UNIVERSITATEA TEHNICA DIN CLUJ-NAPOCA

# **Introducere**

## 1.1 Descrierea Generală a Proiectului

Aplicația este un sistem informatic destinat gestiunii unei platforme de studiu. Functionalitatile pe care le oferă aplicația vizează operații ce țin de gestiunea studentilor, profesorilor și administrarea operațiilor curente din cadrul unor programe de studiu.

## 1.2 Scopul și Obiectivele

Crearea unui cadru online prin care se pot organiza activități didactice și interacțiuni între studenți și profesori.

# **Tema Proiectului**

## 2.1 Prezentarea Aplicației

Aplicația propusă reprezintă un sistem informatic destinat gestionării unei platforme de studiu, care facilitează managementul studenților, profesorilor și activităților academice din cadrul unui program de învățământ. Realizată pe baza unei baze de date MySQL și având o interfață grafică intuitivă, aplicația permite realizarea eficientă a operațiunilor specifice rolurilor definite.

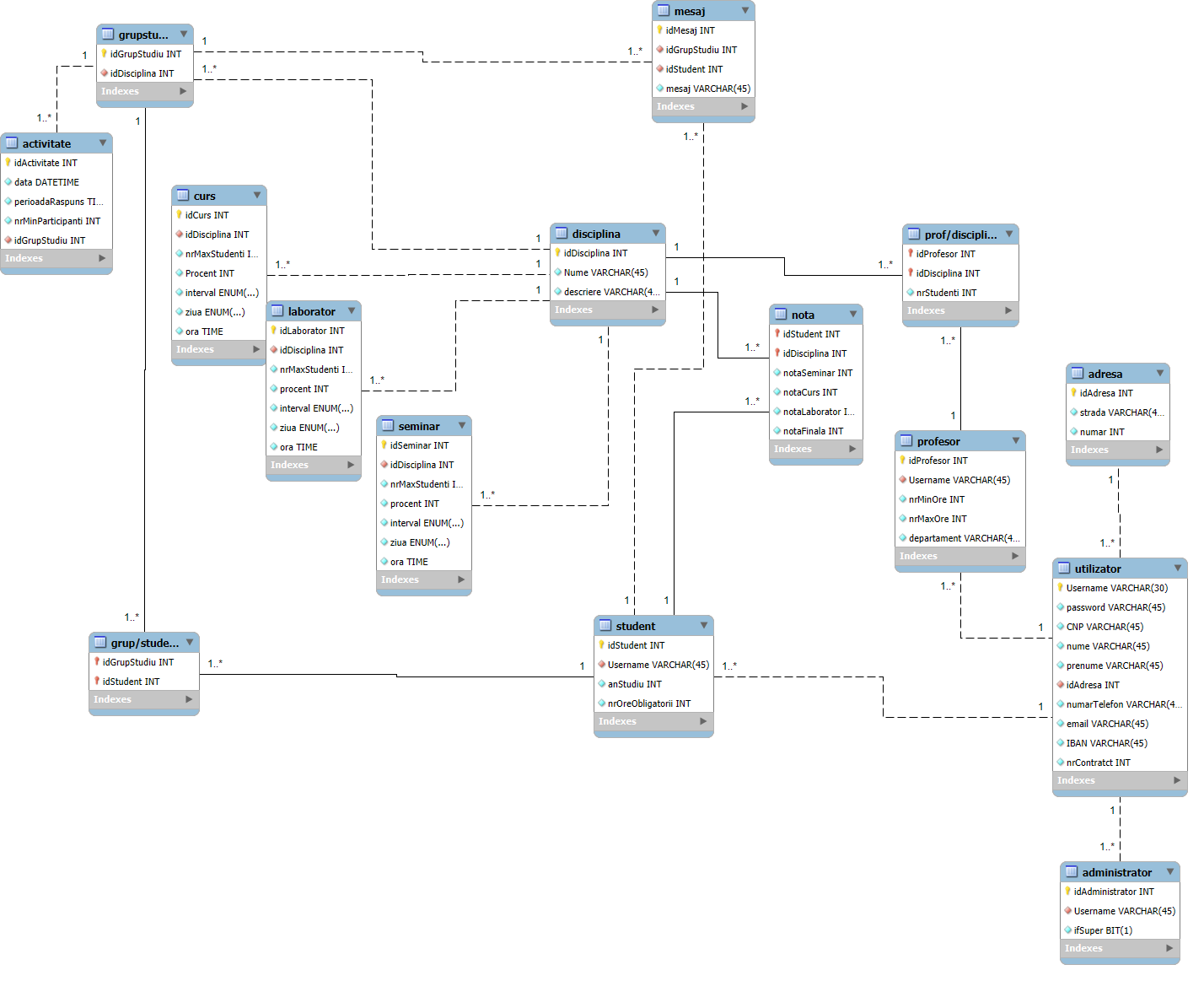
## 2.2 Funcționalități și Utilizatori

Aplicația oferă suport pentru trei tipuri principale de utilizatori: **student**, **profesor** și **administrator**. Printr-un sistem de autentificare și autorizare, fiecare utilizator își poate accesa datele personale precum CNP, nume, adresă, număr de telefon, email sau informații suplimentare precum cont IBAN sau contractul, fără a le putea modifica. Administratorii pot gestiona aceste date și pot atribui cursuri profesorilor și studenților, oferind vizibilitate asupra relațiilor și interacțiunilor din cadrul sistemului.

Pentru profesorii înregistrați, aplicația permite gestionarea cursurilor și activităților asociate (cursuri, seminare, laboratoare), inclusiv configurarea programării acestora, numărul maxim de studenți acceptați și tipurile de activități derulate. Studenții pot vizualiza și accesa catalogul, unde sunt evidențiate rezultatele lor și le pot descărca.

