

# Sistem de evidenta a elevilor

Buşecan Andrei, Sălnicean Răzvan, Titea Ruben, Terpe Victor

Ianuarie 2025

## Contents

<b>1</b>	<b>Introducere</b>	<b>3</b>
<b>2</b>	<b>Cerințe</b>	<b>3</b>
<b>1</b>	<b>Introducere SRS</b>	<b>6</b>
1.1	Scop . . . . .	6
1.2	Public țintă . . . . .	6
1.3	Domeniul de utilizare . . . . .	6
1.4	Domeniul de aplicare al produsului . . . . .	6
1.5	Definiții și Acronime . . . . .	7
<b>2</b>	<b>Descriere Generală</b>	<b>7</b>
2.1	Nevoile Utilizatorilor . . . . .	7
<b>3</b>	<b>Funcționalitățile și Cerințele Sistemului</b>	<b>7</b>
3.1	Cerințe Funcționale . . . . .	7
3.1.1	Gestionarea Informațiilor Elevilor . . . . .	7
3.1.2	Gestionarea Notelor . . . . .	7
3.1.3	Gestionarea Absențelor . . . . .	7
3.1.4	Alocarea Burselor . . . . .	8
3.1.5	Ritmicitatea notării . . . . .	8
3.1.6	Monitorizarea elevilor repetenți . . . . .	8
3.2	Cerințe de Interfață . . . . .	8
3.3	Funcționalitățile Sistemului . . . . .	8
3.3.1	Control de Acces Bazat pe Roluri . . . . .	8
3.3.2	Notificări . . . . .	9
3.3.3	Căutare . . . . .	9
3.4	Cerințe Ne-Funcționale . . . . .	9
3.4.1	Performanță . . . . .	9
3.4.2	Securitate . . . . .	9
3.4.3	Utilitate . . . . .	9
3.4.4	Scalabilitate . . . . .	9

<b>4</b>	<b>Cerințele MoSCoW</b>	<b>10</b>
<b>3</b>	<b>Aspecte teoretice și State of the Art</b>	<b>11</b>
<b>4</b>	<b>Implementare</b>	<b>14</b>
4.1	Diagramele de utilizare . . . . .	14
4.2	Tehnologiile folosite . . . . .	19
4.2.1	Frontend . . . . .	19
4.2.2	Backend . . . . .	19
4.2.3	Baza de date . . . . .	19
4.3	Arhitectura . . . . .	20
4.4	Diagrama claselor . . . . .	23
4.5	Implementarea RestAPI . . . . .	25
4.6	Implementarea Interfeței . . . . .	26
4.6.1	Dashboard Profesori . . . . .	26
4.6.2	Dashboard Secretare . . . . .	27
4.7	Alte Detalii de Implementare . . . . .	27
<b>5</b>	<b>Testare și validare</b>	<b>29</b>
5.1	Unit Test . . . . .	29
5.2	Test de Integrare . . . . .	30
<b>6</b>	<b>Rezultate</b>	<b>30</b>
6.1	Funcționalități implementate . . . . .	30
6.2	Capturi de Ecran pentru Interfețe . . . . .	31
6.2.1	Dashboard pentru secretare . . . . .	31
6.2.2	Dashboard pentru profesori . . . . .	33
6.3	Funcționalități neimplementate . . . . .	34
6.4	Posibile Îmbunătățiri . . . . .	34
<b>7</b>	<b>Concluzii</b>	<b>35</b>
<b>8</b>	<b>Funcționarea ca echipă</b>	<b>36</b>
8.1	Rolurile membrilor . . . . .	36
8.2	Îndeplinirea rolurilor . . . . .	36
8.3	Probleme întâlnite . . . . .	37

## 1 Introducere

Noi am ales acest proiect deoarece vrem să contribuim la tehnologizarea educației.

În ultimii ani, școlile și liceele au început să pună accentul tot mai mult pe digitalizarea procesului educațional, ceea ce a dus inevitabil la nevoia de a dezvolta sisteme informatice care să satisfacă aceste necesități.

Un sistem de evidență al elevilor este o soluție foarte bună pentru a gestiona într-un mod cât mai lipsit de efort și eficient activitatea elevilor din cadrul instituțiilor de învățământ.

Prin sistemul propus de noi, dorim să venim în ajutorul personalului din cadrul școlilor pentru a le ușura munca în ceea ce privește înregistrarea elevilor, vizualizarea și adăugarea notelor, absențelor și burselor. De asemenea, situația școlară va putea fi monitorizată și prin intermediul diferitelor rapoarte care vor putea fi generate în cadrul sistemului.

## 2 Cerințe

Pentru a putea realiza acest sistem de evidență a elevilor am adunat cerințele pentru posibilele funcționalități pe care sistemul trebuie să le îndeplinească. În urma întâlnirii pe care am avut-o cu oficialitățile de la Colegiul Tehnic "George Barițiu" am primit o listă cu cerințele dorite de ei pentru un astfel de sistem, care se regăsește mai jos (din păcate nu am primit ștampilă și semnătură).

### Cerințe – Sistem de evidență a elevilor dintr-un liceu

#### Sistemul trebuie să includă:

- **Informații despre fiecare elev:**
  - Nume
  - Adresa
  - CNP
  - Număr de telefon
  - Clasa
  - Data nașterii
  - Media
- **Notele elevilor**
- **Absențele elevilor**

- **Raport cu absențele pe clasă și școală:**
  - Lunar și anual
  - Calculare număr de baieti și fete
- **Alocarea burselor:**
  - De performanță
  - De merit
  - De reziliență
  - Socială
  - Tehnologică
- **Ritmicitatea notării:**
  - Fiecare profesor trebuie să acorde cel puțin o notă în fiecare modul
  - Înainte cu 2 săptămâni de încheierea modului se generează un raport cu profesorii care au pus/n-au pus notă.
- **Corigențele:**
  - Calculare medie înainte de încheierea modulului și generarea unui raport cu elevii corigenți din fiecare clasă

## Cerinte - Sistem de Evidență a elevilor dintr-un liceu.

Sistemul trebuie să includă:

- Informații despre fiecare elev (nume, adresă, CNP, număr de telefon, clasă, data nașterii, medii)
- Notele elevilor
- Absențele elevilor
- Raport cu absențele pe clasă / scoală → lunar și anual.  
↳ trebuie să includă și nr. de băieți/fete.
- Alocarea buranelor: - de performanță  
- de merit  
- de reziliență  
- socială  
- tehnologică
- Ritmicitatea notării  
↳ fiecare profesor trebuie să oculte cel puțin o notă pe modul  
→ cu 2 săptămâni înainte de închiderea modului  
să genereze un raport cu profesorii care au pus / nu au pus note
- Corigențele  
↳ calcularea mediei dinainte și anunțarea elevilor corigenți din fiecare clasă.

Figure 1: Lista cu cerinte.

Pe baza acestor cerințe am realizat un Document de Specificare a Cerințelor Software care se regăsește mai jos.

## **Document SRS**

### **1 Introducere SRS**

#### **1.1 Scop**

Scopul acestui document SRS este de a descrie modul în care va fi implementat și va funcționa Sistemul de Evidență al Elevilor. Acest sistem va fi utilizat pentru a gestiona informațiile elevilor, absențele, notele, alocarea bursei și diferite rapoarte despre activitatea școlară.

#### **1.2 Public țintă**

Documentul este destinat pentru:

- Echipa de programatori care implementează sistemul(adică noi).
- Administratorii liceului(secretarele și directorii).
- Profesorii, care vor folosi sistemul cel mai frecvent.

#### **1.3 Domeniul de utilizare**

Sistemul va face gestionarea informațiilor elevilor din liceu mult mai ușoară, aducând mai multă eficiență în procesul de înregistrare, vizualizare și analizare a datelor legate de performanțele școlare și prezență. Profesorii, elevii, părinții și secretarele vor fi cei care se vor bucura de beneficiile aduse de sistem.

#### **1.4 Domeniul de aplicare al produsului**

Sistemul de Evidență a Elevilor dintr-un Liceu va:

- Gestiona informațiile personale și școlare ale elevilor.
- Monitoriza absențele și performanțele școlare ale elevilor.
- Genera diferite rapoarte.
- Permite alocarea diferitelor tipuri de burse pe baza performanței și a altor criterii.
- Facilitează ritmicitatea notării și va trimite notificări atunci când profesorii nu respectă termenul limită pentru adăugarea notelor.

## 1.5 Definiții și Acronime

- SRS – Specificații de Cerințe Software

## 2 Descriere Generală

### 2.1 Nevoile Utilizatorilor

- Administratorii au nevoie să gestioneze toate datele elevilor, absențele, alocarea bursei și să vizualizeze rapoarte despre performanțele elevilor.
- Profesorii au nevoie să treacă notele, absențele și să vizualizeze rapoarte despre performanțele elevilor.
- Părinții au nevoie să vizualizeze informații despre notele, absențele și bursele copiilor lor.
- Elevii au nevoie să vizualizeze notele, absențele și bursele proprii.

## 3 Funcționalitățile și Cerințele Sistemului

### 3.1 Cerințe Funcționale

#### 3.1.1 Gestionarea Informațiilor Elevilor

Datele elevilor care vor fi introduse sunt:

- Nume, Prenume, CNP, Sex, Număr de telefon, Adresa, Data nașterii, Clasa din care face parte

Sistemul trebuie să permită adăugarea, editarea și ștergerea elevilor înregistrați.

#### 3.1.2 Gestionarea Notelor

- Profesorii trebuie să poată introduce notele pentru fiecare elev la materia pe care o predă.
- Sistemul calculează automat media pentru fiecare materie.
- Generarea unui raport cu notele elevilor pe materii și clase pentru profesori și administratori.

