

Miloiu Cristi  
Grupa 432A

## Documentatie Proiect Crud-App-Go

### Programarea Interfețelor pentru Baze de Date

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

#### 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.

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: Go, Gin, Gorm, Postgresql.

**Go** este un limbaj de programare compilat, dezvoltat de Google, care pune accent pe eficiență.

**Gin** este un framework web pentru Go.

**Gorm**, ca ORM pentru Go, adaugă un nivel de abstractizare pentru interacțiunea cu bazele de date, permițând dezvoltatorilor să lucreze cu datele folosind obiecte și metode, fără a fi nevoie să gestioneze manual interogările SQL complexe.

**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 acesteia fiind:

A. Tabela studenți are următoarele coloane [1]:

Name	students			Primary	id			
#	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].

Name	subjects	Primary		id				
#	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	public.professors(id)	→ 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	
#	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				
#	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	public.students(id)	→ NULL	
3	subjectId	uuid	YES	subjectId	NULL	public.subjects(id)	→ 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]:

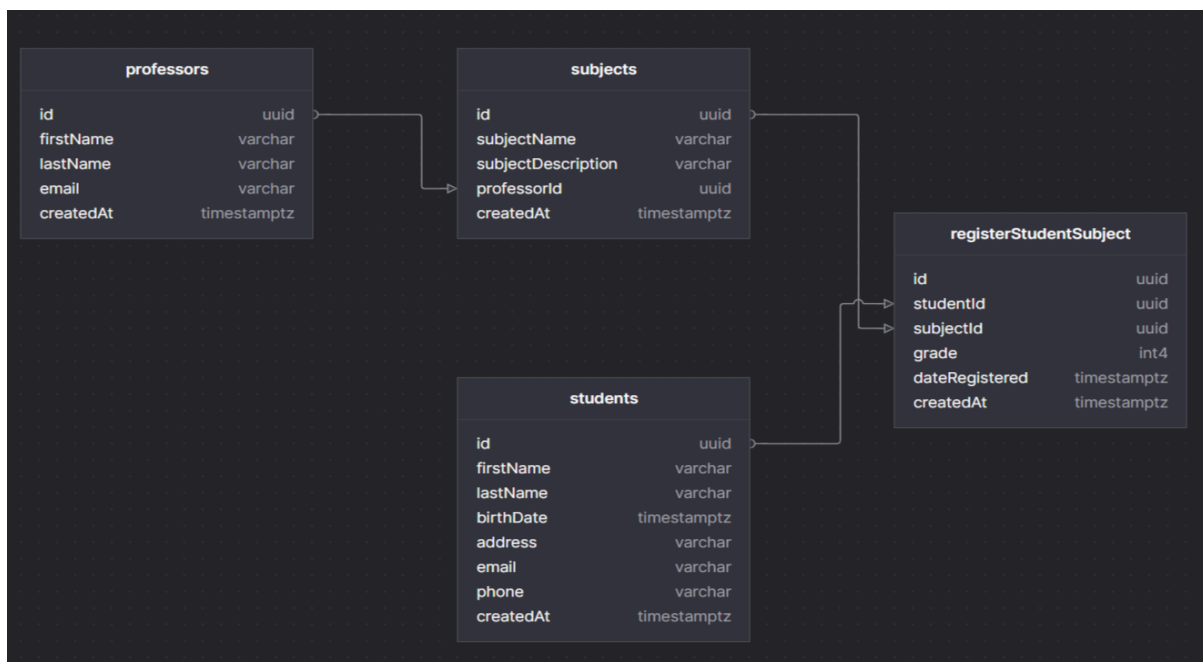


Fig 5: Diagrama bazei de date

Pe partea de backend aplicația are un middleware care așteaptă următoarele requesturi [6]:

```
router.POST( relativePath: "/students", createStudent)
router.GET( relativePath: "/students", getStudents)

router.DELETE( relativePath: "/students/:id", deleteStudent)
router.GET( relativePath: "/students/:id", getStudentByID)
router.PUT( relativePath: "/students/:id", updateStudent)

router.POST( relativePath: "/subjects", createSubject)
router.GET( relativePath: "/subjects", getSubjects)

router.DELETE( relativePath: "/subjects/:id", deleteSubject)
router.GET( relativePath: "/subjects/:id", getSubjectByID)
router.PUT( relativePath: "/subjects/:id", updateSubject)

router.POST( relativePath: "/professors", createProfessor)
router.GET( relativePath: "/professors", getProfessors)

router.DELETE( relativePath: "/professors/:id", deleteProfessor)
router.GET( relativePath: "/professors/:id", getProfessorByID)
router.PUT( relativePath: "/professors/:id", updateProfessor)

router.POST( relativePath: "/registerStudentSubjects", createRegisterStudentSubject)
router.GET( relativePath: "/registerStudentSubjects", getRegisterStudentSubject)

router.DELETE( relativePath: "/registerStudentSubjects/:id", deleteRegisterStudentSubject)
router.GET( relativePath: "/registerStudentSubjects/:id", getRegisterStudentSubjectByID)
router.PUT( relativePath: "/registerStudentSubjects/:id", updateRegisterStudentSubject)
```