# **Descrierea Bazei de Date**

## 3.1 Structura generală



Modelul bazei de date prezentat este unul relațional, fiind organizat în tabele interconectate care descriu clasele și relațiile din aplicație. Structura generală este bine definită, acoperind clase majore precum utilizatorii (studenti, profesori, administratori), cursurile, activitățile academice (seminare, laboratoare, cursuri) și informațiile administrative (note, mesaje, adrese). Relațiile dintre tabele sunt gestionate prin chei primare și chei străine pentru a menține structura datelor.

## 3.2 Detalii specifice

**Utilizatori**:

* Tabelul “utilizator” stochează informații personale (CNP, nume, adresă, email, IBAN) și identifică rolul fiecărei persoane.
* Profesorii și studenții sunt specificați în tabele separate (profesor, student) care conțin specificatii extra, cum ar fi departamentul pentru profesori sau anul de studiu pentru studenți.

**Cursuri și activități**:

* “Curs”, “seminar” și “laborator” sunt tabele distincte care detaliază tipurile de activități academice. Fiecare tabel conține informații despre disciplina asociată, numărul maxim de studenți, zilele și orele în care se desfășoară.
* Tabelul “activitate” conține informațiile despre toate activitățile programate, incluzând data și perioada de răspuns.

**Evaluare și note**:

* Tabelul “nota” stochează rezultatele studenților pentru fiecare tip de activitate, inclusiv nota finală calculată pe baza ponderilor definite printr-o procedura MySQL.

**Relații și comunicare**:

* Relația dintre studenți și grupurile lor este gestionată în tabelul “grup/student”.
* Tabelul “mesaj” permite comunicarea între utilizatori, legată de cursuri sau alte activități.

3.3 Nivelul de normalizare

Modelul pare să fie cel puțin la **forma normală a treia (3NF)**:

* Fiecare atribut este atomic, ceea ce indică prima formă normală (1NF).
* Toate atributele depind complet de cheia primară din fiecare tabel, evitând dependențele parțiale (2NF).
* Nu există dependențe tranzitive între atributele non-cheie, ceea ce respectă cerințele 3NF.

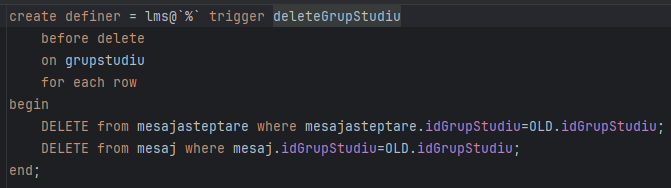
Prin această normalizare, modelul minimizează redundanța datelor și asigură o integritate mai bună a bazei de date, făcând structura flexibilă și ușor de întreținut.

