

Miloiu Cristi  
Grupa 432A

## Documentatie Proiect Crud-App Typescript Programarea Interfețelor pentru Baze de Date

Github: <https://github.com/cristim67/Crud-App>

Link website: <https://pibd.cristimiloiu.com/>

### Introducere:

Proiectul are ca scop implementarea unui sistem de gestionare a datelor pentru o instituție academică, folosind tehnologii moderne și eficiente.

### Tehnologii utilizate:

Version Control: Git, Github, GitHub Actions.

**Git** oferă posibilitatea de urmărire a modificărilor în codul sursă al proiectului. Acesta permite dezvoltatorilor să lucreze colaborativ, gestionând schimbările în timp și furnizând un istoric detaliat al fiecărei versiuni.

**GitHub** reprezintă o platformă de hosting pentru proiecte gestionate cu ajutorul sistemului Git.

**GitHub Actions** este un serviciu oferit de GitHub pentru automatizarea fluxurilor de lucru în cadrul proiectelor software. Această platformă permite configurarea și rularea automată a diverselor acțiuni în răspuns la evenimente specifice, cum ar fi push-uri, pull requests sau crearea de tag-uri. GitHub Actions facilitează integrarea continua (CI) și livrarea continua (CD) într-un mod flexibil și personalizat.

### Standardizarea și identificarea automată a erorilor: ESLint

**ESLint** este o unealtă de analiză statică a codului sursă pentru JavaScript și TypeScript, utilizată pentru identificarea și corectarea erorilor de stil, neregulilor și a altor probleme potențiale. ESLint ajută la menținerea unui cod sursă curat, coerent și în conformitate cu standardele definite de proiect și echipă.

Backend: Typescript, Genezio, Sequelize, Postgresql.

**Typescript** este un limbaj de programare tipizat care adaugă tipuri statice peste JavaScript. Backend-ul este dezvoltat în TypeScript pentru a asigura calitatea și corectitudinea codului, fiind mai ușor de scalat pe viitor.

**Genezio** reprezintă o paradigmă de cloud native care facilitează dezvoltarea simplificată a aplicațiilor moderne. Aceasta furnizează nu doar opțiuni pentru găzduirea aplicațiilor, ci și o modalitate mai accesibilă de a scrie backend-ul în clase, ce ulterior vor fi implementate sub forma unei aplicații serverless.

**Sequelize** este un ORM (Object-Relational Mapping) pentru Node.js, care facilitează interacțiunea cu bazele de date relaționale, cum ar fi PostgreSQL, MySQL și SQLite. ORM-ul acționează ca un strat intermediar între aplicația Node.js și baza de date, permițând dezvoltatorilor să opereze cu datele utilizând obiecte și modele JavaScript în locul limbajului specific bazei de date.

**Postgresql** este un sistem de gestiune a bazelor de date relaționale, oferind o structură robustă pentru manipularea datelor. A fost ales pentru a susține stocarea eficientă a informațiilor.

Frontend: React cu TypeScript, Vite și Tailwind CSS

**React** este o bibliotecă JavaScript pentru construirea interfețelor de utilizator, iar TypeScript adaugă tipuri statice pentru a îmbunătăți dezvoltarea și mentenanța codului.

**Vite** este un instrument de construire a proiectului React extrem de rapid, care optimizează procesul de dezvoltare prin intermediul importurilor ESM (ECMAScript Modules). Acesta oferă o experiență de dezvoltare foarte eficientă. Sistemul de bundling fiind scris în Rust.

**Tailwind CSS** este un framework de CSS utilitar care permite construirea rapidă și eficientă a interfețelor de utilizator. El furnizează clase predefinite pentru stilizarea elementelor, facilitând astfel procesul de dezvoltare.