#### 3.1.3 Gestionarea Absențelor

- Profesorii trebuie să poată pune absența elevilor zilnic.
- Sistemul trebuie să genereze rapoarte de absență pe clasă și școală, lunar și anual.
- Rapoartele trebuie să includă numărul de băieți și fete.

### 3.1.4 Alocarea Burselor

Sistemul trebuie să permită administratorilor să aloce următoarele tipuri de burse:

- Burse de performanță
- Burse de merit
- Burse de reziliență
- Burse sociale
- Burse tehnologice

### 3.1.5 Ritmicitatea notării

- Sistemul trebuie să solicite profesorilor să introducă cel puțin o notă pentru fiecare elev pe modul.
- Cu două săptămâni înainte de sfârșitul modulului, se va genera un raport care arată profesorii care au și care nu au introdus note.

### 3.1.6 Monitorizarea elevilor repetenți

- Sistemul trebuie să calculeze mediile generale înainte cu o lună de sfârșitul anului și să genereze un raport pe clase cu elevii care riscă să rămână repetenți.

## 3.2 Cerințe de Interfață

Interfața cu utilizatorul trebuie să fie bazată pe web, accesibilă din orice browser, cu design receptiv pentru a putea fi utilizată pe orice fel de dispozitiv. Sistemul trebuie să fie compatibil cu hardware-ul standard pentru desktop și dispozitive mobile. Pentru a stoca datele va fi folosit un sistem de gestiune a bazelor de date(SGBD).

## 3.3 Funcționalitățile Sistemului

### 3.3.1 Control de Acces Bazat pe Roluri

Utilizatorii vor avea diferite tipuri de access la sistem în funcție de rolul lor:

- Secretarele sau administratorii vor avea access la toate funcționalitățile sistemului.
- Profesorii vor putea vedea informațiile elevilor doar la clasele în care predau și numai la materiile pe care le predau. Dacă profesorul e și diriginte atunci va putea accesa situația elevilor la toate materiile.



- Părinții și elevii vor putea doar vizualiza notele, absențele și bursele personale.

Conturile utilizatorilor vor fi create de către secretară, care va comunica numele de utilizator și parola ulterior.

### **3.3.2 Notificări**

Sistemul trebuie să implementeze un mecanism de notificare pentru a alerta profesorii să pună note în fiecare modul.

### **3.3.3 Căutare**

Sistemul va avea bări de căutare pentru ca profesorii și secretarele să găsească mai repede elevii pentru care doresc să gestioneze situația școlară. De asemenea la gestionarea notelor și absențelor vor exista opțiuni de căutare după modul și materie.

## **3.4 Cerințe Ne-Funcționale**

### **3.4.1 Performanță**

Sistemul trebuie să permită conectarea a 100 de utilizatori în același timp, fără să existe întârzieri. Timpul de răspuns pentru a afișa datele trebuie să fie sub 3 secunde.

### **3.4.2 Securitate**

Sistemul trebuie să implementeze autentificarea utilizatorilor în mod securizat. Datele sensibile, cum sunt parolele trebuie criptate. Doar utilizatorii care au cont vor avea acces la sistem, iar cei cu cont vor fi restricționați de la anumite funcționalități în funcție de rol.

### **3.4.3 Utilitate**

Interfața va fi una intuitivă și ușor de folosit, astfel încât și profesorii mai în vârstă și fără experiență în utilizarea tehnologiei să poată utiliza sistemul.

### **3.4.4 Scalabilitate**

Sistemul trebuie să poată fi îmbunătățit cu ușurință pentru a extinde alte funcționalități, cum ar fi adăugarea unui mod de comunicare pentru ca profesorii să poată comunica cu părinții referitor la situația elevilor.

După ce am întocmit documentul SRS, am aplicat metoda de clasificare MoSCoW pentru a organiza cerințele în funcție de prioritatea pe care o vor avea în faza de implementare.

## 4 Cerințele MoSCoW

### Must have:

- Gestionarea informațiilor despre elevi.
- Introducerea, modificarea și ștergerea informațiilor elevilor.
- Gestionarea notelor și absențelor.
- Introducerea notelor și absențelor pentru fiecare elev de către profesori și calcularea mediei pentru fiecare materie.

### Should have:

- Alocarea burselor (merit, performanță, reziliență, sociale, tehnologice).
- Atenționarea profesorilor pentru a pune note înainte de sfârșitul modului.
- Rapoarte:
  - Rapoarte de absențe pe clasă și școală (lunar și anual).
  - Rapoarte de note pe materii și clase.
  - Raport cu elevii care riscă să rămână repetenți.
  - Raport cu profesorii care au pus sau n-au pus note în modulul curent.

### Could have:

- Vizualizare notelor și absențelor de către elevi și părinți.
- Algoritm pentru distribuirea burselor.
- Vizualizarea situației școlare din anii anteriori (note, medii, burse).
- Raport general la finalul anului pentru fiecare elev cu toate notele, absențele și mediile obținute.

### Won't have:

- Elevii și părinții nu vor putea vizualiza notele, absențele și bursele altor elevi.
- Profesorii nu vor putea vizualiza notele și absențele elevilor la alte materii (excepție dacă profesorul e diriginte pentru clasa respectivă), nu vor putea modifica/șterge notele și absențele.

### 3 Aspecte teoretice și State of the Art

În această secțiune vom vorbi despre conceptele și cunoștințele care sunt necesare pentru a dezvolta sistemul nostru, precum și starea în care se află domeniul educațional.

Integrarea tehnologiei în instituțiile de învățământ a adus de-a lungul timpului multe beneficii în procesul de educare al elevilor. Mai demult în școli se foloseau table cu cretă care apoi au fost înlocuite de alte table mai îmbunătățite, care folosesc markere, iar acum în foarte multe școli sunt introduse tablele inteligente. Profesorii folosesc rețelele de socializare și mesagerie, cum ar fi Facebook și Whatsapp pentru a organiza și comunica elevilor diferite noutăți [1]. Prin folosirea instrumentelor digitale în cadrul claselor, elevii pot să fie mult mai implicați în ceea ce se predă, profesorii pot să își îmbunătățească planurile de lecții și există chiar posibilitatea de planuri de învățare personalizate. Totodată, îi ajută pe elevi să dezvolte abilități tehnice necesare secolului 21 la viitoarele locuri de muncă [2].

Deși beneficiile sunt multe, există și provocări în introducerea tehnologiei în școli. În primul rând, tehnologiile moderne cresc nevoile de instruire a profesorilor. Atitudinea profesorilor este cel mai important factor pentru ca tehnologizarea să fie una de success. Dacă profesorii nu sunt dispuși să învețe cum funcționează și cum să folosească calculatoarele sau alte tehnologii, beneficiile aduse de acestea nu vor fi văzute [3]. În al doilea rând, școlile sunt nevoite să asigure dispozitivele și sistemele tehnologice necesare, precum și să țină pasul cu dezvoltările tehnologice prin înlocuirea tehnologiilor vechi.

Unul dintre cele mai importante aspecte în care sistemele tehnologice sunt folosite este organizarea și gestionarea elevilor și activităților din cadrul școlilor. Pentru aceasta au fost dezvoltate sistemele de evidență a elevilor, care sunt utilizate pentru a introduce, stoca și gestiona datele referitoare la elevi, cum ar fi datele personale, notele, absențele etc. [4].

Câteva dintre beneficiile utilizării unui sistem de evidență a elevilor:

**Eficiența în administrare este îmbunătățită:** Sarcinile repetitive (de rutină) sunt automatizate, în acest fel probabilitatea ca erorile să apară este redusă semnificativ, iar munca manuală este de asemenea diminuată. Un studiu arată că utilizarea unor sisteme de evidență a elevilor standardizate duce la o organizare mult mai eficientă [5].

**Monitorizarea performanței studenților:** Cu ajutorul sistemelor de evidență a elevilor, performanțele elevilor pot fi urmărite cu ușurință, profesorii având posibilitatea de a identifica elevii care nu se descurcă și au nevoie de mai multă atenție. Cercetările arată că aceste sisteme pot identifica materiile la care elevii nu se descurcă, contribuind la dezvoltarea unor strategii de educație corespunzătoare [6].

**Gestionarea și accesarea datelor într-un singur loc:** Prin stocarea centralizată a informațiilor despre elevi, profesorii și secretarele nu mai trebuie să se bazeze pe metodele tradiționale care implică folosirea unui număr enorm de hârtii și cataloage. [7]

Una dintre arhitecturile care este foarte des întâlnită în implementarea sistemelor de evidență este arhitectura client-server.

**Arhitectura client-server** este un model în care sarcinile sunt împărțite între furnizorii de resurse sau servicii, care sunt numiți servere, și cei care solicită aceste resurse și servicii, numiți clienți. Așadar serverele oferă resursele, iar clienții le accesează printr-o rețea. Clientul este o aplicație sau un dispozitiv care face cereri către server. Acesta pune la dispoziție o interfață utilizatorilor pentru ca ei să poată solicita, iar apoi afișa rezultatele primite. Serverul este un sistem care așteaptă cereri de la clienți și le răspunde în funcție de ce au solicitat. El poate să răspundă la mai mulți clienți în același timp . [8] Comunicarea dintre client și server se realizează printr-o rețea de calculatoare care folosesc protocoale standard cum ar fi: File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP) și Hypertext Transfer Protocol (HTTP) [9].