# **Elemente de programare a functionalitatilor**

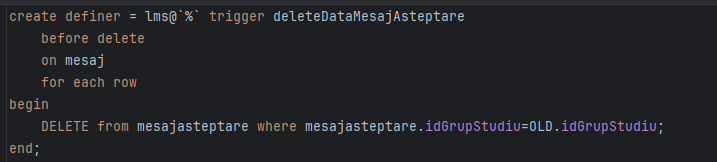
## 4.1 Proceduri și Triggere.Lista completa si exemple

**Triggere:**

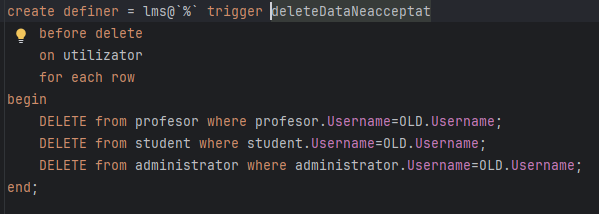
In tabela “grupstudiu” avem trigger-ul deleteGrupStudiu:



În tabela “mesaj” avem trigger-ul “deleteDataMesajAsteptare”:

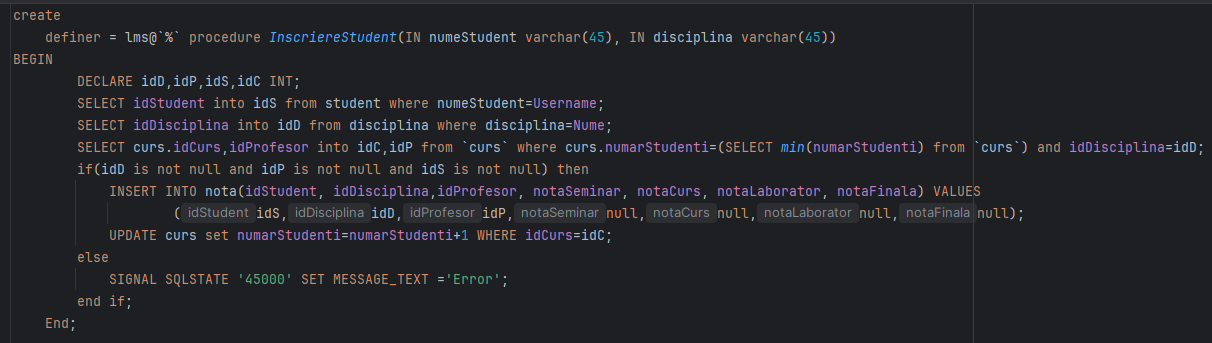


În tabela “Utilizator” avem “deleteDataNeacceptat”:

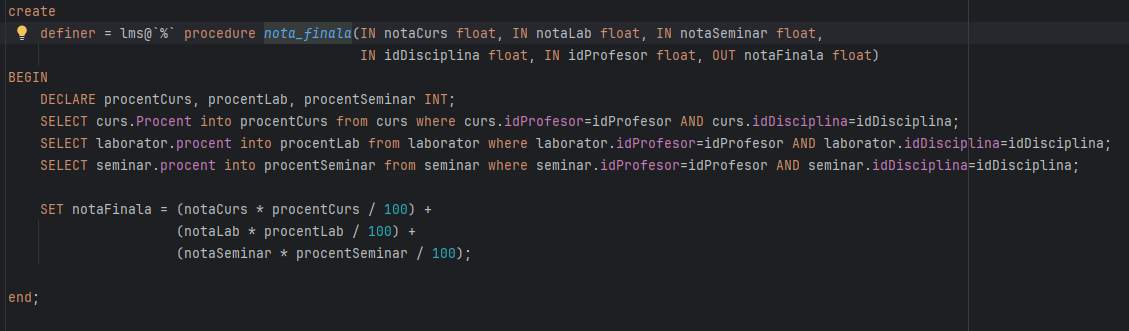


**Proceduri:**

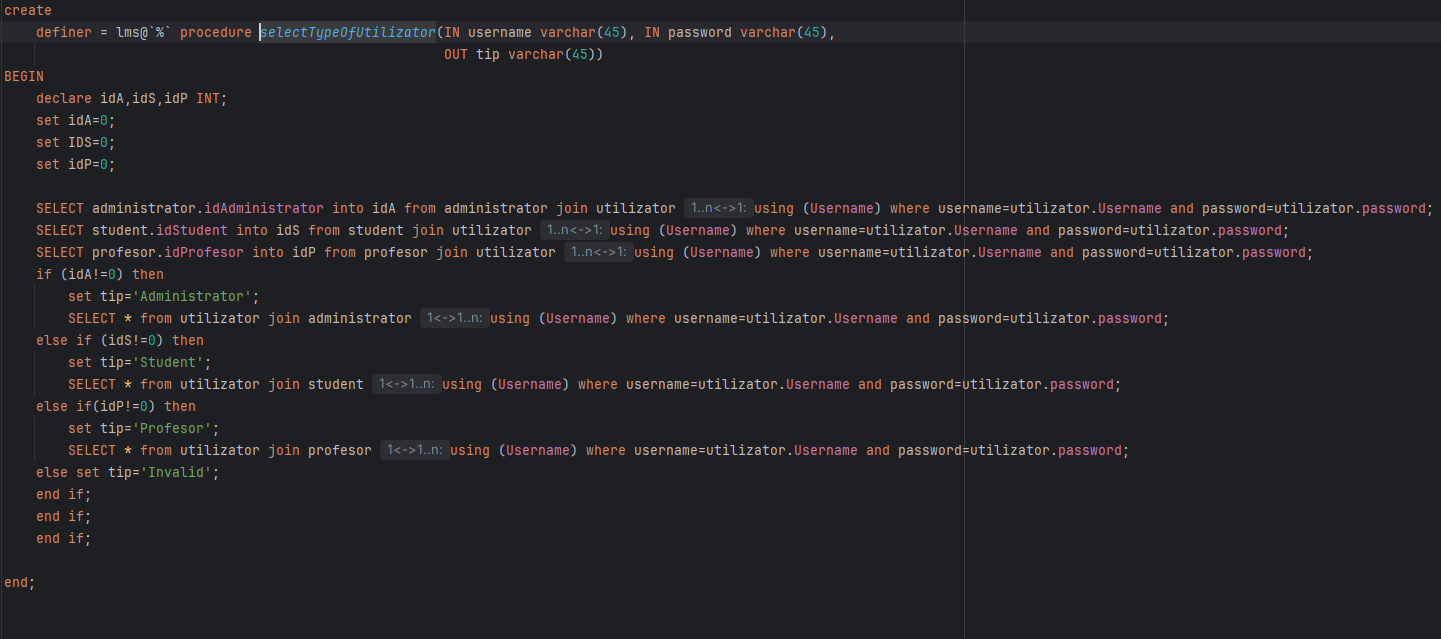
Procedura “inscriereStudent” care înscrie studentul la o anumita disciplina:



Procedura de calculare a notei finale a fiecărui student pe baza notelor de la curs, seminar si laborator (aceasta mai e bazata si pe profesorul de care apartine studentul):



Procedura care selectează tipul de utilizator.



## 4.2 Alte exemple de Cod SQL

Codul SQL pentru selectarea datelor unui utilizator:

String url = "jdbc:mysql://139.144.67.202:3306/lms?user=lms&password=WHlQjrrRDs5t";

Connection conn = DriverManager.*getConnection*(url);

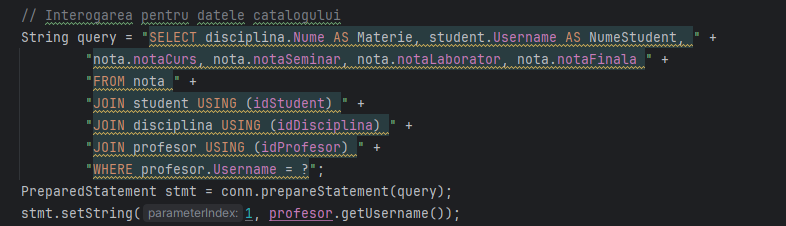
PreparedStatement stmt = conn.prepareStatement(

"SELECT u.\*, s.anStudiu, p.departament, a.strada, a.numar FROM utilizator u LEFT JOIN student s ON u.Username = s.Username LEFT JOIN profesor p ON u.Username = p.Username LEFT JOIN adresa a ON u.idAdresa = a.idAdresa WHERE u.Username = ?");

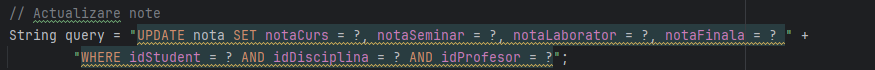
stmt.setString(1, usernameText);

ResultSet rs = stmt.executeQuery();

Codul SQL pentru datele necesare din catalogul școlar:



Updatare de note:



Codul SQL pentru selectarea grupurilor din care utilizatorul face parte:

String url="jdbc:mysql://139.144.67.202:3306/lms?user=lms&password=WHlQjrrRDs5t";

