantității de cod boilerplate, sporind astfel productivitatea și lizibilitatea aplicațiilor enterprise (Schmidt, 2021). (Schmidt, 2021)

## Spring Security + JWT

Spring Security este un framework puternic și extensibil din ecosistemul Spring, destinat autentificării și autorizării în aplicațiile Java. Acesta oferă un set complet de mecanisme pentru securizarea aplicațiilor web și RESTful, fiind compatibil cu standarde moderne precum OAuth2, JWT și LDAP. (Walls, 2022)

În contextul aplicațiilor REST, o abordare des utilizată este securizarea prin JSON Web Token (JWT). JWT este un standard deschis (RFC 7519) care permite transmiterea sigură a informațiilor între client și server sub forma unui obiect JSON. (Jones, 2015) Într-un flux tipic de autentificare JWT cu Spring Security:

* Utilizatorul trimite datele de autentificare către un endpoint (/login sau /authenticate).
* Serverul validează credențialele și generează un token JWT, semnat criptografic.
* Clientul stochează acest token (de regulă în localStorage sau sessionStorage) și îl atașează în antetul Authorization: Bearer al fiecărei cereri viitoare.
* Serverul validează token-ul și extrage informațiile despre utilizator (fără a menține o sesiune pe server), implementând astfel o arhitectură stateless.

Această abordare este preferată pentru aplicații distribuite, microservicii sau aplicații mobile, datorită scalabilității și separării dintre client și server. (Rahman, 2021) În Spring Security, integrarea JWT se face de obicei printr-un filtru personalizat (OncePerRequestFilter) care validează token-ul și setează contextul de securitate (SecurityContextHolder).

Combinația Spring Security + JWT permite o securizare robustă și flexibilă, fără necesitatea gestionării sesiunilor pe server, facilitând dezvoltarea aplicațiilor moderne, conforme cu cerințele actuale de performanță și siguranță.

## Spring WebSocket

Spring WebSocket este un modul din ecosistemul Spring Framework care oferă suport pentru comunicarea bidirecțională, asincronă, între client și server, utilizând protocolul WebSocket. Acesta permite dezvoltarea de aplicații în timp real, cum ar fi chat-uri, dashboard-uri sau jocuri multiplayer, unde actualizarea imediată a datelor este esențială. (Walls, 2022)

Protocolul WebSocket oferă o conexiune persistentă între client și server, permițând trimiterea de mesaje în ambele direcții, fără a necesita o cerere HTTP separată pentru fiecare schimb de informații. Comparativ cu modelul tradițional request-response, WebSocket reduce latența și utilizarea resurselor serverului în aplicațiile interactive.

Spring WebSocket este construit peste specificația Java WebSocket API (JSR 356**)** și oferă, de asemenea, integrare cu STOMP (Simple Text Oriented Messaging Protocol), un protocol sub WebSocket care facilitează modelul de mesagerie publish-subscribe. Pentru gestionarea mesajelor, Spring utilizează o infrastructură bazată pe broker intern (SimpleBroker) sau extern (precum RabbitMQ), iar pentru configurarea rutelor și endpoint-urilor se utilizează adnotări precum @MessageMapping și @SendTo.

Un exemplu tipic de flux în Spring WebSocket:

* Clientul inițiază o conexiune WebSocket către un endpoint definit (/ws).
* După stabilirea conexiunii, se abonează la un topic (ex: /topic/messages).
* Serverul primește mesajele trimise de client (ex: pe /app/chat) și le difuzează altor clienți abonați la topicul respectiv.

Această arhitectură facilitează dezvoltarea unor aplicații scalabile și interactive, oferind un model de comunicare performant și eficient pentru aplicațiile moderne în timp real. (Mak, 2024)

## PostgreSQL

PostgreSQL este un sistem de gestiune a bazelor de date relaționale (RDBMS) open-source, avansat, care pune accent pe conformitatea cu standardele SQL, extensibilitate. A fost inițial dezvoltat la Universitatea din California, Berkeley, și continuă să fie întreținut de o comunitate activă de dezvoltatori sub licență PostgreSQL. (Group, 2024)

Printre caracteristicile definitorii ale PostgreSQL se numără:

* Extensibilitatea ridicată, oferind posibilitatea de a defini tipuri de date proprii, funcții, operatori și chiar limbaje de programare;
* Sistemul avansat de gestionare a permisiunilor, care permite control granular al accesului la resurse;
* Suport pentru JSON și XML, făcându-l potrivit atât pentru aplicații relaționale, cât și semi-structurate;

În contextul aplicațiilor moderne Java Spring Boot, PostgreSQL este adesea utilizat în combinație cu JPA/Hibernate, datorită compatibilității excelente și suportului pentru dialect PostgreSQL.

## Lombok

Lombok este o bibliotecă open-source pentru Java care automatizează generarea codului tipic (boilerplate) prin intermediul unor adnotări, simplificând astfel semnificativ dezvoltarea aplicațiilor Java. Aceasta este frecvent utilizată în cadrul aplicațiilor Spring Boot pentru a reduce cantitatea de cod necesară pentru definirea claselor model, DTO-urilor sau serviciilor. (Mokito, 2021)

Prin utilizarea adnotărilor precum @Getter, @Setter, @ToString, @EqualsAndHashCode, @NoArgsConstructor, @AllArgsConstructor sau @Builder, Lombok generează automat metodele corespunzătoare în timpul compilării, fără a le scrie explicit în codul sursă. Acest lucru conduce la o creștere a lizibilității codului și o întreținere mai eficientă a proiectului.

Un avantaj major îl reprezintă integrarea transparentă cu IDE-uri moderne precum IntelliJ IDEA sau Eclipse, datorită pluginurilor dedicate care permit recunoașterea și validarea codului generat de Lombok. (Project, 2024)

## Jakarta Validation

Jakarta Validation, cunoscut anterior sub numele de Bean Validation (JSR 380/303), este o specificație standardizată pentru validarea datelor în aplicațiile Java. Este parte a ecosistemului Jakarta EE și este utilizată frecvent în cadrul aplicațiilor Spring Boot, în special împreună cu Hibernate Validator, implementarea de referință a specificației. (EE, 2023)

Această tehnologie oferă un set de adnotări standard pentru a defini regulile de validare aplicabile asupra câmpurilor unei clase. Printre cele mai utilizate se numără:

* @NotNull, @NotBlank, @NotEmpty : pentru validarea prezenței valorilor;
* @Size: pentru dimensiuni minime și maxime ale colecțiilor, șirurilor de caractere sau array-uri;
* @Min, @Max, @Positive, @Negative: pentru validarea valorilor numerice;
* @Email, @Pattern: pentru validări pe expresii regulate sau email-uri.

Validarea se realizează în mod automat, de obicei la nivel de controller, prin adnotarea parametrului cu @Valid sau @Validated. În cazul unei erori de validare, Spring generează un răspuns HTTP corespunzător, ce poate fi gestionat prin mecanisme de tratare globală a excepțiilor (ex: @ControllerAdvice).