# Implementare si functionalitati.

Baza de date este formată din 4 tabele. Structura acestora fiind:

A. Tabela studenți are următoarele coloane [1]:

Name	students	Primary	id	Search column name...				
#	column_name	data_type	is_nullable	check	column_default	foreign_key	comment	
1	id	uuid	NO	id	uuid_generate_v4()	EMPTY	→ NULL	
2	firstName	varchar(255)	YES	firstName	NULL	EMPTY	→ NULL	
3	lastName	varchar(255)	YES	lastName	NULL	EMPTY	→ NULL	
4	birthDate	timestamptz	YES	birthDate	NULL	EMPTY	→ NULL	
5	address	varchar(255)	YES	address	NULL	EMPTY	→ NULL	
6	email	varchar(255)	YES	email	NULL	EMPTY	→ NULL	
7	phone	varchar(20)	YES	phone	NULL	EMPTY	→ NULL	
8	createdAt	timestamptz	YES	createdAt	NULL	EMPTY	→ NULL	

Fig 1: Structură tabela studenți.

B. Tabela subiecte are următoarele coloane [2].

#	column_name	data_type	is_nullable	check	column_default	foreign_key	comment	
1	id	uuid	NO	id	uuid_generate_v4()	EMPTY	→ NULL	
2	subjectName	varchar(255)	YES	subjectName	NULL	EMPTY	→ NULL	
3	subjectDescription	varchar(255)	YES	subjectDescription	NULL	EMPTY	→ NULL	
4	professorId	uuid	YES	professorId	NULL	EMPTY	→ NULL	
5	createdAt	timestamptz	YES	createdAt	NULL	EMPTY	→ NULL	

Fig 2: Structura tabela subiecte.

C. Tabela profesori are următoarele coloane [3].

Name	professors	Primary	id	Search column name...				
#	column_name	data_type	is_nullable	check	column_default	foreign_key	comment	
1	id	uuid	NO	id	uuid_generate_v4()	EMPTY	→ NULL	
2	firstName	varchar(255)	YES	firstName	NULL	EMPTY	→ NULL	
3	lastName	varchar(255)	YES	lastName	NULL	EMPTY	→ NULL	
4	email	varchar(255)	YES	email	NULL	EMPTY	→ NULL	
5	createdAt	timestamptz	YES	createdAt	NULL	EMPTY	→ NULL	

Fig 3: Structura tabela profesori

D. RegisterStudentSubject

Name	registerStudentSubject	Primary	id	Search column name...				
#	column_name	data_type	is_nullable	check	column_default	foreign_key	comment	
1	id	uuid	NO	id	uuid_generate_v4()	EMPTY	→ NULL	
2	studentId	uuid	YES	studentId	NULL	EMPTY	→ NULL	
3	subjectId	uuid	YES	subjectId	NULL	EMPTY	→ NULL	
4	grade	int4	YES	grade	NULL	EMPTY	→ NULL	
5	dateRegistered	timestamptz	YES	dateRegistered	NULL	EMPTY	→ NULL	
6	createdAt	timestamptz	YES	createdAt	NULL	EMPTY	→ NULL	

Fig 4: Structura tabela înregistrare legatura student-subiect

Diagrama bazei de date asociată tabelelor este reprezentată în figura de mai jos [5]:

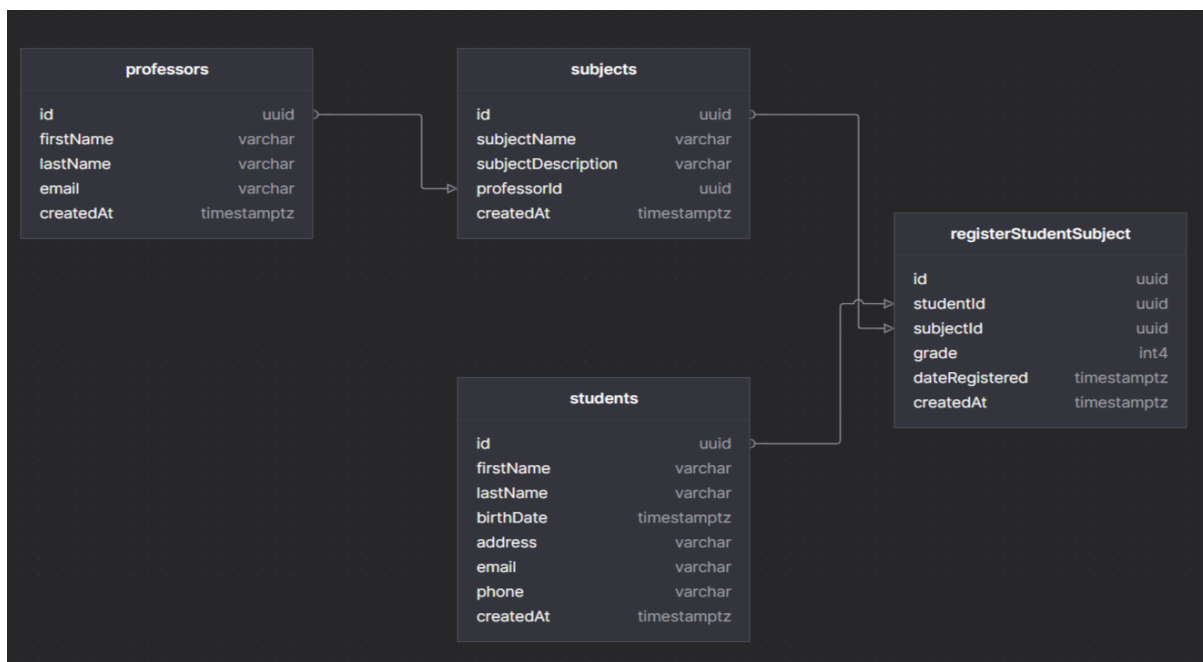


Fig 5: Diagrama bazei de date

Pe partea de backend aplicația are o clasă care știe să facă următoarele operații [6]:

```
2  /**
3   * @class BackendService
4   * @method createStudent
5   * @method createSubject
6   * @method createProfessor
7   * @method createRegisterStudentSubject
8   * @method getStudents
9   * @method getSubjects
10  * @method getProfessors
11  * @method getRegisterStudentSubject
12  * @method deleteStudent
13  * @method deleteSubject
14  * @method deleteProfessor
15  * @method deleteRegisterStudentSubject
16  * @method updateStudent
17  * @method updateSubject
18  * @method updateProfessor
19  * @method updateRegisterStudentSubject
20  * @method searchStudentbyId
21  * @method searchSubjectbyId
22  * @method searchProfessorbyId
23  * @method searchRegisterStudentSubjectbyId
24  * @description This class is responsible for managing the database operations.
25  */
```

Fig 6: Clase BackendService, estructura.

Functionalitati principale CRUD:

CRUD - Create, read, update, delete.

- A. Create - Functionalitatea oferă posibilitatea de a crea o entitate pentru oricare tabela existența.

Exemplu - Crearea unui student [7],[8],[9].