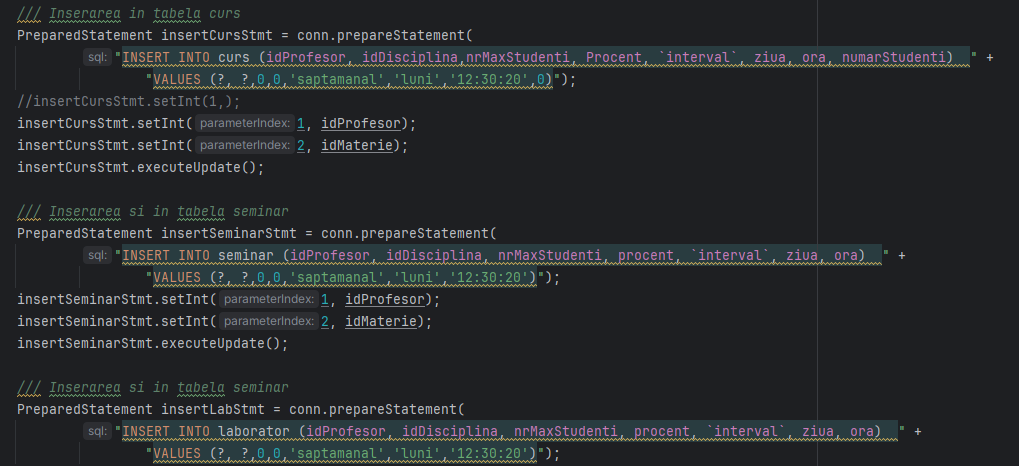
Connection conn= DriverManager.*getConnection*(url);

PreparedStatement stmt=conn.prepareStatement("SELECT distinct numeGrup,idGrupStudiu from grupstudiu join mesaj using (idGrupStudiu) where mesaj.idStudent=(select idStudent from student where Username=?) and mesaj='';");

stmt.setString(1,student.getUsername());

ResultSet rs=stmt.executeQuery();

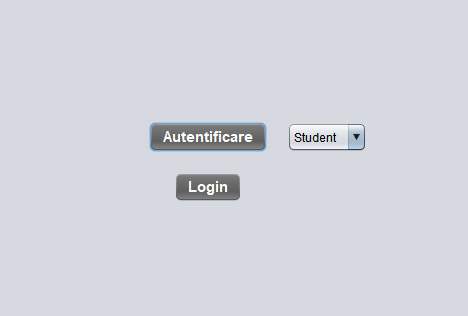
Inserare de valori in tabele:



# **Interfața Grafică a Utilizatorului (GUI)**

## 5.1 Descrierea functionalitatilor per tip de utilizator

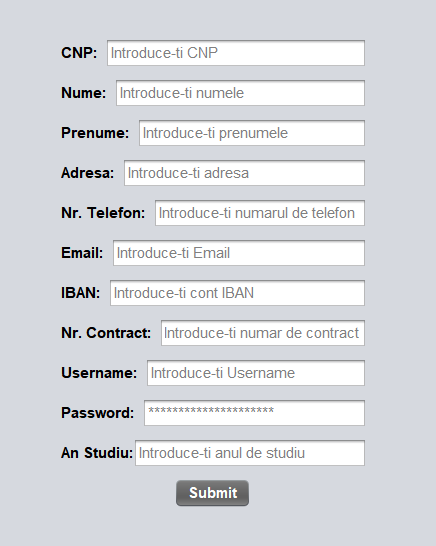
### **Pagina Inițiala:**



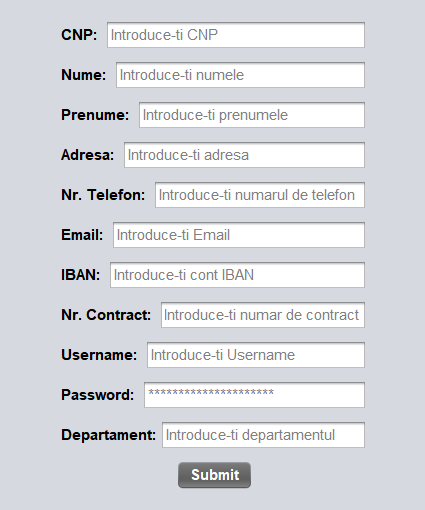
Conține butonul pentru autentificare și butonul pentru conectare. Lângă butonul de logare se alege tipul de cont cu care dorești sa te autentifici.