Un beneficiu major al Jakarta Validation este capacitatea de a extinde sistemul cu validatori personalizați prin implementarea interfeței ConstraintValidator, oferind astfel un cadru flexibil și robust pentru asigurarea integrității datelor. (Seacord, 2021)

# INSTRUMENTE software folosite



## WebStorm

WebStorm este un mediu de dezvoltare integrat (IDE) dezvoltat de compania JetBrains, specializat în dezvoltarea aplicațiilor front-end și full-stack, în special utilizând JavaScript și framework-urile moderne asociate (React, Angular, Vue etc.). Este considerat unul dintre cele mai avansate IDE-uri pentru dezvoltarea aplicațiilor web datorită suportului extins pentru tehnologii moderne, a integrării cu sisteme de control al versiunilor și a funcțiilor inteligente de completare a codului, refactorizare și depanare. (JetBrains, 2024)

WebStorm oferă o serie de funcționalități care îmbunătățesc semnificativ productivitatea dezvoltatorilor:

* **Debugging integrat**: instrumente de depanare pentru JavaScript, TypeScript și Node.js;
* **Suport pentru testare**: integrare cu framework-uri de testare precum Jest, Mocha sau Cypress;
* **Tooling modern**: suport pentru ESLint, Prettier, Git, Webpack, npm și alte instrumente esențiale;
* **Live Editing**: actualizare în timp real a aplicației în browser în timpul dezvoltării.

Un avantaj important al WebStorm este integrarea sa profundă cu ecosistemul **React**, oferind suport pentru JSX, React Hooks și chiar pentru TypeScript, ceea ce îl face ideal pentru aplicații moderne dezvoltate cu aceste tehnologii.

## Redux DevTools

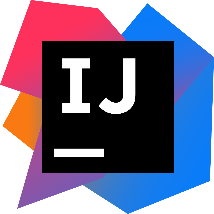
Redux DevTools este un instrument esențial pentru dezvoltatorii care utilizează biblioteca Redux în aplicații frontend, oferind facilități avansate pentru monitorizarea, inspectarea și depanarea stării aplicației. Acesta permite vizualizarea în timp real a stării globale (store), a acțiunilor (actions) și a reducerilor (reducers), fiind disponibil sub forma unei extensii pentru browserele moderne precum Chrome și Firefox. (DevTools, 2024)

Funcționalități principale:

* **Time-travel debugging** - permite navigarea între stările anterioare și urmărirea modificărilor în fluxul aplicației;
* **Vizualizarea acțiunilor** - fiecare acțiune dispecerizată este înregistrată, împreună cu starea aplicației de dinainte și de după;
* **Export/import de sesiuni** - utile pentru partajarea cazurilor de testare sau depanarea colaborativă;
* **Filtrarea și limitarea acțiunilor** - pentru o analiză mai eficientă a fluxului aplicației în aplicații mari.

Redux DevTools contribuie semnificativ la creșterea transparenței și predictibilității aplicației, facilitând procesul de testare și mentenanță, în special în aplicații complexe care utilizează o stare distribuită și numeroase interacțiuni asincrone.

## IntelliJ IDEA

**IntelliJ IDEA** este un mediu integrat de dezvoltare (Integrated Development Environment - IDE) dezvoltat de JetBrains, recunoscut pentru suportul său avansat în dezvoltarea aplicațiilor Java și a altor limbaje de programare. Este apreciat pentru facilitățile sale care cresc productivitatea dezvoltatorilor, printre care se numără suportul extins pentru refactorizare, analiză statică a codului, completare inteligentă, navigare rapidă și integrarea facilă cu sisteme de control al versiunilor precum Git.

IntelliJ IDEA oferă suport nativ pentru tehnologiile utilizate în dezvoltarea aplicațiilor backend Java, inclusiv Spring Boot, Hibernate, și alte framework-uri populare, facilitând astfel crearea, testarea și mentenanța aplicațiilor complexe. De asemenea, include instrumente puternice pentru depanare și gestionare a bazelor de date, ceea ce îl face o alegere preferată atât în mediile academice, cât și industriale.

Prin interfața intuitivă și numeroasele pluginuri disponibile, IntelliJ IDEA sprijină un flux de lucru eficient și colaborativ, contribuind la reducerea timpului necesar dezvoltării și îmbunătățirea calității codului. (JetBrains, 2025)

## Postman

**Postman** este o platformă populară utilizată pentru dezvoltarea, testarea și documentarea API-urilor (Application Programming Interfaces). Aceasta oferă o interfață prietenoasă și intuitivă pentru realizarea cererilor HTTP (GET, POST, PUT, DELETE etc.) și vizualizarea răspunsurilor primite de la servere, facilitând astfel procesul de testare și depanare a serviciilor web.

Postman suportă crearea de colecții de cereri, ceea ce permite organizarea și reutilizarea ușoară a testelor API, dar și automatizarea acestora prin intermediul scripturilor pre și post-request scrise în JavaScript. De asemenea, include instrumente pentru validarea răspunsurilor, gestionarea variabilelor de mediu și generarea de documentație API.

Prin integrarea cu sisteme de control al versiunilor și instrumente CI/CD, Postman devine un aliat important în fluxurile de dezvoltare moderne, sprijinind echipele în livrarea rapidă și calitativă a serviciilor web.

## PgAdmin

**PgAdmin** este o aplicație open-source de administrare și gestionare a bazelor de date PostgreSQL. Este cel mai popular instrument grafic folosit pentru administrarea serverelor și bazelor de date PostgreSQL, oferind o interfață intuitivă și ușor de utilizat pentru utilizatori de toate nivelurile.

Cu PgAdmin, dezvoltatorii și administratorii de baze de date pot crea, modifica și interoga baze de date, gestiona utilizatori și permisiuni, vizualiza structura și conținutul tabelelor, precum și executa scripturi SQL direct din interfața grafică. Instrumentul include facilități avansate precum monitorizarea performanței, generarea de rapoarte și suport pentru backup și restaurare.

PgAdmin este compatibil cu multiple platforme (Windows, macOS, Linux) și facilitează lucrul colaborativ prin opțiuni de configurare a conexiunilor și securității, fiind un instrument indispensabil în administrarea bazelor de date PostgreSQL.

# PROIECTAREA ȘI ARHITECTURA APLICAȚIEI



## Prezentarea generala

Dezvoltarea aplicației CampusConnect a fost ghidată de o serie de cerințe generale, menite să asigure utilitatea și performanța acesteia într-un mediu academic. Aplicația trebuie să răspundă nevoilor principale ale utilizatorilor (studenți și tutori) în ceea ce privește comunicarea, partajarea de resurse și colaborarea.

În cadrul procesului de analiză, au fost identificate următoarele cerințe generale:

* Aplicația trebuie să ofere un **sistem de autentificare** securizat;
* Comunicarea între utilizatori trebuie să fie **rapidă și eficientă**, cu suport pentru **mesagerie instant**;
* Sistemul trebuie să permită **transmiterea de fișiere**, inclusiv imagini și documente;
* Aplicația trebuie să fie **accesibilă din browser**, fără a necesita instalare locală;
* Interfața trebuie să fie **intuitivă**, prietenoasă și adaptabilă la diferite dimensiuni de ecran (responsive design);
* Datele utilizatorilor trebuie să fie stocate în siguranță, iar interacțiunile să fie gestionate în mod organizat;
* Sistemul trebuie să permită extinderea ulterioară, fiind **scalabil** și **modular**;

Aceste cerințe au fost conturate în urma analizei nevoilor concrete din mediul universitar, completate de consultarea unor studii de specialitate și bune practici din domeniul dezvoltării software. De asemenea, a existat dorința de a propune o soluție modernă, scalabilă, cu aplicabilitate reală în contextul educațional actual.