**Hypertext Transfer Protocol (HTTP)** este un protocol fundamental de transmitere a informațiilor pe internet. Funcționează pe un model cerere-răspuns, ceea ce îl face să fie foarte compatibil cu arhitectura client-server. Poate fi utilizat pentru a transmite o gamă variată de tipuri de date, cum ar fi text, imagini, videoclipuri etc. HTTP are mai multe metode pentru cereri:

- Metoda GET este folosită pentru a solicita resursele de pe un server.
- Metoda POST este folosită pentru a adăuga o resursă nouă.
- Metoda PUT este folosită pentru a actualiza o resursă care există deja.
- Metoda DELETE este folosită pentru a șterge o resursă din server.

După ce cererea este procesată de server, acesta trimite un cod de stare în funcție de rezultat. Codurile care încep cu **2\*\*** indică faptul că cererea a fost procesată cu succes (exemplu 200 OK, 201 Created), cele care încep cu **4\*\*** semnaleză o eroare în cererea făcută de client(exemplu: celebrul 404 Not Found, 400 Bad Request), iar cele care încep cu **5\*\*** desemnează o eroare pe partea de server(exmplu 500 Internal Server Error, 502 Bad Gateway). [10]

**REST API**(Representational State Transfer Application Programming Interface) este o interfață de programare care permite ca mai multe aplicații diferite să comunice între ele, utilizând protocolul HTTP. Este bazat pe arhitectura REST care are șase constrângeri pe care un REST API(numit RESTful) trebuie să le îndeplinească:

1. Arhitectura client-server
2. Fără stare(Stateless): serverul nu păstrează informații despre starea clientului între cereri.
3. Cache: Datele care sunt utilizate frecvent trebuie stocate pentru a îmbunătăți performanța.

4. Interfață Uniformă(Uniform Interface)

- Identificarea resurselor: Fiecare resursă trebuie să fie accesibilă printr-un identificator unic(URI).
- Manipularea resurselor prin reprezentare: Resursele sunt returnate într-un format standard (JSON sau XML).
- Mesajele autodescriptive returnate clientului trebuie să conțină informații suficiente pentru a descrie cum trebuie procesată.
- HATEOAS (Hypermedia As The Engine Of Application State): Răspunsul serverului trebuie să includă informații despre ce poate face clientul în alte cereri.

5. Sistem Stratificat (Layered System): Între un client și un server pot să existe alte straturi intermediare(proxy, load-balancere etc.) care sunt invizibile clientului.

6. Cod la Cerere (optional): Serverul poate trimite cod executabil (de exemplu, JavaScript) către client. [11]

## 4 Implementare

În această secțiune vom expune modul în care am implementat sistemul de evidență al elevilor și tehnologiile pe care le-am utilizat.

### 4.1 Diagramele de utilizare

Am început prin identificare cazurilor de utilizare(Use Case) pentru utilizatori, adică secretarele și profesorii. După cum am specificat și în analiza MosCoW aceștia doi vor fi utilizatorii inițiali ai sistemului, urmând ca în viitor să fie adăugați elevii și părinții.

În sistemul nostru secretara va fi și admin, așadar va avea access la toate funcționalitățile sistemului. Cazurile identificate sunt următoarele:

- **Gestionare conturi:** Secretara poate vizualiza, adăuga, modifica sau șterge conturile utilizatorilor, poate și să vizualizeze sau modifice profilul propriu.
- **Gestionare elevi:** Administrarea datelor personale ale elevilor.
- **Gestionare profesori:** Administrarea datelor personale ale profesorilor și asocierile cu clasele și materiile pe care le predau.
- **Gestionare burse:** Adăugarea și modificarea criteriilor de acordare a burselor, alocarea sau ștergerea burselor pentru elevi.
- **Gestionare note și absențe:** Vizualizarea, adăugarea, modificarea și ștergerea notelor și absențelor primite de elevi.
- **Generare rapoarte:** Secretara va putea genera următoarele tipuri de rapoarte: rapoart de note, rapoart de absențe, raport cu elevii corigenți și raport profesori care au pus/n-au pus note într-un modul.

Cazurile de utilizare pentru profesor:

- **Gestionare cont:** Profesorul poate să vadă contul cu informațiile personale și să modifice username-ul sau parola.
- **Gestionare note și absențe:** Vizualizarea și adăugarea notelor pentru clasele și materiile la care predă(dacă profesorul e și dirigintele clasei atunci va putea vedea informațiile la toate materiile din clasă).
- **Generare Rapoarte:** Profesorul poate genera următoarele rapoarte: raport de note, raport de absențe, raport elevi corigenți. Se aplică aceleași constrângeri ca la gestionarea notelor, dacă e diriginte poate genera raport pentru orice materie din clasă.

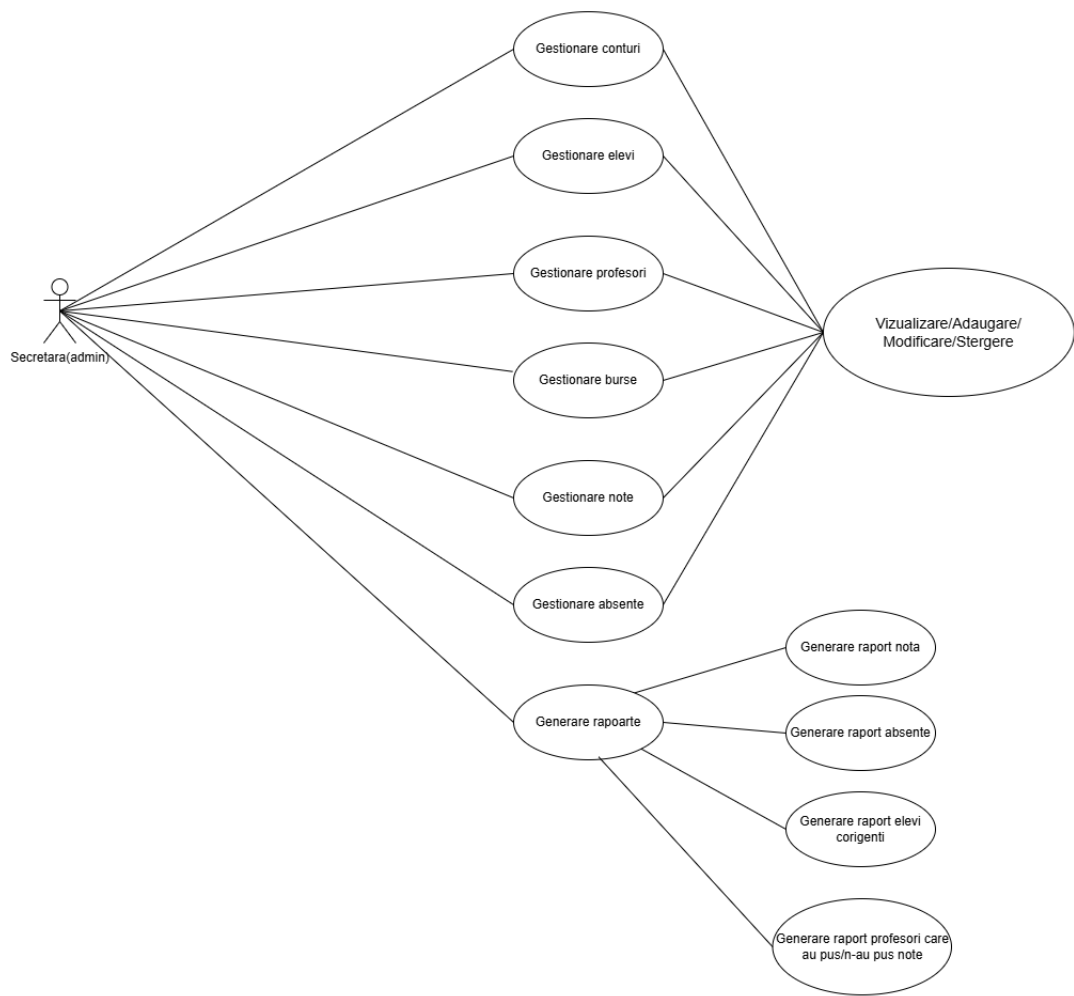


Figure 2: Diagrama Use Case pentru Secretara.



Figure 3: Diagrama Use Case pentru Profesor.

Am întocmit câte o persona pentru profesor și secretară, pentru a putea înțelege mai bine nevoile utilizatorilor.

## PERSONA - Profesor

- **Nume:** Blaga Maria
- **Vârsta:** 42 de ani
- **Stare civilă:** Căsătorită, cu doi copii
- **Funcție:** Profesoară de Matematică
- **Experiență profesională:** 18 ani
- **Educație:** Masterat în Matematică la Facultatea de Științe, UTCN

Maria este o profesoară de matematică cu aproape 20 de ani de experiență în învățământul liceal. Fiind pasionată de meseria sa, ea încearcă să le arate elevilor săi frumusețea matematicii printr-un stil de predare interesant , folosind



numeroase exemple din viața de zi cu zi. Ea este cunoscută de elevi pentru buna dispoziție și dorința sinceră de a-i ajuta să își atingă potențialul, lucru care se poate vedea în efortul suplimentar depus pentru a-i sprijini pe cei care au nevoie de mai multă atenție și explicații. Maria este într-o continuă căutare a echilibrului între viața de profesor și cea familială, fiind căsătorită și având doi copii cu vârste de 8, respectiv 12 ani.

### Obiective profesionale

- Maria este dedicată îmbunătățirii performanțelor elevilor săi, urmărind cu atenție progresul celor care întâmpină dificultăți.
- Își dorește o modalitate simplă și rapidă pentru a trece notele și absențele.

### Provocări

- Între predare, evaluare și pregătirea lecțiilor, Maria se simte adesea presată de timp.
- Deși utilizează confortabil aplicații de bază (browsere, Office etc), ar avea dificultăți în folosirea unor funcții mai avansate ale unui software complex.