Fig 6: Rutele middlewareului.

Functionalitati principale CRUD:

CRUD - Create, read, update, delete.

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

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

```
func createStudent(c *gin.Context) { 1 usage  cristim67
    var student Student
    if err := c.ShouldBindJSON(&student); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    student.CreatedAt = time.Now()

    if err := db.Create(&student).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusCreated, student)
}
```

Fig 7: Codul sursă al metodei createStudent.

The screenshot displays a web application interface. On the left, a sidebar titled 'Crud App - Cristi Miliu' contains navigation links: 'Students', 'Professors', 'Subjects', and 'RegisterStudentSubject'. The main area shows a modal window titled 'Add Student' with a close button. Inside the modal, a red error message 'Phone number is invalid' is visible. The form includes fields for 'First Name' (cristi), 'Last Name' (miliu), 'Birth Date' (05.01.2024), 'Address' (str ion mihalache), 'Email' (miliuc4@gmail.com), and 'Phone' (07912423). A '+ Add new student' button is at the bottom. In the background, a table of students is partially visible, showing columns for ID, Phone, Created At, Edit, and Delete.

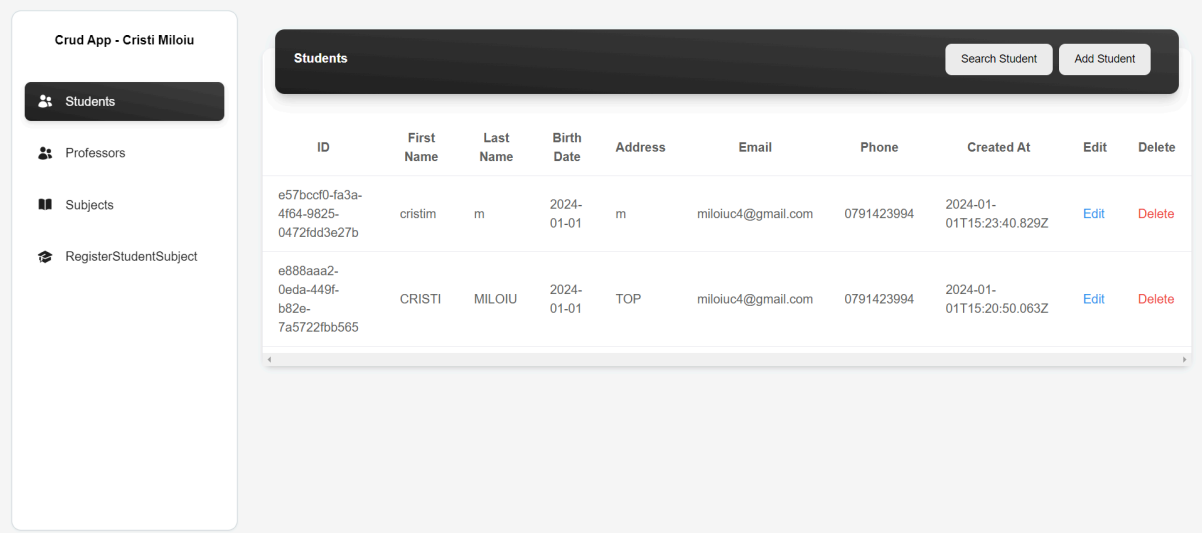
Fig 8: Interfata grafica a Modalului pentru adaugarea unui student

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

Exemplu - Citirea tuturor studentilor [9], [10].

```
func getStudents(c *gin.Context) { 1 usage  cristim67
    var students []Student
    if err := db.Find(&students).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }
    c.JSON(http.StatusOK, students)
}
```

Fig 9: Codul sursă pentru metoda getStudents.