## Descrierea funcționalităților sistemului

Specificațiile funcționale definesc comportamentul așteptat al aplicației din perspectiva utilizatorului final. Acestea descriu în mod concret ce acțiuni poate realiza un utilizator și cum răspunde sistemul la acestea. CampusConnect oferă un set de funcționalități esențiale pentru susținerea unui mediu de comunicare eficient în cadrul comunității universitare.

Funcționalitățile principale implementate sunt:

* **Autentificare și înregistrare utilizatori:** Aceștia își pot crea un cont sau se pot autentifica utilizând doar mail-urile institutionale. Sistemul validează datele și oferă acces în funcție de rolul utilizatorului (student, tutore sau admin).
* **Gestionarea profilului de utilizator:** Fiecare utilizator are un profil care poate fi personalizat ( fotografie de profil, descriere etc ).
* **Mesagerie individuală și de grup:** Utilizatorii pot trimite și primi mesaje text în timp real, fie în conversații private, fie în cadrul grupurilor.
* **Crearea și administrarea grupurilor:** Tutorii pot crea grupuri tematice (ex. pe bază de materie sau proiecte), pot adăuga membri și pot gestiona setările grupului.
* **Partajarea de fișiere:** Aplicația permite trimiterea și descărcarea de imagini și documente (ex. PDF, Word), atât în conversații private, cât și în grupuri.
* **Căutare și filtrare conversații:** Utilizatorii au posibilitatea de a căuta conversații existente după nume sau de a iniția altele noi. De asemenea, conversațiile importante pot fi evidențiate prin funcționalitatea de „fixare” (pinned), fiind afișate prioritar în partea superioară a listei de conversații.
* **Marcarea mesajelor importante:** Este posibilă evidențierea unor mesaje prin opțiuni precum „starred” (favorit), pentru o accesare mai rapidă ulterioară.
* **Vizualizarea și organizarea conținutului media:** Toate mesajele ce conțin imagini, fișiere, documente sau linkuri sunt grupate într-o secțiune dedicată pentru fiecare conversație, facilitând astfel accesul rapid la aceste resurse.
* **Previzualizarea conținutului linkurilor:** Linkurile trimise în conversații pot fi afișate cu o previzualizare a conținutului.
* **Funcționalități avansate pentru administratori:** Administratorii pot vizualiza statistici privind activitatea utilizatorilor (ex. numărul de mesaje trimise într-un grup) și sunt singurii care pot modifica rolurile membrilor dintr-un grup.
* **Profil cu avatar personalizabil:** Utilizatorii beneficiază de un profil care include un avatar generat automat pe baza inițialei numelui. Ulterior, aceștia au posibilitatea de a-și personaliza imaginea de profil, încărcând o fotografie proprie pentru o identificare mai facilă în cadrul aplicației.

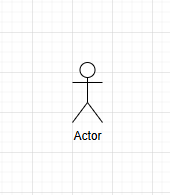
## Modelarea interacțiunii utilizatorilor prin diagrame

### Introducere în modelarea UML

Diagramele cazurilor de utilizare reprezintă un instrument esențial în analiza și proiectarea sistemelor software, fiind parte integrantă a limbajului de modelare UML (Unified Modeling Language). Acestea oferă o reprezentare grafică a interacțiunilor dintre actorii externi (utilizatori sau alte sisteme) și sistemul propriu-zis, evidențiind funcționalitățile pe care acesta le oferă. Prin intermediul acestui tip de diagramă, se poate obține o înțelegere clară a cerințelor funcționale, permițând o comunicare eficientă între analiști, dezvoltatori și utilizatori finali. Fiecare caz de utilizare descrie un scenariu specific de interacțiune, contribuind la definirea comportamentului așteptat al sistemului în diverse situații.

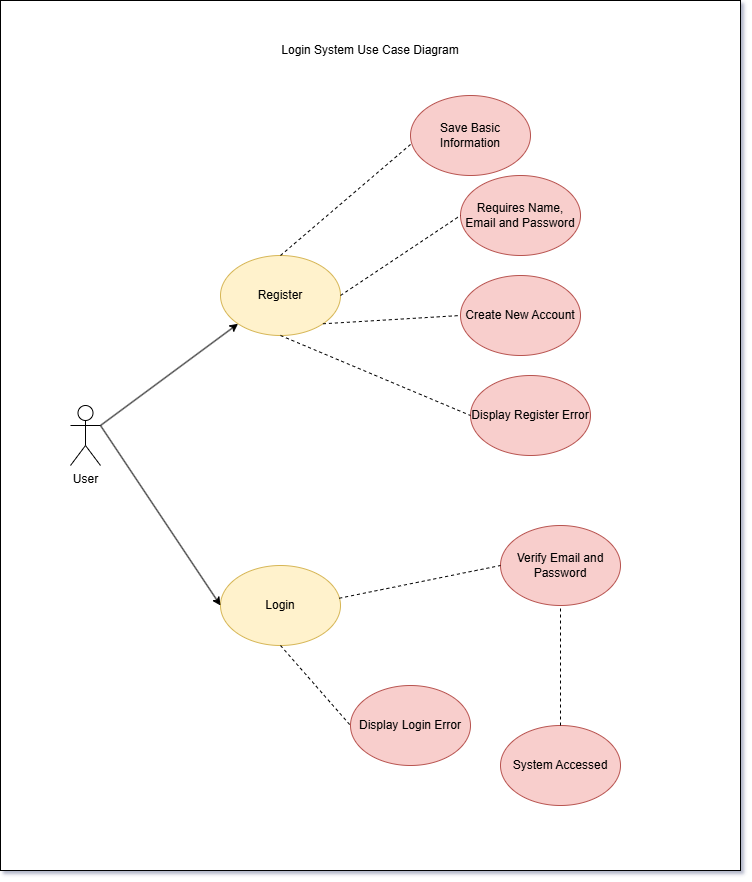
Potrivit lui Booch, Rumbaugh și Jacobson, autorii principali ai UML, „o diagramă de cazuri de utilizare este utilizată pentru a arăta setul de cazuri de utilizare și actorii sistemului, precum și relațiile dintre aceștia”. (Booch, 2005)

Actorul - reprezintă o entitate externă sistemului, care interacționează cu acesta. Poate fi un utilizator uman, un alt sistem software sau un dispozitiv. Actorii pot fi primari (inițiază interacțiunea) sau secundari (participă pasiv).