### Nevoi de la Sistemul de Evidență al Elevilor

- **Interfață Simplă și Ușor de Utilizat:** Sistemul trebuie să-i permită Mariei să adauge și să gestioneze rapid notele și absențele, fără a fi necesară prea multă instruire.
- **Alerte și Notificări:** Ar fi util un sistem care să îi amintească termenele pentru a trece notele.
- **Generare de Rapoarte:** Pentru Maria, un sistem care să genereze rapoarte cu notele și performanțele elevilor ar fi de un foarte mare ajutor în identificarea elevilor care au dificultăți.

### PERSONA - Secretară

- **Nume:** Andreea Dumitrescu
- **Vârstă:** 26 de ani
- **Stare civilă:** Necăsătorită
- **Funcție:** Secretară a Liceului
- **Experiență profesională:** 2 ani
- **Educație:** Facultatea de Științe Economice și Gestiunea Afacerilor (FSEGA) Babeș-Bolyai

Andreea Dumitrescu este o secretară tânără, entuziastă și dedicată, aflată la începutul carierei sale în domeniul de secretariat. Fiind nou angajată la un liceu, este responsabilă de gestionarea informațiilor elevilor, totodată oferind ajutor profesorilor și conducerii școlii în ceea ce privește administrația. Andreea învață rapid și este motivată să se dezvolte din punct de vedere profesional. Este o persoană organizată și comunicativă, fiind apreciată de colegi pentru energia și dorința de a învăța lucruri noi. Andreea dorește să fie cât mai eficientă în gestionarea datelor elevilor și să se familiarizeze cât mai repede cu sistemul de evidență.

### **Obiective profesionale**

- Andreea își dorește să deprindă rapid cunoștințele necesare pentru a utiliza sistemul de evidență și pentru a gestiona eficient datele despre elevi, notele și prezența acestora.
- Este motivată să devină mai eficientă în sarcinile pe care trebuie să le facă în fiecare zi, utilizând sistemul astfel încât timpul pe care îl dedică activităților repetitive să fie cât mai mic.
- Andreea vrea să învețe să genereze și să interpreteze rapoartele despre prezența, notele și progresul elevilor, pentru a putea oferi informații cât mai actuale conducerii liceului.

### **Provocări**

- Fiind la începutul carierei, Andreea are nevoie de ajutor pentru a înțelege procedurile și modul în care funcționează liceul.
- Andreea se descurcă cu tehnologia de bază, însă are foarte puțină experiență în lucrul cu sisteme avansate de gestiune a datelor, motiv pentru care apreciază interfețele simple și intuitive, care au de asemenea explicații ale funcționalităților.
- Ea încearcă să fie precaută în gestionarea datelor sensibile și dorește să evite orice eroare care ar putea afecta evidența elevilor.

### **Nevoi de la Sistemul de Evidență a Elevilor**

- Pentru cineva cu experiență minimă, sistemul trebuie să aibă o interfață intuitivă și ușor de utilizat, să permită introducerea de date rapid, fără a fi nevoie de cunoștințe avansate.
- Andreea ar fi recunoscătoare dacă sistemul ar avea ghiduri pas cu pas și informații suplimentare pentru a învăța funcționalitățile sistemului fără dificultăți.
- Generarea rapoartelor de prezență, note și absențe ar ajuta-o să ofere mai rapid informațiile necesare conducerii, nefiind nevoită să le scrie manual.

## 4.2 Tehnologiile folosite

Pentru dezvoltarea sistemului am folosit mai multe tehnologii, iar în continuare vom vorbi despre ce rol are fiecare.

### 4.2.1 Frontend

Pentru partea de frontend am utilizat următoarele:

- **REACT:** O librărie JavaScript pe care am folosit-o pentru a construi interfața cu utilizatorii. Se pot crea componente care pot să fie reutilizate în mai multe părți ale aplicației. Interfețele sunt actualizate rapid în funcție de interacțiunea utilizatorilor.
- **Bootstrap și React-Bootstrap:** Framework-uri folosite pentru a crea mai rapid interfețele deoarece au componente predefinite(butoane, formulare, bari de navigare etc.), dar care pot fi și modificate.
- **Axios:** O bibliotecă JavaScript utilizată pentru a comunica cu backend-ul prin cereri HTTP.
- **Vite:** Un build tool pentru a dezvolta mai rapid și eficient aplicația frontend. De asemenea, depen

### 4.2.2 Backend

Partea de backend am dezvoltat-o folosind:

- **SpringBoot:** Un framework Java cu ajutorul căruia dezvoltarea aplicațiilor web este mult mai simplă. De asemenea, gestionarea dependențelor e foarte eficientă. Are multe lucruri care sunt gata configurate.
- **Spring Web:** Pentru ca să gestionăm cererile HTTP și să implementăm RestAPI.
- **Spring Security:** Am folosit-o pentru securitatea aplicației, pentru a autentifica și autoriza utilizatorii în funcție de roluri.
- **Spring Data JPA:** Pentru interacțiunea cu baza de date,interogările nu mai e nevoie să fie scrise manual.
- **Flyway:** Instrument folosit pentru a migra baza de date, și astfel toți am avut aceeași versiune actualizată.
- **MySQL Connector:** Un driver JDBC pentru a putea conecta aplicația Java la baza de date MySQL.

### 4.2.3 Baza de date

Ca sistemul de gestiune a bazelor de date(SGBD) am folosit MySQL.

### 4.3 Arhitectura

Sistemul de evidență a elevilor este compus din mai multe module:

- **Modulul Users:** Acest modul se ocupă de informațiile utilizatorilor: secretare(admini), profesori, elevi, părinți.
- **Modulul Clase:** Gestionează informațiile despre clase, cum ar fi elevii din clasă, materiile care se predau în clasă, profesorii care predau materiile.
- **Modulul Materii:** În acest modul sunt specificate materiile care sunt predate.
- **Modulul Absente și Modulul Note:** Se ocupă de gestionarea absențelor și notelor primite de elevi la fiecare materie și în fiecare modul școlar.
- **Modulul Burse:** Gestionează informațiile despre burse și alocarea lor pentru elevi.
- **Modulul Rapoarte:** Este pentru ca profesorii și secretarele să genereze diferite tipuri de raporte.
- **Modulul Notificări:** Acest modul se ocupă de trimiterea notificărilor către profesorii pentru a-i avertiza să pună note pentru modulul care se apropie de final.

Mai jos este o imagine care arată arhitectura generală și interacțiunea dintre module.

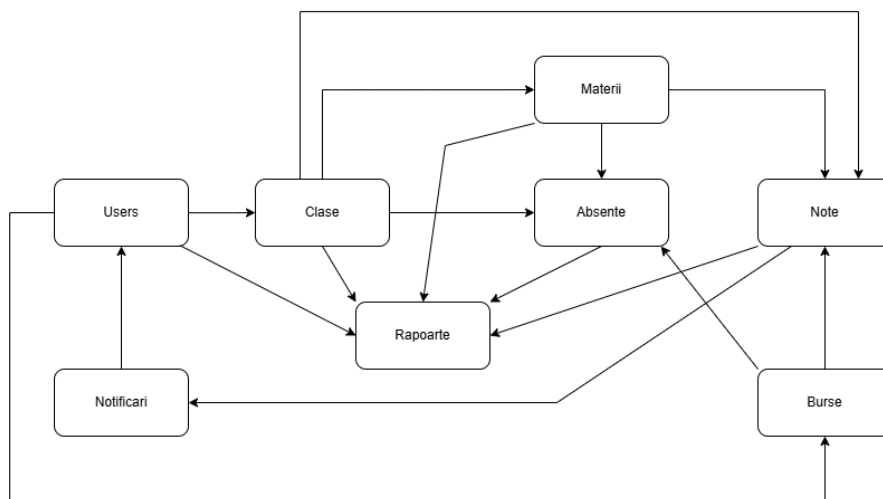


Figure 4: Arhitectura Generală.

Pentru arhitectura detaliată am folosit arhitectura recomandată de Spring-Boot. Așa că vom avea mai multe straturi care se ocupă fiecare de un anumit domeniu:

- **Entitate:** Este stratul care reflectă tabelul din baza de date.
- **Repository:** Se ocupă de interacțiunea cu baza de date, face operații de adăugare, citire, actualizare sau ștergere pe entități.
- **Service:** Conține logica aplicației și este folosit pentru a procesa datele și pentru a face legătura dintre celelalte straturi.

Am folosit și DTO-uri(Data Transfer Object) pentru a transmite datele între straturile aplicației, iar mapper-ele pentru a face conversia între DTO și entități.