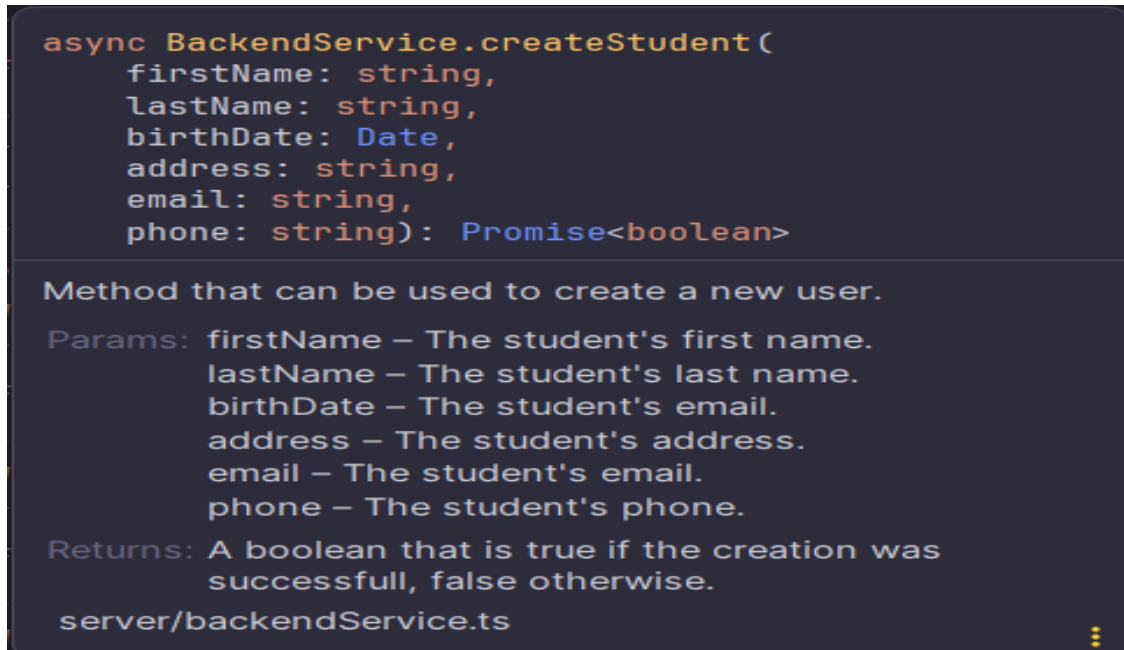


Fig 7: Descrierea metodei createStudent.

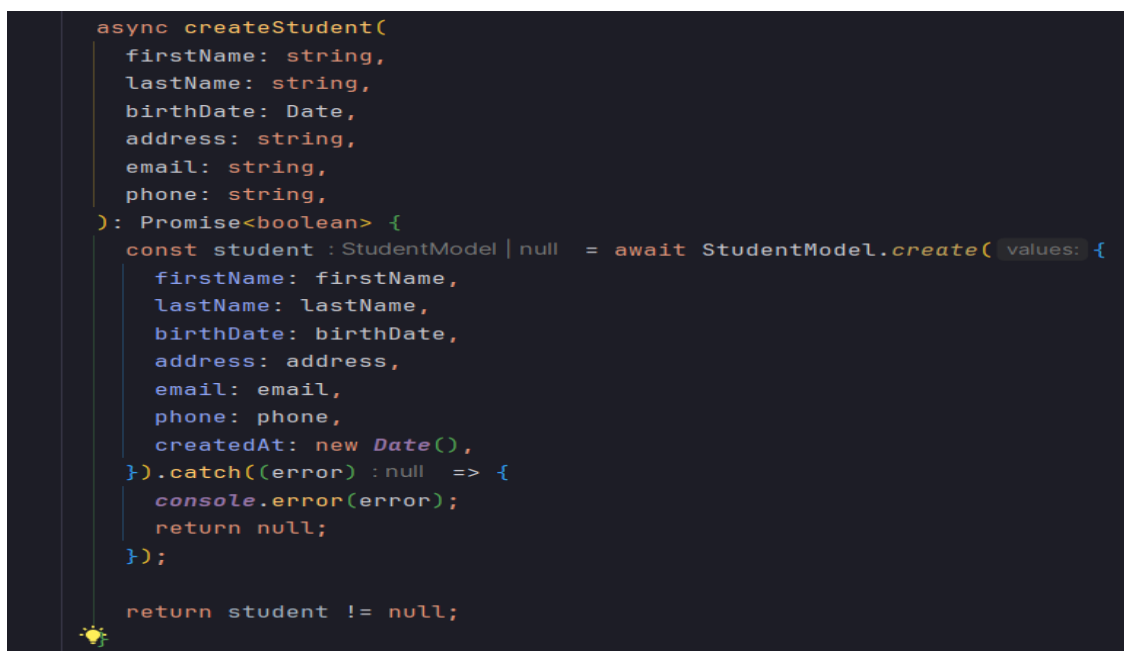


Fig 8: Codul sursă al metodei createStudent.