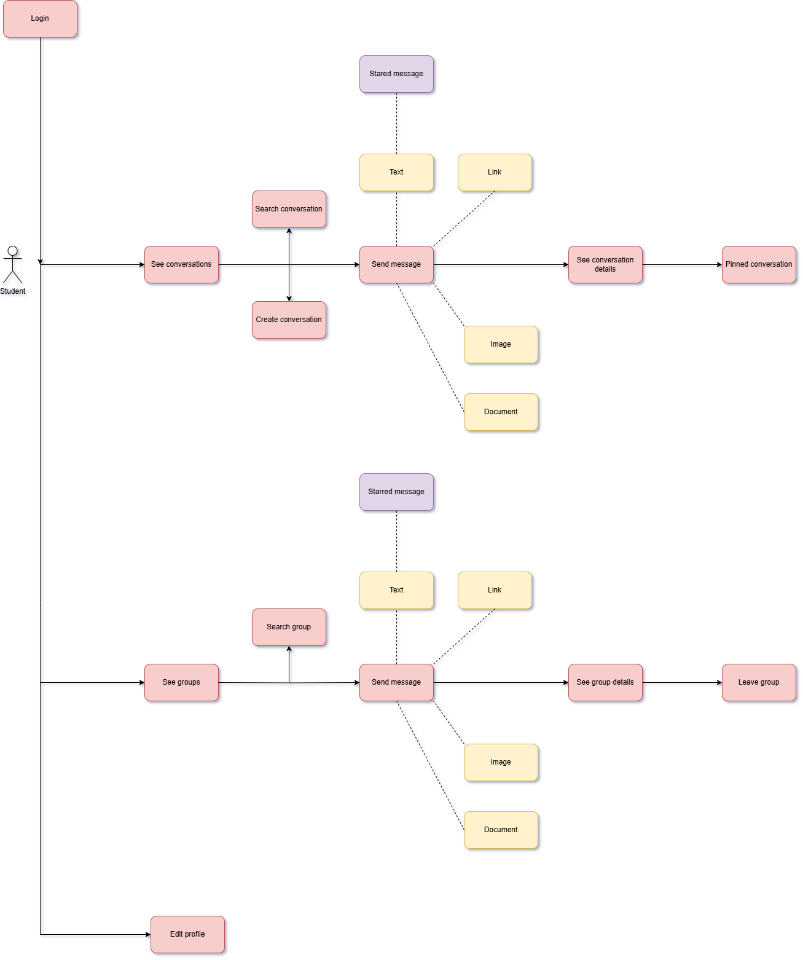
Cazul de utilizare (Use Case) reprezintă o funcționalitate specifică pe care sistemul o oferă unui actor. Este descris printr-un oval și denotă o acțiune valoroasă din perspectiva utilizatorului, precum „Autentificare” sau „Trimitere mesaj”.

### Caz de utilizare: Procesul de autentificare



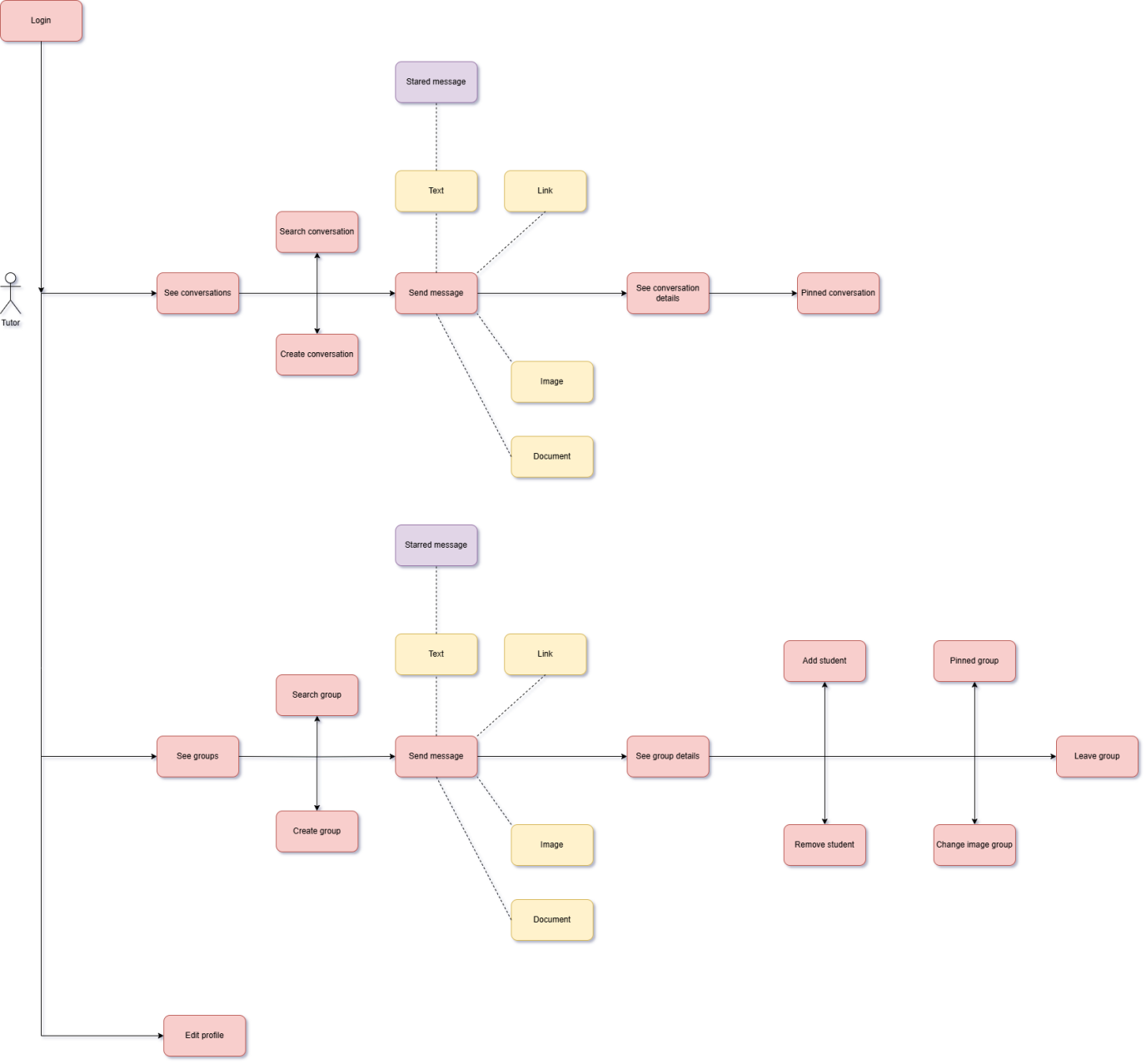
Figură 1 - Fluxul procesului de autentificare