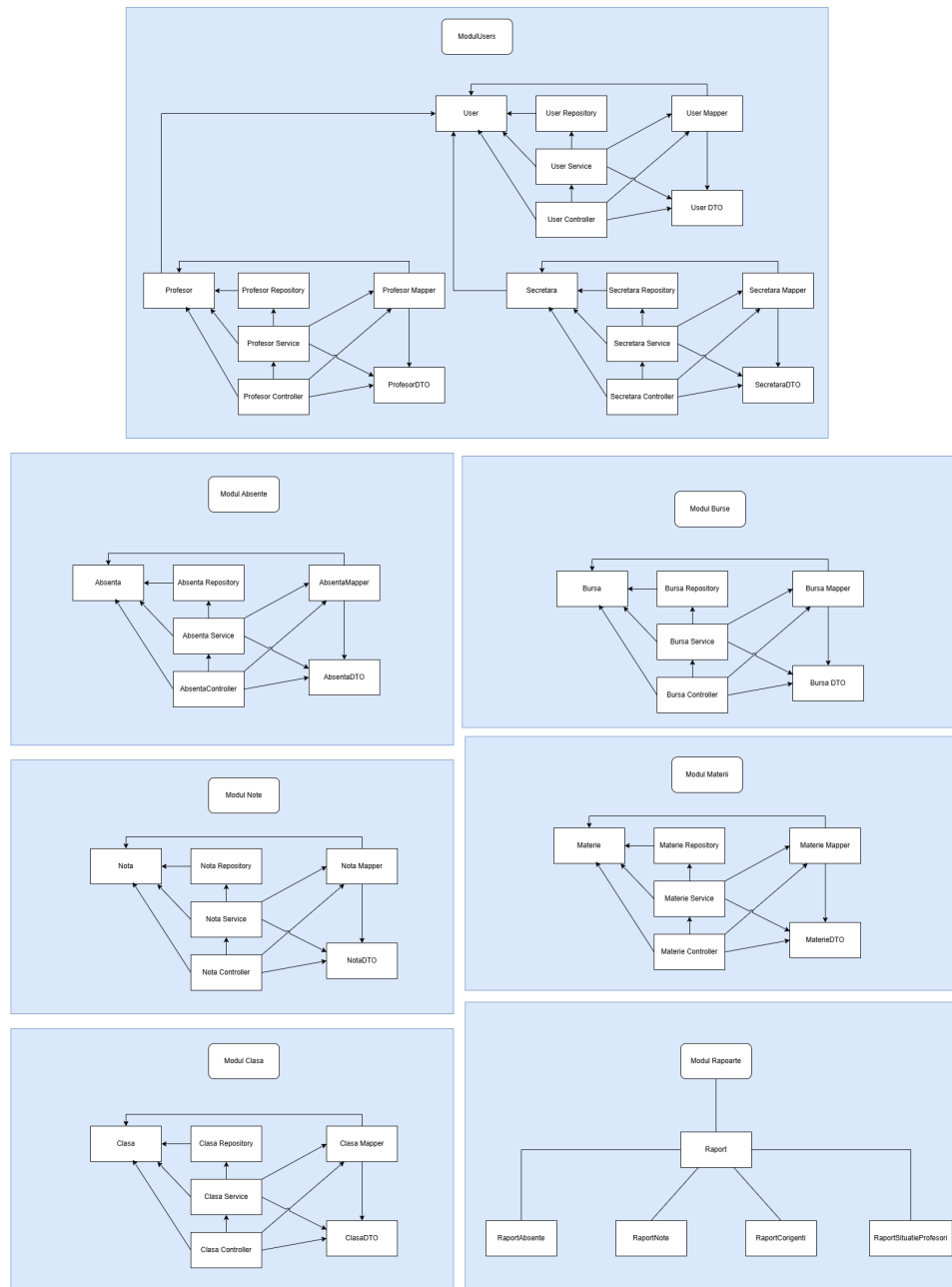


Figure 5: Arhitectura Detaliată.

#### 4.4 Diagrama claselor

Diagrama claselor este unealta care ne ajută să putem vedea structura claselor și relațiile dintre ele. Noi am realizat trei diagrame de clase: una care reflectă entitățile pe care le-am folosit, una care arată clasele modulului absență pentru a vedea cum interacționează toate straturile aplicației și una care expune relațiile dintre servicii. În a doua diagrama am introdus și un controller care este responsabil de gestionarea cererilor HTTP prin implementarea unui API Rest, tot în controller am folosit mappere pentru a converti datele primite de la service.

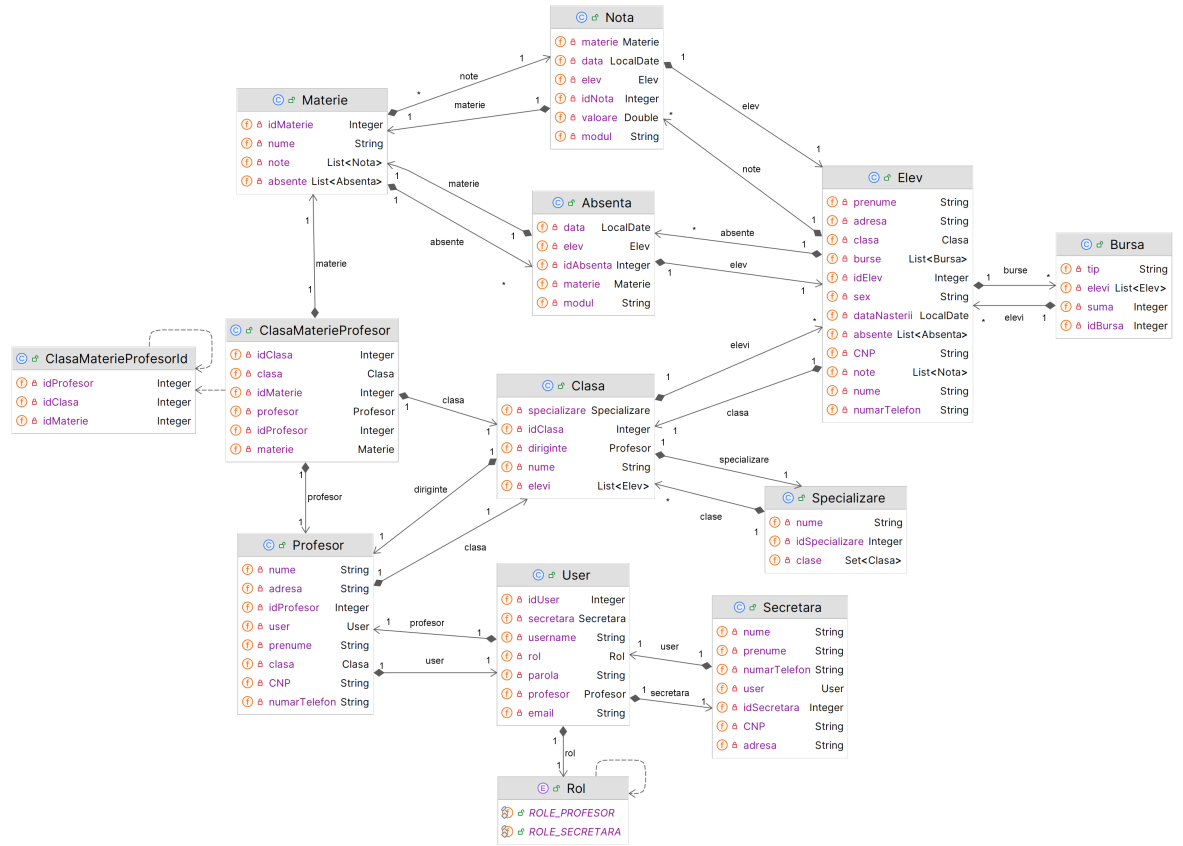


Figure 6: Diagrama claselor pentru entități.

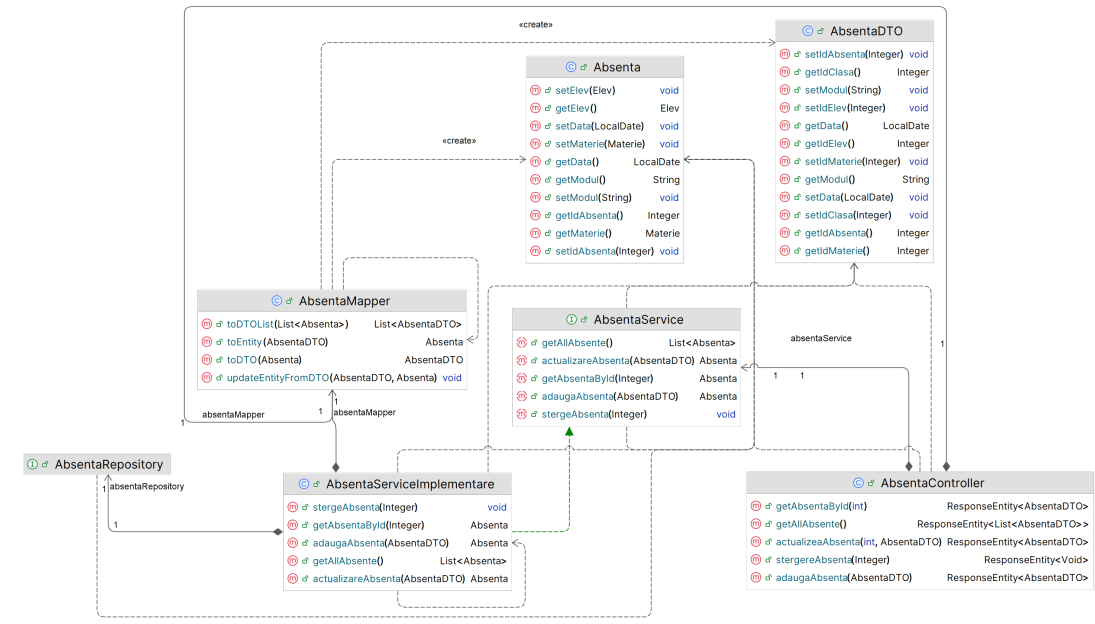


Figure 7: Diagrama claselor pentru modulul absențe.

Celealte module folosesc aceeași structură ca modulul absențe, așadar am realizat o singură diagramă, care poate fi utilizată ca referință pentru a înțelege și celelalte module.