ID	First Name	Last Name	Birth Date	Address	Email	Phone	Created At	Edit	Delete
e57bccf0-fa3a-4f64-9825-0472fdd3e27b	cristim	m	2024-01-01	m	miloiuc4@gmail.com	0791423994	2024-01-01T15:23:40.829Z	Edit	Delete
e888aaa2-0eda-449f-b82e-7a5722fbb565	CRISTI	MILOIU	2024-01-01	TOP	miloiuc4@gmail.com	0791423994	2024-01-01T15:20:50.063Z	Edit	Delete

Fig 10: Interfața grafică de afișare a studentilor.

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

Exemplu - Modificarea datelor unui student [11], [12].

```

func updateStudent(c *gin.Context) { 1 usage  cristim67
    id := c.Param(key: "id")

    var student Student
    if err := db.Where(query: "id = ?", id).First(&student).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    if err := c.ShouldBindJSON(&student); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }

    if err := db.Save(&student).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    c.Status(http.StatusNoContent)
}

```

Fig 11: Codul sursă pentru metoda updateStudent.

Crud App - Cristi Miloiu

- Students
- Professors
- Subjects
- RegisterStudentSubject

Students

Search Student Add Student

Phone Created At Edit Del

1423994 2024-01-01T15:23:40.829Z Edit Del

1423994 2024-01-01T15:20:50.063Z Edit Del

Phone number is invalid

First Name Last Name

cristim m

Birth Date Address

01.01.2024 m

Email Phone

miloic4@gmail.com 07914239922

SAVE CHANGES CANCEL

Fig 12: Interfața grafică pentru Modalul care se ocupă cu actualizarea unui student

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

Exemplu - Ștergerea unui student [13], [14].

```
func deleteStudent(c *gin.Context) { 1 usage  cristim67
    id := c.Param(key: "id")

    if err := db.Where(query: "id = ?", id).Delete(&Student{}).Error; err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    c.Status(http.StatusNoContent)
}
```

Fig 13: Codul sursă pentru metoda deleteStudent.

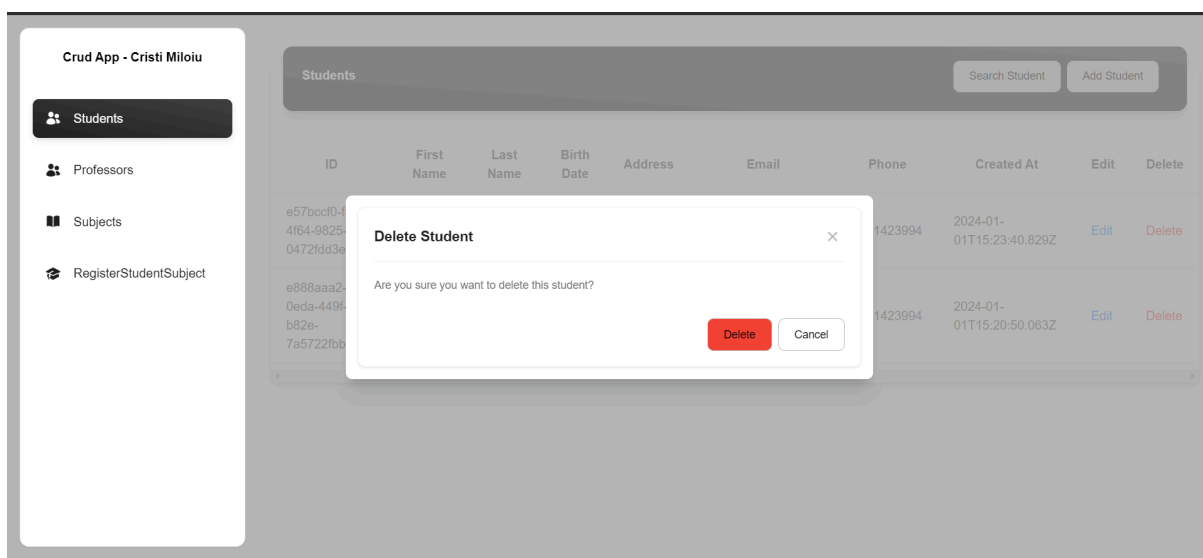


Fig 14: Interfața grafică a Modalul de ștergere a unui student.

#### Bibliografie:

- <https://www.material-tailwind.com/>
- <https://vitejs.dev/guide/>
- <https://heroicons.com/>
- <https://react.dev/reference/react>
- <https://neon.tech/>
- <https://docs.github.com/en>
- <https://chat.openai.com/>
- <https://go.dev/doc/>
- <https://gin-gonic.com/docs/>
- <https://gorm.io/docs/>