### Caz de utilizare: Rolul studentului



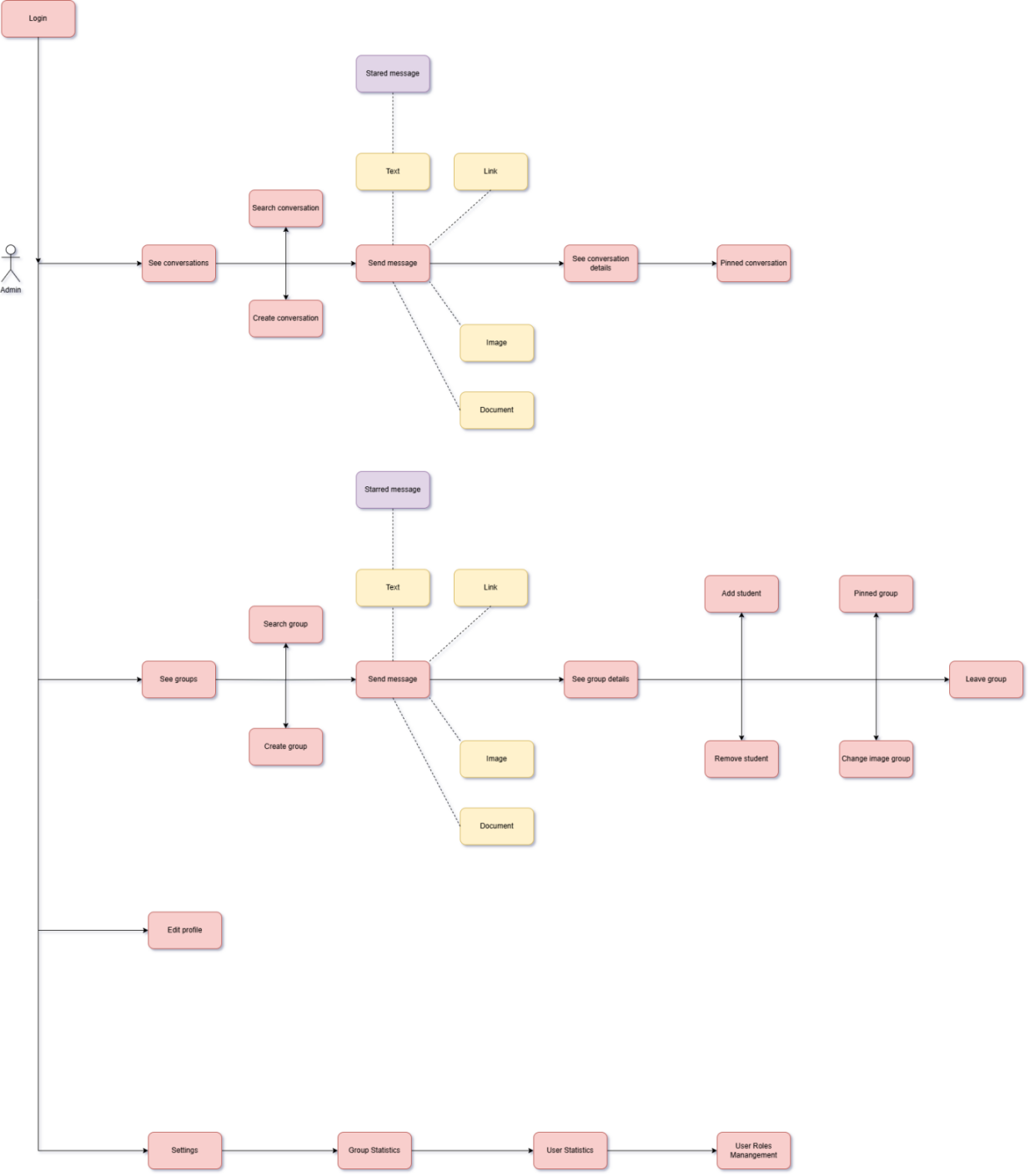
Figură 2 - Fluxul de utilizare pentru student

### Caz de utilizare: Rolul tutorelui



Figură 3 - Fluxul de utilizare pentru tutore

### Caz de utilizare: Funcții specifice administartorului



Figură 4 - Fluxul de utilizare pentru Admin

## Structura bazei de date

În contextul dezvoltării aplicației CampusConnect, organizarea bazei de date a reprezentat un aspect esențial pentru asigurarea unei funcționări stabile, eficiente și scalabile a sistemului.

### Modelul conceptual și relațional

Modelul relațional, introdus de Edgar F. Codd (1970), reprezintă fundamentul teoretic al bazelor de date moderne. Acesta organizează datele sub forma unor tabele (relații), fiecare compusă din rânduri (tupluri) și coloane (atribute). Acest model oferă o structură clară și riguroasă, ideală pentru gestionarea relațiilor dintre entități, precum cele existente între utilizatori și mesaje, sau între conversații și grupuri.

Baza de date a fost proiectată cu respectarea principiilor **normalizării** relaționale, un proces care presupune organizarea datelor astfel încât să se minimizeze redundanța și să se evite anomaliile de inserare, ștergere sau actualizare (Elmasri & Navathe, 2017). S-au aplicat formele normale până la cel puțin **a treia formă normală (3NF)**, ceea ce înseamnă că fiecare tabel conține doar atribute ce depind în mod direct de cheia primară și sunt independente tranzitiv.

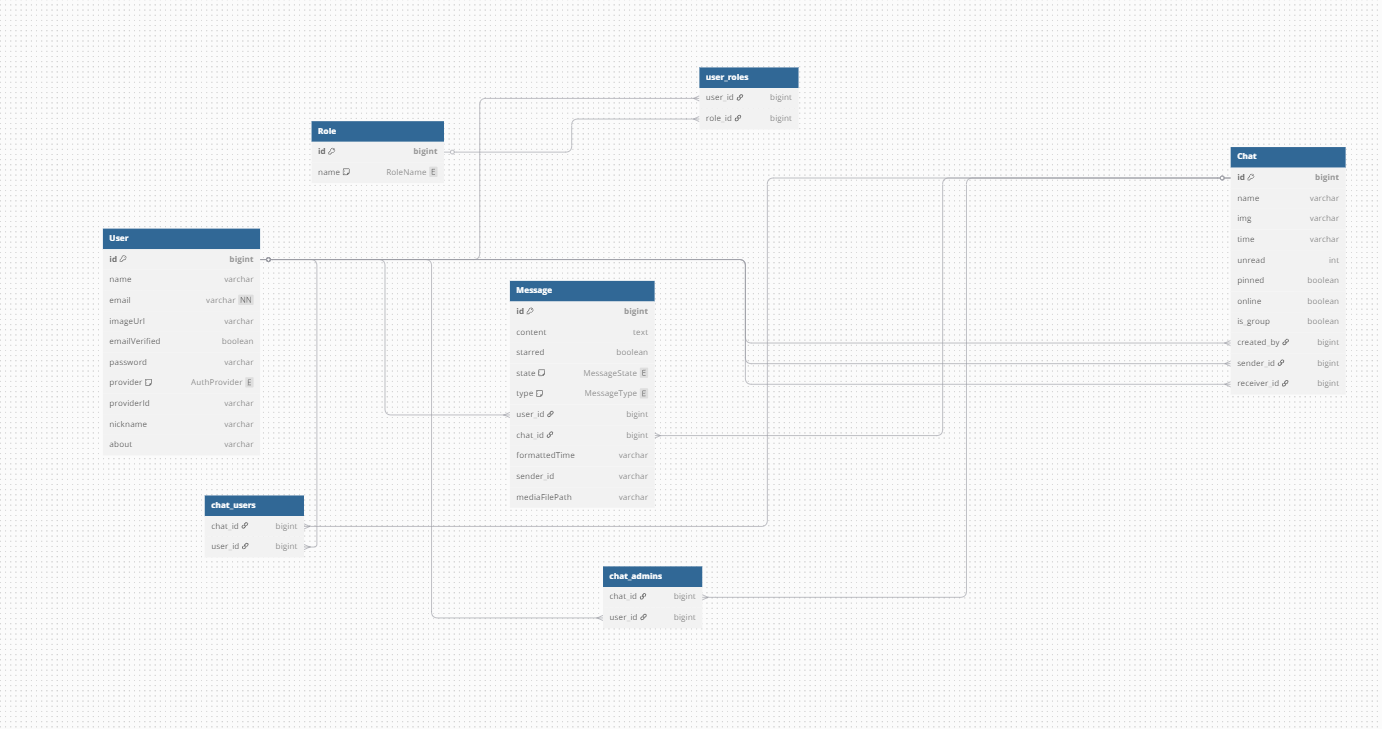
### Tehnologii utilizate pentru stocare

Baza de date a fost implementată folosind **PostgreSQL**, un sistem de gestiune a bazelor de date relaționale open-source, recunoscut pentru suportul avansat pentru tranzacții ACID (Atomicitate, Consistență, Izolare, Durabilitate), extensibilitate, suport JSON, indecși complecși și securitate avansată (PostgreSQL Global Development Group, 2024).