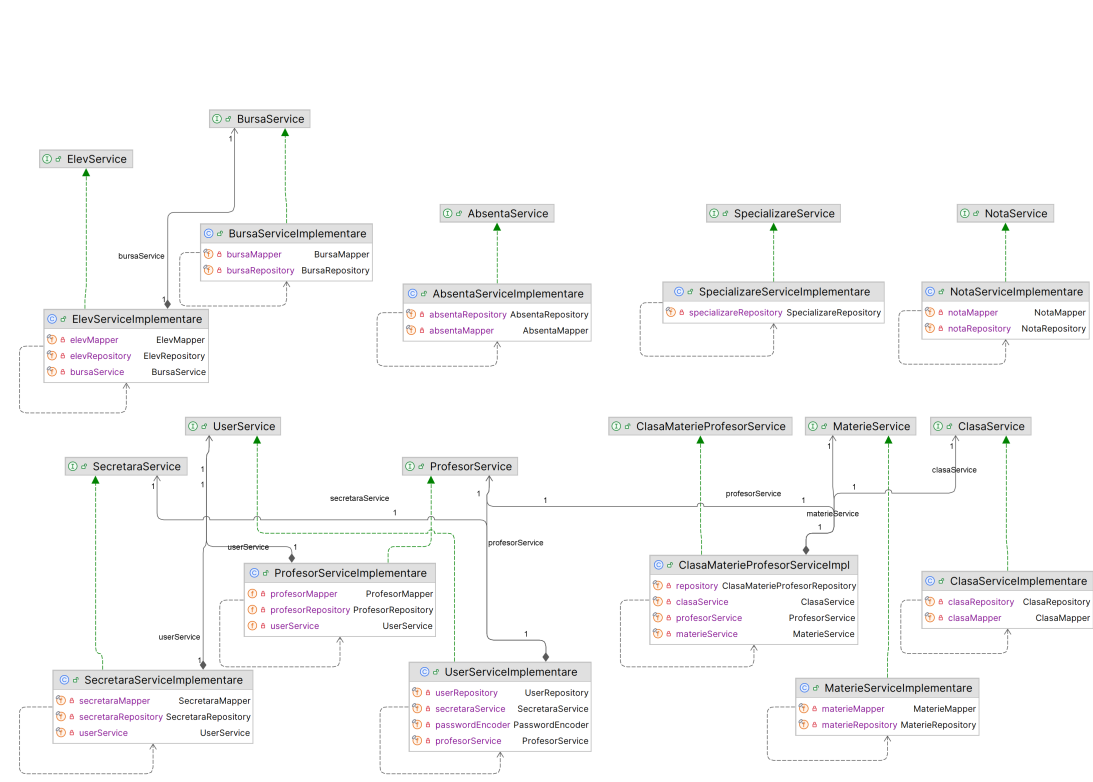


Figure 8: Diagrama claselor pentru servicii.

## 4.5 Implementarea RestAPI

Am implementat API-ul Rest folosind adnotările specifice puse la dispoziție de SpringBoot pentru a configura și gestiona endpoint-urile.

În controller am folosit adnotarea `@RestController` pentru a marca faptul că această clasă va gestiona cererile și răspunsurile HTTP. Am definit endpoint-uri(URL) prin care să poată fi identificate resursele. Pentru fiecare endpoint am folosit o adnotare corespunzătoare în funcție de tipul cererii:

- **@RequestMapping:** adnotare generală pentru a lega cererile HTTP de o anumită clasă sau metodă. Am folosit această adnotare la nivel de clasă pentru a defini endpointul de intrare pentru toate metodele(pentru a accesa oricare metodă trebuie specificat și endpoint-ul de intrare).
- **@GetMapping:** o versiune specializată a adnotării anterioare, folosită pentru a gestiona cererile GET, cu ea se obțin datele de pe server.
- **@PostMapping:** utilizează pentru gestionarea cererilor POST, care sunt responsabile pentru adăugarea unei noi resurse.

- **@PutMapping:** Adnotare folosită pentru a gestiona cererile PUT, care actualizează o resursă care există deja.
- **@DeleteMapping:** adnotare folosită pentru gestionarea cererilor DELETE, care șterg o resursă.
- **@RequestBody:** în cazul cererilor POST și PUT sunt trimise date suplimentare care sunt preluate prin această adnotare.

Pentru endpointuri am folosit parametrii pentru a putea identifica resursa specifică pentru care a fost făcută cererea.

După ce cererile sunt procesate, controller-ul folosește serviciile corespunzătoare pentru a le îndeplini. Răspunsurile sunt mapate în DTO-uri și returnate sub formă de JSON(JavaScript Object Notation).

## 4.6 Implementarea Interfeței

Am realizat 2 dashboard-uri: unul pentru profesori și unul pentru secretare, pentru ca aceștia să poată gestiona informațiile. Pentru ambele structura este similară, au o bară de navigare care include numele liceului și opțiuni pentru cont (vizualizare profil, setări și logout), precum și o bară laterală unde pot să aleagă acțiunea dorită.

### 4.6.1 Dashboard Profesori

Dashboard-ul pentru profesori conține bara de navigare cu cele specificate anterior, iar bara laterală are opțiuni pentru gestionarea elevilor și pentru rapoarte. În pagina de gestionare a elevilor, în partea de sus există un dropdown pentru ca profesorul să selecteze clasa și lângă el o bară de căutare unde să poată căuta elevii după nume. Odată aleasă clasa, elevii apar într-un tabel unde se afișază pentru fiecare numele, clasa și alte 4 butoane cu opțiuni:

- **Detalii:** pentru a vizualiza informațiile personale ale elevului.
- **Note:** pentru a putea vedea și adăuga note.
- **Absențe:** pentru a putea vedea și adăuga absențe.
- **Burse:** pentru a putea bursele primite de elev.

Aceste opțiuni sunt implementate ca modale care apar pe mijlocul ecranului fără a părăsi pagina curentă. Pentru modalele de note și absențe există posibilitatea de filtrare după modul și materie. În funcție de filtrarea aleasă, media este calculată automat, la fel ca numărul de absențe. De exemplu dacă este selectat modulul 1 și materia matematică, va fi afișată media elevului în modulul 1 la matematică.

#### 4.6.2 Dashboard Secretare

Pentru dashboard-ul secretelor este exact aceeași structură cu modale și tabele, dar vor fi câteva funcționalități suplimentare. În bara laterală vor fi opțiuni pentru rapoarte, gestionarea elevilor, profesorilor și conturilor. Pentru pagina de gestionare a elevilor apare în plus un buton de Adaugă pentru ca secretara să poată adăuga un elev nou. În tabelul cu elevi este disponibil un buton de ștergere pentru a putea șterge elevul.

Funcționalități suplimentare pentru modale:

- **Detalii:** apare în plus un buton de edit pentru a putea modifica informațiile elevilor(odată apăsat se va schimba în salvează).
- **Note:** aici apar 2 butoane noi pentru fiecare notă și anume editează și șterge.
- **Absențe:** cele 2 butoane ca și la note.
- **Burse:** butoane de adăugare și ștergere bursă.

Cererile HTTP către backend sunt făcute în fișiere denumite servicii, astfel în componente vor fi apelate doar funcțiile necesare.

Accesarea frontendului duce automat la pagina de login unde utilizatorul trebuie să se autentifice. Odată autentificat, este redirecționat pe baza rolului la dashboard-ul pentru profesori sau secretare. Prin apăsarea butonului de logout userul este deconectat și retrimis din nou la pagina de login.

#### 4.7 Alte Detalii de Implementare

Frontend-ul și backend-ul rulează pe porturi diferite, iar accesarea datelor de pe un port pe altul este restricționată de browser, așa că a fost necesar să configurăm CORS(Cross-Origin Resource Sharing) pentru a permite comunicarea dintre ele. Acest lucru l-am realizat în backend configurând SpringSecurity să permită accesul pentru portul pe care rulează frontend-ul.

În cadrul sistemului, pentru adăugarea unui cont de utilizator, vor trebui specificate și informațiile personale necesare în funcție de rol. De exemplu dacă dorim să adăugăm un user nou (username, parolă, email) care are rol de profesor, va trebui la momentul creării să se introducă și informațiile personale(nume, prenume, numar de telefon, adresa etc.).

Parolele userilor sunt criptate pentru a asigura protecția conturilor.

Fără o bază de date, sistemul nu ar putea să funcționeze, așadar am proiectat și implementat baza de date în MySQL. Tabelul clase-materii-profesori l-am adăugat pentru a reprezenta relația dintre cele trei tabele și pentru a putea identifica ce materii sunt predate de un profesor într-o anumită clasă. Schema rezultată se găsește mai jos.



## 5 Testare și validare

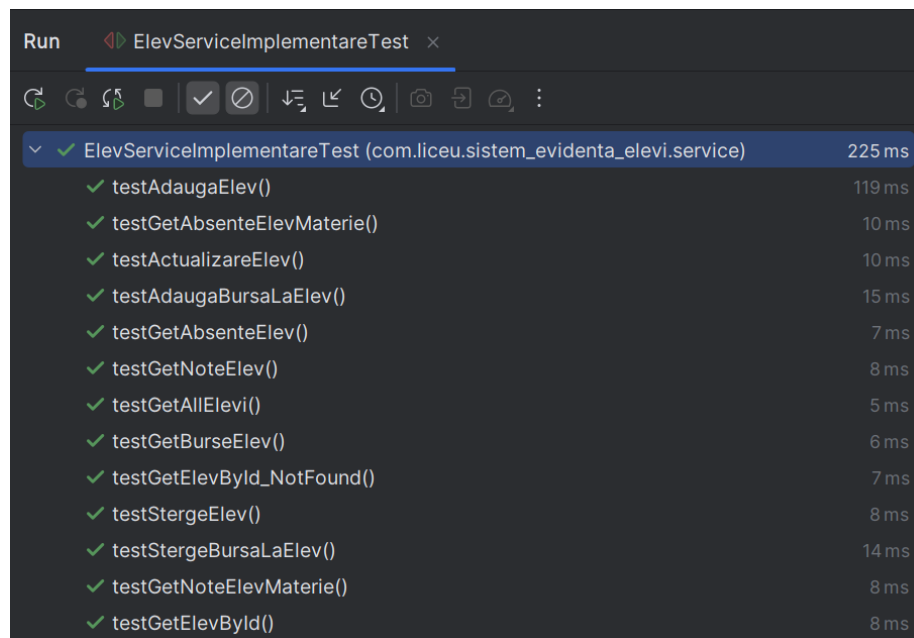
Această secțiune va cuprinde detalii despre testarea pe care am făcut-o pentru a verifica dacă aplicația funcționează corect.