### **Pagina de Autentificare:**

Autentificare ca student:

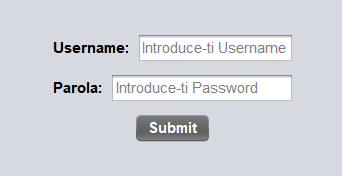


Autentificare ca profesor:



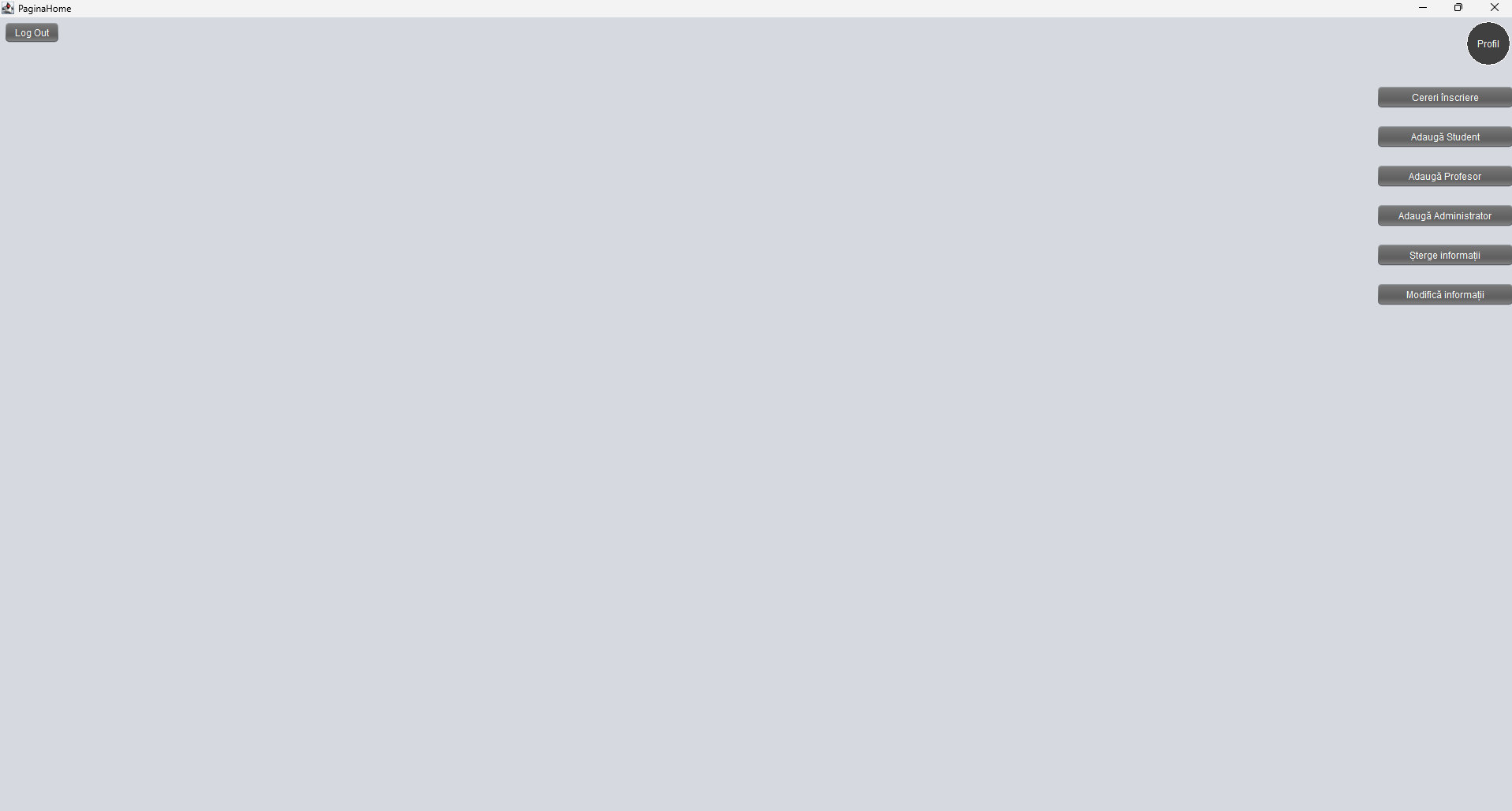
Câmpurile trebuie completate cu date valide specifice pentru fiecare tip de utilizator.