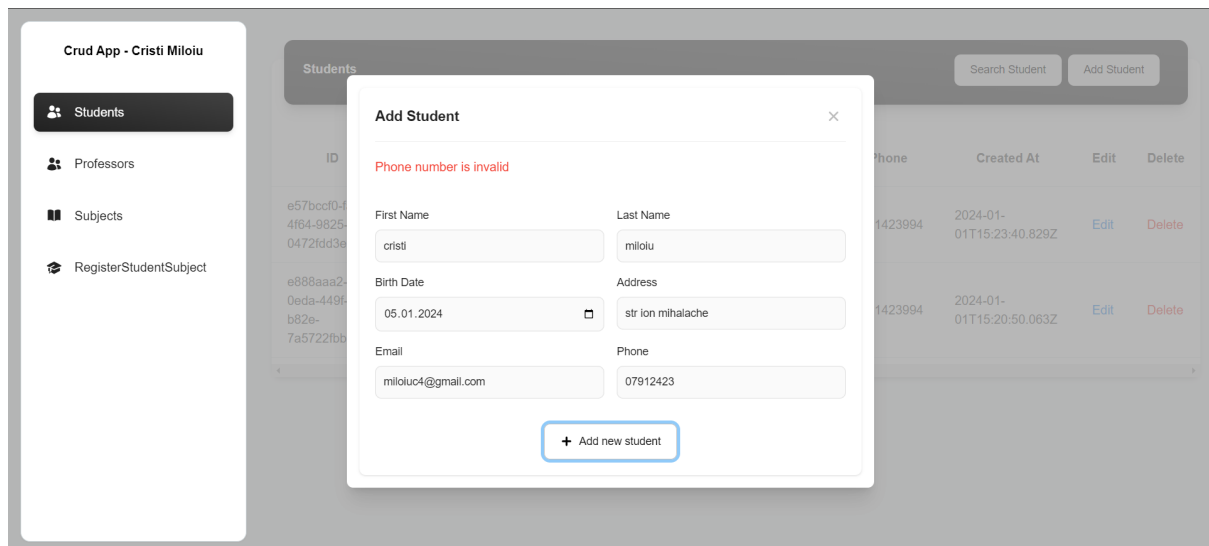


Fig 9: Interfata grafica a Modalului pentru adaugarea unui student

B. Read - Functionalitatea oferă posibilitatea de a citi, toate entitățile unei tabele.

Exemplu - Citirea tuturor studentilor [10], [11], [12].

```

async BackendService.getStudents(): Promise<StudentType[]>

Method that can be used to get all the students.

Returns: An array of students.

server/backendService.ts

```

Fig 10: Descrierea metodei getStudents.

```

async getStudents(): Promise<StudentType[]> {
  const students : StudentModel[] | null = await StudentModel.findAll( options: {
    order: [["createdAt", "DESC"]],
  }).catch((error) : null => {
    console.error(error);
    return null;
  });

  return students || [];
}

```

Fig 11: Codul sursă pentru metoda getStudents.