### 5.1 Unit Test

Am dezvoltat unit test pentru entități și servicii. Aceste teste ne-au ajutat să identificăm erorile pe care le-am avut în logica de implementare.

Pentru entități am testat ca atributele să fie inițializate corect, getteri și setteri să funcționeze (aici am descoperit de mai multe ori că am uitat să punem getteri și setteri pentru toate atributele), relațiile dintre entități să fie configurate corect.

Serviciile au fost testate folosind Mockito pentru a crea mockuri ale dependențelor (alte servicii) și astfel am putut testa fiecare metodă a serviciului în parte. Aici am descoperit mai multe erori, una dintre ele fiind la actualizarea entităților faptul că după ce am setat datele noi, am returnat entitatea fără a o salva în baza de date.



Test Name	Duration
✓ ElevServiceImplementareTest (com.liceu.sistem_evidenta_elevi.service)	225 ms
✓ testAdaugaElev()	119 ms
✓ testGetAbsenteElevMaterie()	10 ms
✓ testActualizareElev()	10 ms
✓ testAdaugaBursaLaElev()	15 ms
✓ testGetAbsenteElev()	7 ms
✓ testGetNoteElev()	8 ms
✓ testGetAllElevi()	5 ms
✓ testGetBurseElev()	6 ms
✓ testGetElevById_NotFound()	7 ms
✓ testStergeElev()	8 ms
✓ testStergeBursaLaElev()	14 ms
✓ testGetNoteElevMaterie()	8 ms
✓ testGetElevById()	8 ms

Figure 10: Rezultate teste pentru ElevService.

## 5.2 Test de Integrare

Pentru a verifica cum funcționează API-ul am făcut teste de integrare folosind Postman. Prin această testare am putut să ne asigurăm că cererile de GET, POST, PUT și DELETE sunt procesate corect și datele sunt returnate în modul dorit. Codurile de stare HTTP ne-au ajutat să aflăm mai exact ce nu funcționează corect. De exemplu, dacă codul de eroare era 404, dar în baza de date aveam resursa cerută, am realizat că am scris greșit endpoint-ul. Aici am găsit o eroare care ne-a determinat să schimbăm modul în care transmitem datele către frontend. Când am făcut un GET request pentru a obține elevii din clase rezultatul a fost o buclă infinită din cauza relației dintre entitățile Clasa și Elev. După ce am căutat informații despre cum să rezolvăm eroare, am hotărât să nu mai transmitem entitățile direct, ci să creăm DTO-uri pentru a transmite datele.

## 6 Rezultate

Aici vom prezenta ce rezultate am obținut după ce am dezvoltat sistemul de evidență a elevilor.

### 6.1 Funcționalități implementate

După foarte multă muncă, am reușit să implementăm următoarele funcționalități în cadrul sistemului:

- **Gestionarea Informațiilor Elevilor:** Sistemul nostru permite adăugarea, editarea și ștergerea informațiilor despre elevilor. Un elev poate fi adăugat, datele sale personale pot fi modificate și de asemenea, elevul poate fi șters din sistem.
- **Gestionarea Notelor și Absențelor:** Profesorii pot adăuga note și absențe pentru fiecare elev la materia pe care o predau. Sistemul calculează automat media și numărul de absențe pentru fiecare materie.
- **Căutare:** A fost implementate funcții de căutare în funcție de numele elevilor, clasă, materie și modul pentru ca profesorii și secretarele să găsească rapid informațiile dorite.
- **Control de acces bazat pe roluri(RBAC) în Frontend:** În interfețe userii vor putea vedea doar funcționalitățile permise pentru rolul pe care îl au.

Așadar toate cerințele specificate în analiza MoSCoW la secțiunea MUST HAVE au fost îndeplinite.

## 6.2 Capturi de Ecran pentru Interfețe

### 6.2.1 Dashboard pentru secretare

The screenshot shows a web application interface for managing students. The header includes the school name 'Colegiul Tehnic "George Baritiu"' and a 'Profil' dropdown. A sidebar on the left contains navigation links: 'Gestionare Elevi', 'Gestionare Profesori', 'Gestionare Conturi', and 'Raporte'. The main content area is titled 'Gestioneaza Elevi' and features a search bar with a 'Selectati clasa' dropdown and a 'Cauta dupa nume' input field. Below the search bar is an 'Adauga Elev' button. A table lists students with columns for ID, Nume, and Clasa. Each row includes buttons for 'Detalii', 'Note', 'Absente', 'Burse', and 'Sterge'.

ID	Nume	Clasa	Detalii	Note	Absente	Burse	Sterge
1	Ciobanu Codruț	10A					
2	Gabriela Florescu	10A					
4	Popescu George	10A					

Figure 11: Dashboard Secretara.

The screenshot shows a modal form titled 'Adauga Elev' with a close button. The form contains the following fields: 'Nume' and 'Prenume' (text inputs), 'CNP' (text input), 'Data Nasterii' (date picker with format mm/dd/yyyy), 'Adresa' (text input), 'Numar Telefon' (text input), and 'Sex' (dropdown menu with 'Selectează Sex' as the current selection). A green 'Adauga' button is at the bottom.

Figure 12: Modal adaugare elev.

**Detalii Ciobanu Codruț**

Nume:  Prenume:

CNP:  Data Nasterii:

Adresa:

Numar Telefon:  Sex:

**Edit**

Figure 13: Modal cu detaliile elevului.

**Notele pentru Ciobanu Codruț**

Modul:  Materie:

Nota	Data	
<input type="text"/>	<input type="text" value="mm/dd/yyyy"/>	<b>Adaugă Nota</b>
9	10/16/2024	<b>Editează</b> <b>Șterge</b>
8	12/27/2024	<b>Editează</b> <b>Șterge</b>
4	12/18/2024	<b>Editează</b> <b>Șterge</b>

Media: 7.00

Figure 14: Modal cu notele elevului.



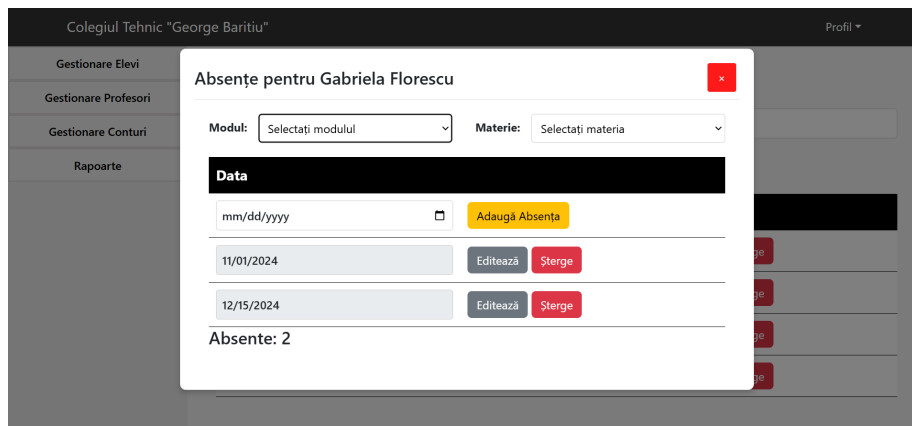


Figure 15: Modal cu absentele elevului.

### 6.2.2 Dashboard pentru profesori

Modalaele pentru note abesnte si detalii sunt la fel ca la dashboard-ul secretarelor, doar că lipsesc butoanele de editare și ștergere.

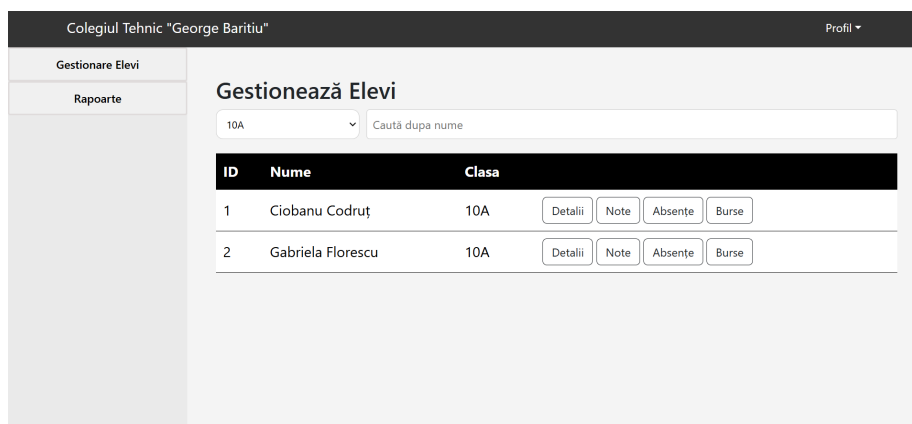


Figure 16: Dashboard Profesor.