Pentru administrarea grafică a bazei de date, s-a utilizat **pgAdmin**, un instrument oficial dezvoltat pentru PostgreSQL. pgAdmin oferă facilități precum editarea vizuală a tabelelor și relațiilor, rularea interogărilor SQL, monitorizarea performanței și vizualizarea statisticilor, fiind extrem de util în faza de dezvoltare și testare a aplicației (pgAdmin Team, 2023).

### Maparea obiect-relatională (ORM)

În cadrul aplicației, persistarea datelor a fost gestionată prin utilizarea **JPA (Java Persistence API)** în combinație cu **Hibernate**, unul dintre cele mai utilizate frameworkuri ORM din ecosistemul Java. Prin această abordare, entitățile Java (ex: User, Chat, Message) sunt mapate direct în tabele din baza de date. Relațiile între entități (ex: @ManyToOne, @OneToMany, @ManyToMany) sunt implementate cu ajutorul adnotărilor JPA, ceea ce reduce complexitatea codului SQL și asigură consistența datelor (Bauer & King, 2020).



Figură 5 - Baza de date

### Considerații legate de performanță și consistență

Pentru a asigura integritatea datelor și performanța sistemului, s-au adoptat următoarele bune practici:

* Definirea de **chei primare** (@Id) și **chei externe** (@JoinColumn) pentru modelarea relațiilor între entități.
* Utilizarea **constrângerilor** (@Column(nullable = false), @UniqueConstraint) pentru a preveni introducerea datelor invalide.
* Crearea de **indici** pe coloane frecvent accesate (ex: email, chat\_id) pentru a accelera interogările.
* Selectarea strategiilor adecvate de **fetching** (EAGER sau LAZY), în funcție de frecvența accesării relațiilor.

Aceste decizii arhitecturale contribuie la un sistem robust, capabil să gestioneze eficient un volum mare de date și interacțiuni în timp real.

### Referințe bibliografice

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, 13(6), 377–387.

Elmasri, R., & Navathe, S. B. (2017). Fundamentals of Database Systems (7th ed.). Pearson.

Bauer, C., & King, G. (2020). Java Persistence with Hibernate (2nd ed.). Manning Publications.

PostgreSQL Global Development Group. (2024). PostgreSQL Documentation. [https://www.postgresql.org/docs/](https://www.postgresql.org/docs/" \t "_new)

pgAdmin Team. (2023). pgAdmin 4 Documentation. <https://www.pgadmin.org/docs/>

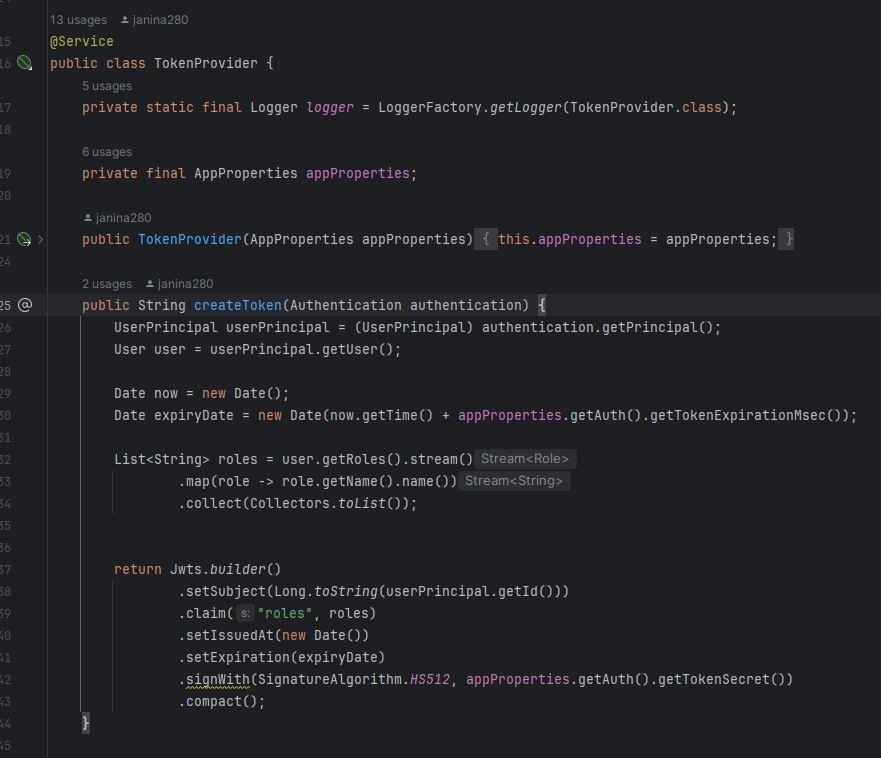
# IMPLEMENTAREA APLICAȚIEI

## Securitatea aplicației

Aplicația utilizează framework-ul Spring Security pentru protejarea resurselor expuse de backend. Autentificarea utilizatorilor se realizează pe baza unui token JWT (JSON Web Token), generat în urma unui proces de login și transmis ulterior în fiecare cerere HTTP, prin antetul Authorization.

Arhitectura componentelor de securitate:

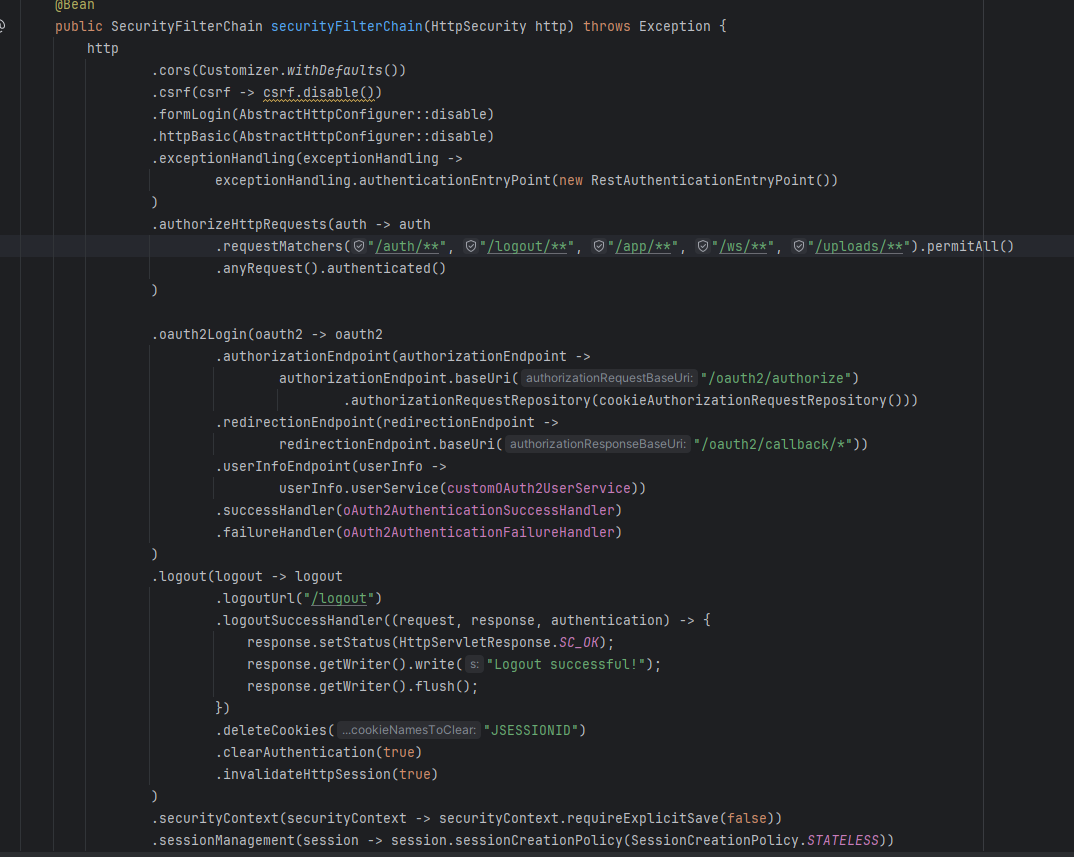
1. TokenProvider: Componentă responsabilă pentru generarea, extragerea și validarea token-urilor JWT. Token-ul include în payload identificatorul utilizatorului (sub) și o listă de roluri.



1. TokenAuthenticationFilter: Implementare a clasei OncePerRequestFilter care interceptează fiecare cerere HTTP, extrage token-ul JWT, îl validează și setează utilizatorul curent în contextul de securitate (SecurityContextHolder).



1. CustomUserDetailsService: Implementare a interfeței UserDetailsService care permite încărcarea unui utilizator fie pe baza adresei de email, fie pe baza ID-ului. Este utilizată pentru extragerea datelor persistente ale utilizatorului din baza de date.
2. UserPrincipal: Implementare custom a interfețelor UserDetails care servește ca reprezentare internă a utilizatorului autentificat. Include informații despre roluri și poate fi convertită în entitate User.
3. SecurityConfig: Clasa de configurare principală, care definește un SecurityFilterChain și setează reguli de acces pentru diferite endpoint-uri. De asemenea, adaugă TokenAuthenticationFilter în lanțul de filtre Spring Security.

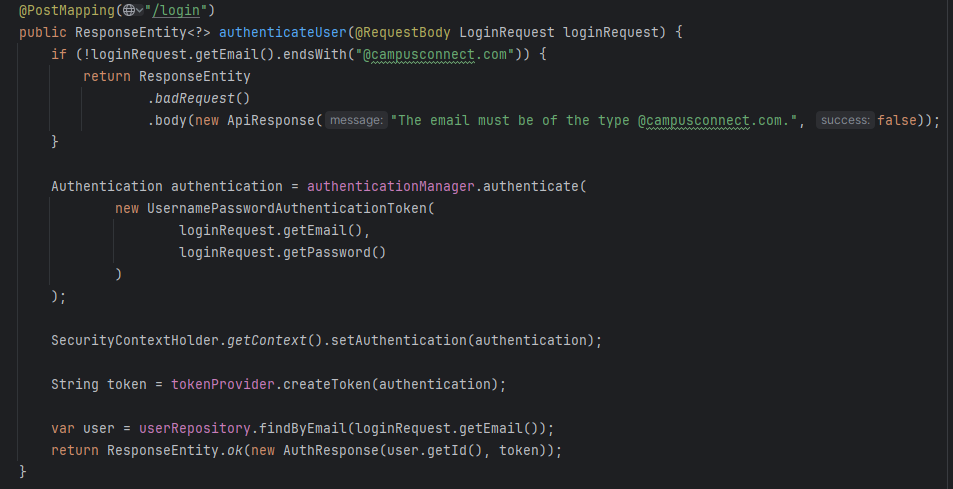


## Sistemul de autentificare

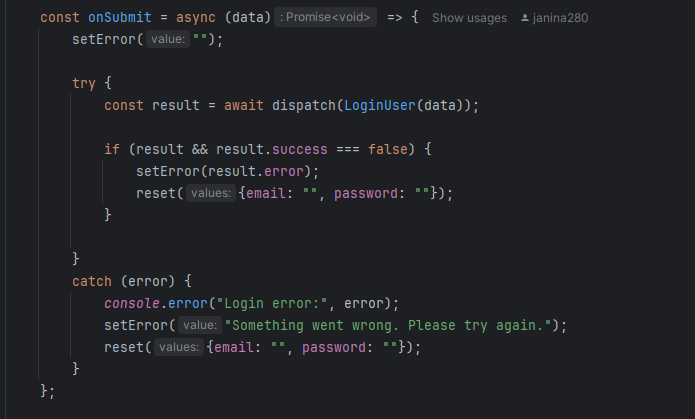
### Autentificarea (/auth/login)

Endpoint-ul /auth/login primește un obiect de tip LoginRequest, ce conține adresa de email și parola utilizatorului. Înainte de procesul de autentificare propriu-zis, se validează formatul emailului pentru a se asigura că acesta aparține domeniului @campusconnect.com.

În cazul în care datele sunt valide, se creează un obiect UsernamePasswordAuthenticationToken, care este transmis managerului de autentificare (AuthenticationManager). Dacă autentificarea este reușită, contextul de securitate este populat cu datele utilizatorului, iar TokenProvider generează un token JWT corespunzător.



Răspunsul conține atât tokenul JWT, cât și ID-ul utilizatorului, care vor fi ulterior folosite pe partea de client care este construită folosind React și gestionează autentificarea utilizatorului cu ajutorul Redux și a formularului de login validat prin schema Yup.



Când utilizatorul trimite formularul, datele sunt transmise prin intermediul metodei dispatch către thunk-ul asincron LoginUser definit în auth. Acesta trimite un request către endpoint-ul de autentificare al serverului și, în cazul unui răspuns de succes, salvează tokenul JWT în store-ul Redux și în localStorage.

În caz de eroare, componenta afişează un mesaj descriptiv, iar câmpurile formularului sunt resetate. Este disponibil și un toggle vizual pentru afişarea sau mascarea parolei, utilizând iconuri din biblioteca Phosphor.

# Concluzii

Autorul prezintă concluziile sale…

# Bibliografie

Abramov, D. (2019). *Introducing Hooks*. Preluat de pe React Blog: https://legacy.reactjs.org/docs/hooks-intro.html

Abramov, D. ș. (fără an). *A JS library for predictable and maintainable global state management*. Preluat de pe Redux: https://redux.js.org/

Alex Banks, E. P. (2016). *Learning React: Functional Web Development with React and Redux.* O’Reilly Media.