### **Pagina de conectare:**

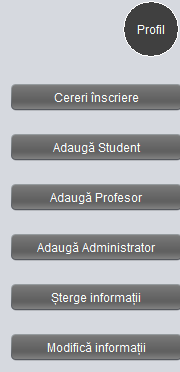


Utilizatorul introduce Username-ul și parola iar dacă acestea sunt valide se va deschide Pagina principală pentru tipul de utilizator corespunzător contului respectiv.

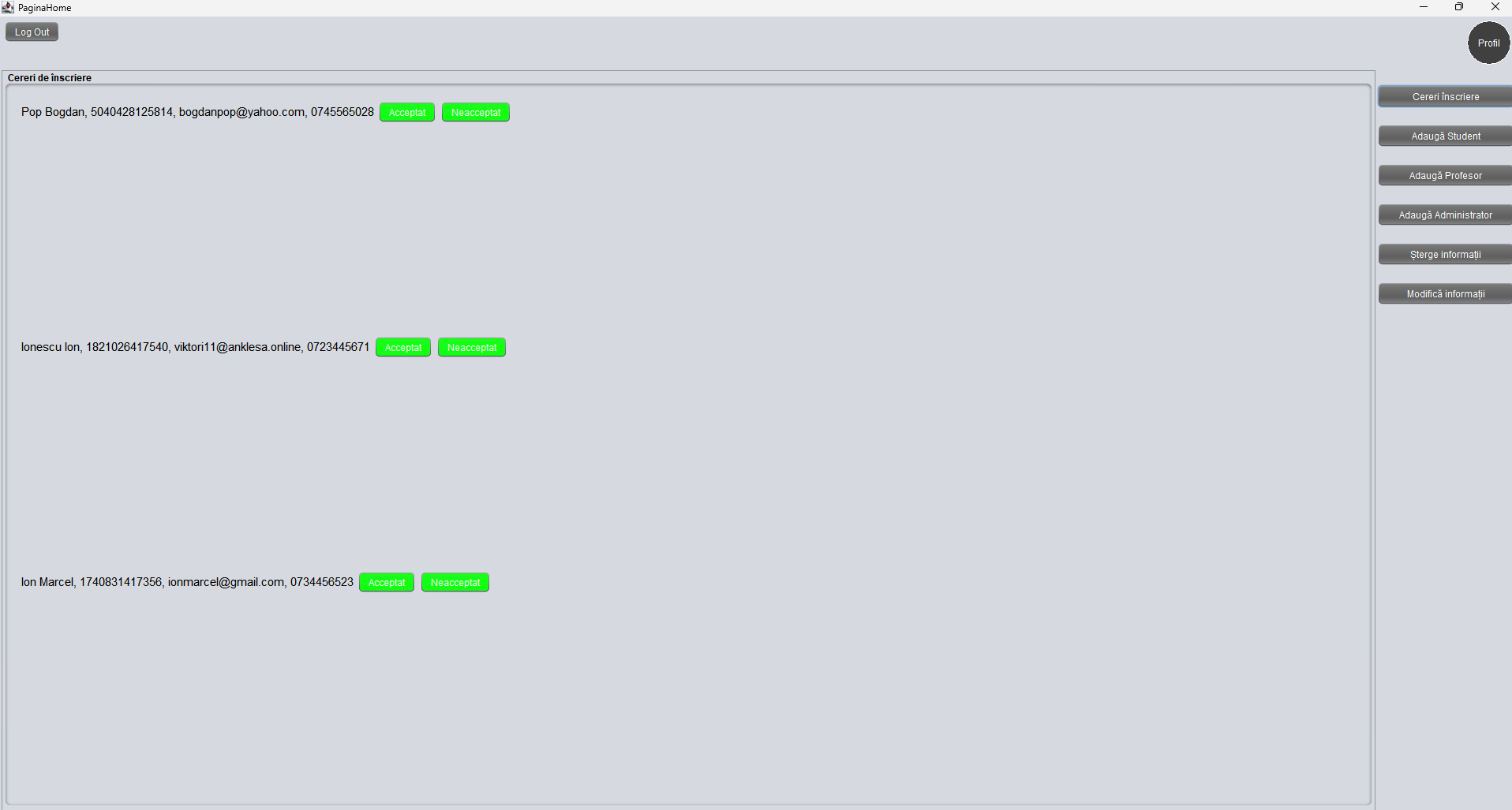
### **Pagina Administratorului:**



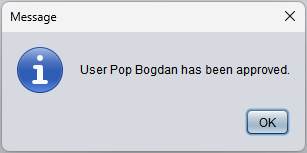
În partea dreapta sus se află butonul de proful unde se pot vizualiza datele utilizatorului conectat. Iar sub acel buton sunt 5 butoane pentru administratori, respectiv 6 pentru superAdministratori.