### 6.3 Funcționalități neimplementate

În principiu, cele mai importante funcționalități pentru a putea gestiona elevii au fost implementate, însă din cauza lipsei de timp nu au fost implementate următoarele:

- Interfața pentru gestionarea burselor nu a fost implementată, dar în partea de backend există.
- Modulul de rapoarte nu a fost integrat.
- Interfața pentru adăugarea, modificarea și ștergerea conturilor de utilizator nu a fost realizată în totalitate.
- Controlul de acces bazat pe roluri (RBAC) nu a fost dezvoltat în backend.
- Posibilitatea de a vedea și modifica profilul contului propriu de către utilizatori.
- Notificările pentru ca profesori să fie avertizați să introducă note înainte de sfârșitul modului nu au fost introduse.
- Vizualizarea notelor și absențelor de către elevi și părinți nu este posibilă.
- Algoritmul pentru distribuirea burselor nu a fost realizat.
- Vizualizarea situației școlare din anii anteriori nu a fost introdusă.

### 6.4 Posibile Îmbunătățiri

Pentru funcționalitățile implementate cu succes există lucruri care pot fi îmbunătățite.

- Pentru interfață s-ar putea modifica culorile pentru ca utilizatorul să aibă o experiență mai bună.
- Pentru modalele de note și absențe materiile sunt afișate doar dacă în pagina de gestionare elevi este selectată clasa din care face parte, deci dacă secretara caută levul după nume nu poate să deschidă direct modalele ci trebuie să selecteze mai întâi clasa. Lucrul acesta poate fi îmbunătățit prin afișarea materiilor în funcție de clasa din care face parte elevul, fără a fi nevoie de selecție.
- Tot în modalele de note și absențe tabelul trebuie îmbunătățit deoarece dacă apar mai multe note sau absențe, modalul se deformează și iese din pagină.
- Când se apasă butonul de ștergere, să apară un modal de confirmare pentru ca să nu se șteargă ceva din greșeală.
- Adăugare validări pentru DTO-uri și pentru formulare.
- Crearea de excepții pentru fiecare modul în backend.

## 7 Concluzii

În acest proiect am dezvoltat un sistem de evidență a elevilor care îndeplinește nevoile liceelor în ceea ce privește introducerea tehnologiei în procesul educațional. Sistemul a fost implementat pentru ca gestionarea informațiilor despre elevi, notele și absențele să fie una simplă și eficientă pentru profesori și secretare, având ca rezultat o administrare mai bună a liceului. Am reușit să implementăm cele mai importante funcționalități, și anume cele legate de gestionarea notelor, absențelor și informațiile personale ale elevilor.

Sistemul are o interfață simplă și ușor de utilizat pentru ca utilizatorii să interacționeze cu datele. Am automatizat calcularea mediilor și numărului de absențe pentru elevi după diferite criterii, ceea ce reduce semnificativ timpul pe care profesorii sau secretarele trebuie să îl aloce acestui aspect.

Deși funcționalitățile esențiale au fost implementate, sunt și altele care pot fi adăugate pentru a îmbunătăți sistemul, cum ar fi interfața de gestionare a burselor și modulul de generare a rapoartelor.

Pe parcursul implementării am întâmpinat și dificultăți. De exemplu partea de conectare a frontend-ului la backend a fost una dificilă, deoarece fiind prima dată când facem asta am dat de multe erori pe care a fost nevoie de foarte mult timp să le rezolvăm. Dezvoltarea acestui sistem a fost o oportunitate de a învăța și dobândi cunoștințe noi. Înainte de a începe acest proiect nu aveam idee despre ce înseamnă arhitectura client-server sau despre ce este și cum funcționează un RestAPI, dar în timpul realizării proiectului am acumulat cunoștințe despre toate acestea.

În concluzie, sistemul de evidență va aduce o schimbare semnificativă în bine pentru orice liceu care dorește să digitalizeze procesul de gestionare a elevilor. Continuarea dezvoltării și îmbunătățirii sistemului va face ca procesul de gestiune a elevilor din liceu să fie unul foarte eficient din toate punctele de vedere, transformând astfel munca profesorilor și secretarelor într-o experiență plăcută.

## 8 Funcționarea ca echipă

### 8.1 Rolurile membrilor

Echipa noastră a fost formată din 4 membri, și fiecare am avut câte un rol sau mai multe în cadrul proiectului:

- **Sălnicean Răzvan:** a avut rolul de tester și dezvoltator backend.
- **Bușecan Andrei:** a fost responsabil cu implementarea unei părți din frontend și cu realizarea documentației JavaDoc pentru backend.
- **Terpe Victor:** a avut rolul de dezvoltator frontend principal.
- **Titea Ruben:** a fost responsabil cu dezvoltarea părții de backend, integrarea frontend-ului cu backend-ul, gestionarea bazei de date și coordonarea echipei.

### 8.2 Îndeplinirea rolurilor

Fiecare membru și-a îndeplinit rolul său, dar ne-am și ajutat unii pe alții când au apărut probleme.

Realizările fiecăruia:

- **Sălnicean Răzvan:** A întocmit testările necesare pentru toate entitățile și serviciile folosite. A contribuit la implementarea modului de clase, materii și note în partea de backend. Pentru frontend a ajutat la realizarea barelor de navigare(laterala si cea de sus) și la componenta tabelului de elevi.
- **Bușecan Andrei:** A realizat interfața pentru dashboard-ul profesorilor și modulele pentru adăugarea și vizualizarea detaliilor elevilor. A documentat mapperele, serviciile și repository-urile folosite în backend.
- **Terpe Victor:** A realizat proiectul pentru frontend, a proiectat modul în care să funcționeze și să arate interfețele și a realizat structura componentelor. A implementat pagina de login, interfața pentru dashboard-ul profesorilor, componentele pentru gestionarea conturilor și profesorilor, dropdown-ul pentru selectarea clasei și un modal de confirmare care nu a fost integrat în sistemul final.
- **Titea Ruben:** A coordonat echipa prin atribuirea sarcinilor și i-a ajutat pe ceilalți când au întâmpinat probleme. A proiectat și implementat baza de date a sistemului și a făcut migrările. A realizat proiectul pentru backend, adăugarea dependențelor și configurărilor necesare pentru funcționarea aplicației, a implementat modulul de useri, clase, materii, note, absente și burse în backend. De asemenea, a făcut și conectarea dintre frontend și backend prin implementarea și testarea API-ului Rest, creare serviciilor corespunzătoare pentru a face cererile HTTP

și încărcarea datelor primite ca răspuns în stările componentelor din frontend. Tot în frontend a realizat componenta de gestionare a elevilor, modulele pentru note și absențe. A fost responsabil de autentificarea utilizatorilor atât în frontend cât și în backend și a realizat documentarea componentelor din frontend folosind JSDoc precum și pe cea a entităților, DTO-urilor, controller-elor și a claselor folosite pentru configurarea securității.

### 8.3 Probleme întâlnite

Inițial, am început să realizăm proiectul folosind Java simplu și Swing, dar în urma sugestiei de a folosi un framework făcute de către domnul profesor Erdei Rudolf, am hotărât să folosim SpringBoot. După ce ne-am documentat despre acest framework, am descoperit că este o combinație bună cu React, și având un alt curs la care am început să învățăm React, ne-am decis să facem o aplicație web folosind aceste 2 tehnologii. Deoarece nu am mai folosit aceste tehnologii niciodată a fost nevoie ca mai întâi să le învățăm, lucru care ar fi durat ceva timp, așa că am învățat doar elementele de bază despre cum funcționează, iar pe parcursul realizării proiectului am învățat și altele.

O altă problemă pe care am avut-o a fost gestionarea timpului, adică găsirea unei perioade de timp în care să fim toți disponibili pentru a ne întâlni deoarece fiecare aveam și alte responsabilități. Pentru a rezolva problema am lucrat la proiect în timpul pauzelor de la facultate și când aveam timp liber între cursuri. De asemenea, ne-am întâlnit și pe discord pentru a putea comunica.

În final, realizarea acestui proiect a fost o experiență interesantă, deși nu a fost una simplă. Am învățat o mulțime de lucruri noi, dar mai important este că am reușit să le și implementăm.

## References

- [1] Koen Timmers. Evolution of technology in the classroom. In *Teaching in the Fourth Industrial Revolution*, pages 106–123. Routledge, 2018.
- [2] School of Education, American University. How important is technology in education? benefits, challenges, and impact on students, June 2020.
- [3] Rahat Raja and PC Nagasubramani. Impact of modern technology in education. *Journal of Applied and Advanced Research*, 3(1):33–35, 2018.
- [4] Eric Carriere. What is a student management system?, October 2024.
- [5] Yousef A Baker El-Ebiary, Najeeb Abbas Al-Sammarraie, Yazeed Al Moaiad, and Mohammad Mahmoud Saleem Alzubi. The impact of management information system in educational organizations processes. In *2016 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, pages 166–169. IEEE, 2016.

- [6] Xinyu Zhang, Vincent CS Lee, Duo Xu, Jun Chen, and Mohammad S. Obaidat. An effective learning management system for revealing student performance attributes, 2024.
- [7] Classe365. What is a student information system (sis): Features and benefits, October 2024.
- [8] The Editors of Encyclopaedia Britannica. Client-server architecture, 2024. Ultima actualizare: Noiembrie 6, 2024.
- [9] Haroon Shakirat Oluwatosin. Client-server model. *IOSR Journal of Computer Engineering*, 16(1):67–71, 2014.
- [10] Roy T. Fielding, Mark Nottingham, and Julian Reschke. HTTP Semantics. RFC 9110, June 2022.
- [11] Cássio Cappellari. Fundamentals of rest api. <https://dev.to/cassiocappellari/fundamentals-of-rest-api-2nag>, August 2021.