Banks, A. &. (2020). *Learning React: Modern Patterns for Developing React Apps.* O’Reilly Media.

Banks, R. &. (2023). *Modern Full-Stack Development with React, Redux, and React Router.* O’Reilly Media.

Bauer, C. (2016). *Hibernate Validator Reference Guide*. Preluat de pe https://hibernate.org/validator/

Booch, G. R. (2005). *The Unified Modeling Language User Guide (2nd ed.).* Addison-Wesley.

Contributors, A. (2024). *Axios – Promise based HTTP client for the browser and node.js.* Preluat de pe https://axios-http.com

DevTools, R. (2024). *Redux DevTools Extension Documentation*. Preluat de pe https://github.com/reduxjs/redux-devtools

Documentation, S. (fără an). *STOMP*. Preluat de pe GitHub: https://stomp-js.github.io/

Documentation, S. (fără an). *WebSocket Support*. Preluat de pe https://docs.spring.io/spring-framework/docs/current/reference/html/web.html#websocket

EE, J. (2023). *Jakarta Bean Validation Specification*. Preluat de pe https://jakarta.ee/specifications/bean-validation

Fitzgerald, A. (2021). *Declarative Animations with Framer Motion.* . Preluat de pe https://www.smashingmagazine.com/2021/02/declarative-animations-framer-motion/

Form, R. H. (2024). *Performant, flexible and extensible forms with easy-to-use validation*. Preluat de pe React Hook Form: https://react-hook-form.com/

Framer. (2024). *Framer Motion – Declarative animations for React.* Preluat de pe https://www.framer.com/motion/

Fulton, A. (2022). *Mastering React Router 6.* Packt Publishing.

Group, P. G. (2024). *PostgreSQL 16 Documentation.* Preluat de pe https://www.postgresql.org/docs/

Inc., F. (2023). *Quick Start*. Preluat de pe React: https://react.dev/learn

Inc., F. (2023). *The library for web and native user interfaces.* Preluat de pe React: https://react.dev/

Inc., G. (2023). *Material Design*. Preluat de pe https://m3.material.io/

Inc., P. (2025). *Postman – The Collaboration Platform for API Development*. Preluat de pe https://www.postman.com/

Inc., R. S. (2024). *React Router – Declarative routing* .

JetBrains. (2024). *WebStorm – The Smartest JavaScript IDE.* Preluat de pe https://www.jetbrains.com/webstorm/

JetBrains. (2025). *IntelliJ IDEA – The Java IDE for Professional Developers*. Preluat de pe https://www.jetbrains.com/idea/

Jones, M. B. (2015). *JSON Web Token (JWT) – RFC 7519*. Preluat de pe https://datatracker.ietf.org/doc/html/rfc7519

Jquense. (2023). *Yup – Dead simple object schema validation.* Preluat de pe Yup: https://github.com/jquense/yup

Kurniawan, B. (2022). *Spring Boot for Beginners: A Hands-On Guide to Building Web Applications Using Spring Boot and Spring MVC.*

Larsen, O. &. (2021). *Mastering Material-UI.* Packt Publishing.

Long, J. (2021). *Spring Boot Reference Documentation*. Preluat de pe https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

Mak, J. (2024). *WebSocket Support in Spring Framework*. Preluat de pe https://docs.spring.io/spring-framework/reference/web/websocket.html

Meier, R. &. (2022). *Spring Data – Modern Data Access for Enterprise Java.* Preluat de pe https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

Microlink. (2024). *Microlink – Turn websites into data*. Preluat de pe https://microlink.io

Mokito, S. &. (2021). *Efficient Java Development with Lombok and Spring Boot.* Apress.

Ng, W. (2017). *Pro React 16.* Apress.

Project, L. (2024). *Project Lombok – Java boilerplate code reduction tool*. Preluat de pe https://projectlombok.org/

Rahman, B. (2021). *Spring Security in Action.*

Recharts. (fără an). *Recharts - Documentation*. Preluat de pe https://recharts.org/

Schmidt, M. (2021). *Pro JPA 2 in Java EE 8: An In-Depth Guide to Java Persistence APIs. 3rd .* Apress.

Seacord, R. (2021). *Effective Java Validation.* Addison-Wesley.

SockJs. (2024). *SockJs*. Preluat de pe GitHub: https://github.com/sockjs/sockjs-client

Stonebraker, M. K. (2020). *The Design of PostgreSQL: Lessons from the Past and Future Directions*. Preluat de pe https://queue.acm.org/detail.cfm?id=3424300

Team, M. (2024). *Material UI – React components for faster and easier web development.* Preluat de pe https://mui.com/

Team, P. D. (2025). *pgAdmin – The most popular and feature rich Open Source administration and development platform for PostgreSQL*. Preluat de pe https://www.pgadmin.org/

Walls, C. (2022). *Spring Boot in Action. 2nd ed. .* NY: Manning Publications.

Zhang, X. (2022). *Building Web APIs with Spring Boot and Axios.* Packt Publishing.

# Referințe web

Recomandăm și aici respectarea regulilor enunțate pentru secțiunea 5.

[Alm08] – Pedro de Almeida, Patrik Fuhrer, Documentation Guidelines for Diploma and Master Thesis, Universitatea din Fribourg, Elveția, 2008, disponibil on-line la adresa http://diuf.unifr.ch/drupal/softeng/teaching/guidelines

[Olt07] – Th. Olteanu, C. Albu, *Ghid pentru redactarea lucrării de diplomă sau a disertaţiei de masterat*, Universitatea Română de Arte și Științe „Gheorghe Cristea”, 2007, disponibil via web la adresa http://www.ugc.ro/tpl/GHID REDACTARE DIPLOMA LICENTA.pdf

# Codul sursă

În această anexă se adaugă codul sursă al aplicației…

# Site-ul web al proiectului

Autorul prezintă în această anexă (opțională) site-ul web asociat proiectului său.

# CD / DVD

Autorul atașează în această anexă obligatorie, versiunea electronică a aplicației, a acestei lucrări, precum și prezentarea finală a tezei.



# Index

B

Bibliografie 9

C

CUPRINSUL xi

D

Dimensiuni 3

F

Figuri 4

Formulele matematice 4

I

Ilustrațiile 4

L

Legenda 6

LISTA FIGURILOR xii

LISTA TABELELOR xiii

R

Referințe web 10

S

Structura documentului 2

T

Tabele 5

1. API (Application Programming Interface) reprezintă un set de reguli și protocoale care permit aplicațiilor software să comunice între ele. În contextul aplicațiilor web, un API expune funcționalități ale serverului ce pot fi accesate de către client prin cereri HTTP. (GeeksforGeeks, 2024) [↑](#footnote-ref-0)
2. Nested routes sunt rute imbricate într-o aplicație web, în care o pagină principală conține subpagini, fiecare reprezentată de o rută copil. [↑](#footnote-ref-1)
3. Data loaders sunt funcții asociate rutelor React Router care încarcă datele necesare unei componente înainte ca aceasta să fie randată, oferind o experiență de utilizare mai fluentă și coerentă. [↑](#footnote-ref-2)