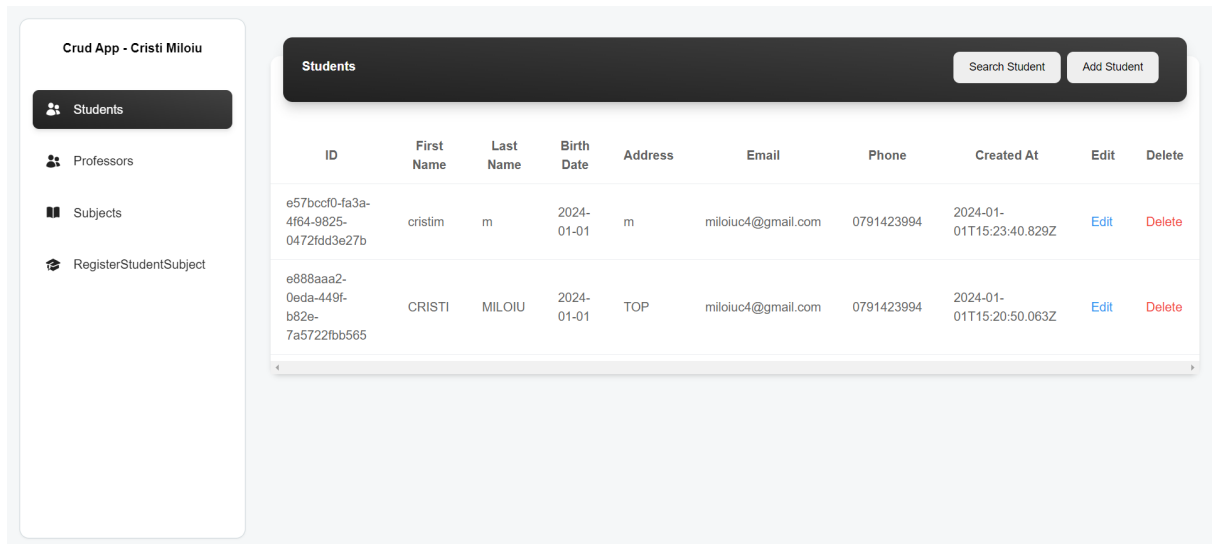


Fig 12: Interfața grafică de afișare a studenților.

C. Update - Funcționalitatea oferă posibilitatea de a edita proprietățile unei entități specifice.

Exemplu - Modificarea datelor unui student [12], [13], [14].

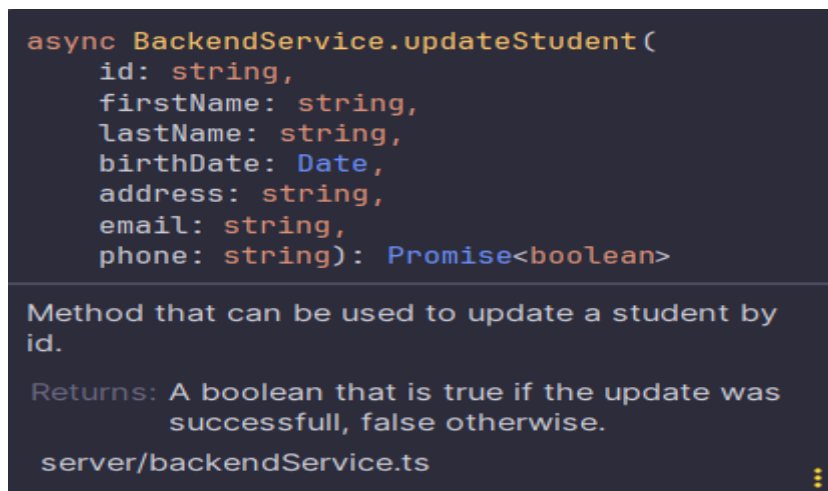


Fig 12: Descrierea metodei updateStudent.

```

1+ usages  cristim67
async updateStudent(
  id: string,
  firstName: string,
  lastName: string,
  birthDate: Date,
  address: string,
  email: string,
  phone: string,
): Promise<boolean> {
  const student : [affectedCount: number] | null = await StudentModel.update(
    values: {
      firstName: firstName,
      lastName: lastName,
      birthDate: birthDate,
      address: address,
      email: email,
      phone: phone,
    },
    options: {
      where: {
        id: id,
      },
    },
  ).catch((error) : null => {
    console.error(error);
    return null;
  });

  return student != null;
}

```

Fig 13: Codul sursă pentru metoda updateStudent.

Crud App - Cristi Miloiu

- Students
- Professors
- Subjects
- RegisterStudentSubject

Students

Search Student Add Student

**Edit Student**

Phone number is invalid

First Name: cristim Last Name: m

Birth Date: 01.01.2024 Address: m

Email: miloluc4@gmail.com Phone: 07914239922

SAVE CHANGES CANCEL

ID	Phone	Created At	Edit	Delete
e57bccf0-4f64-9825-0472fdd3e...	1423994	2024-01-01T15:23:40.829Z	Edit	Delete
e888aaa2-0eda-449f-b82e-7a572fbb...	1423994	2024-01-01T15:20:50.063Z	Edit	Delete

Fig14: Interfata grafica pentru Modalul care se ocupă cu actualizarea unui student



D. Delete - Functionalitatea oferă posibilitatea de a șterge o entitate specifice.

Exemplu - Ștergerea unui student [14], [15], [16].

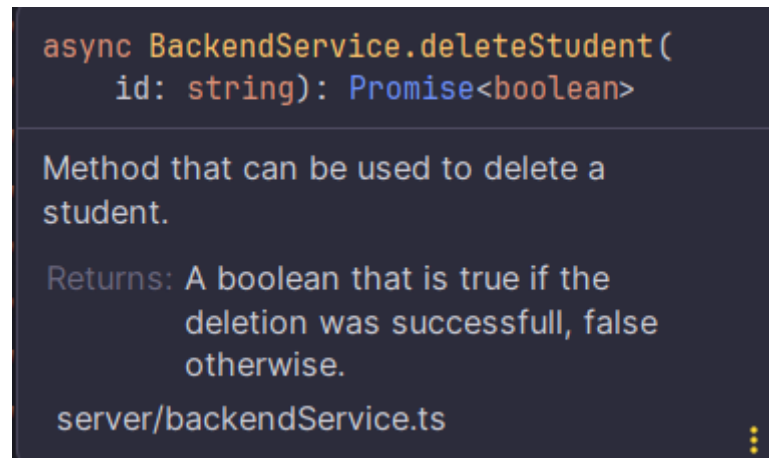


Fig 14: Descrierea metodei deleteStudent.

```
async deleteStudent(id: string): Promise<boolean> {  
  const student : number | null = await StudentModel.destroy({ options: {  
    where: {  
      id: id,  
    },  
  }).catch((error) : null => {  
    console.error(error);  
    return null;  
  });  
  
  return student != null;  
}
```

Fig 15: Codul sursă pentru metoda deleteStudent.

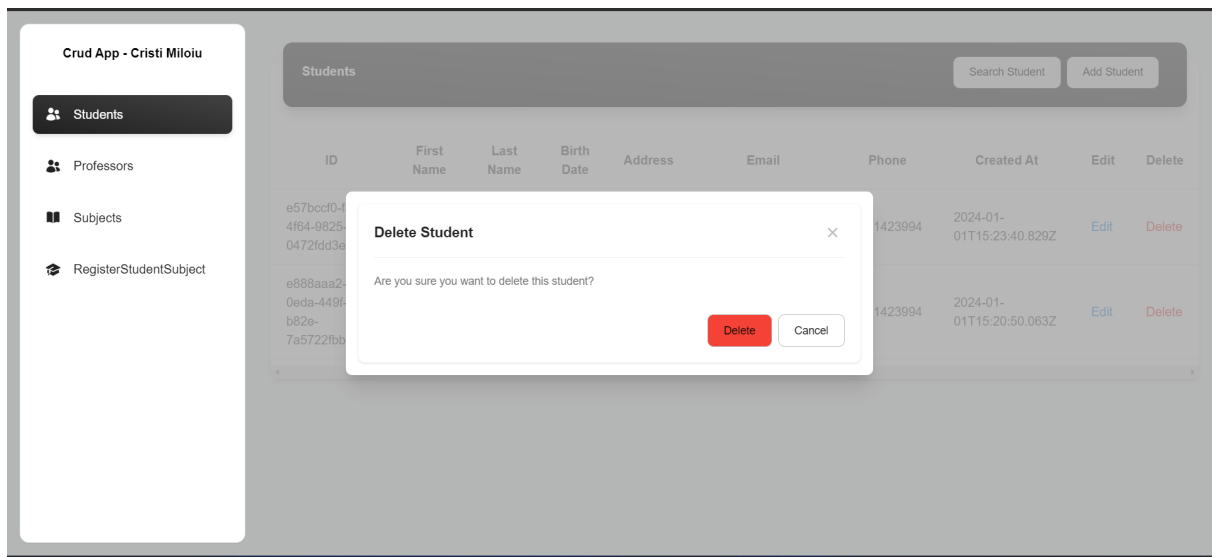


Fig 16: Interfata grafica a Modalul de ștergere a unui student.

#### Bibliografie:

- <https://www.material-tailwind.com/>
- <https://vitejs.dev/guide/>
- <https://docs.genez.io/genezio-documentation/>
- <https://heroicons.com/>
- <https://react.dev/reference/react>
- <https://neon.tech/>
- <https://docs.github.com/en>
- <https://chat.openai.com/>