**Butonul “Cerere înscriere”** afișează lista utilizatorilor care s-au autentificat și așteaptă aprobare din partea administratorilor.

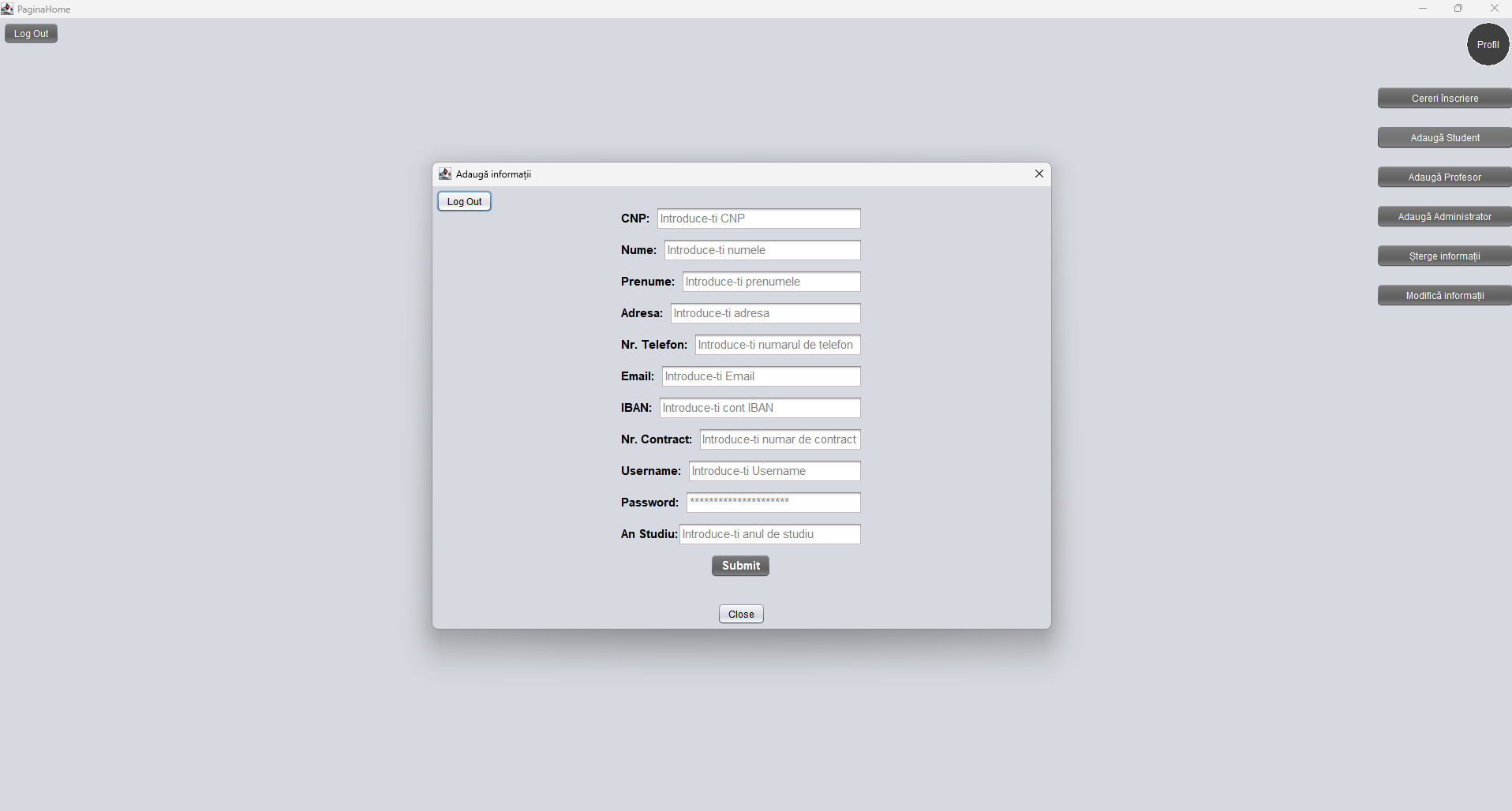


Dacă utilizatorul este acceptat se marchează și rămâne în baza de date

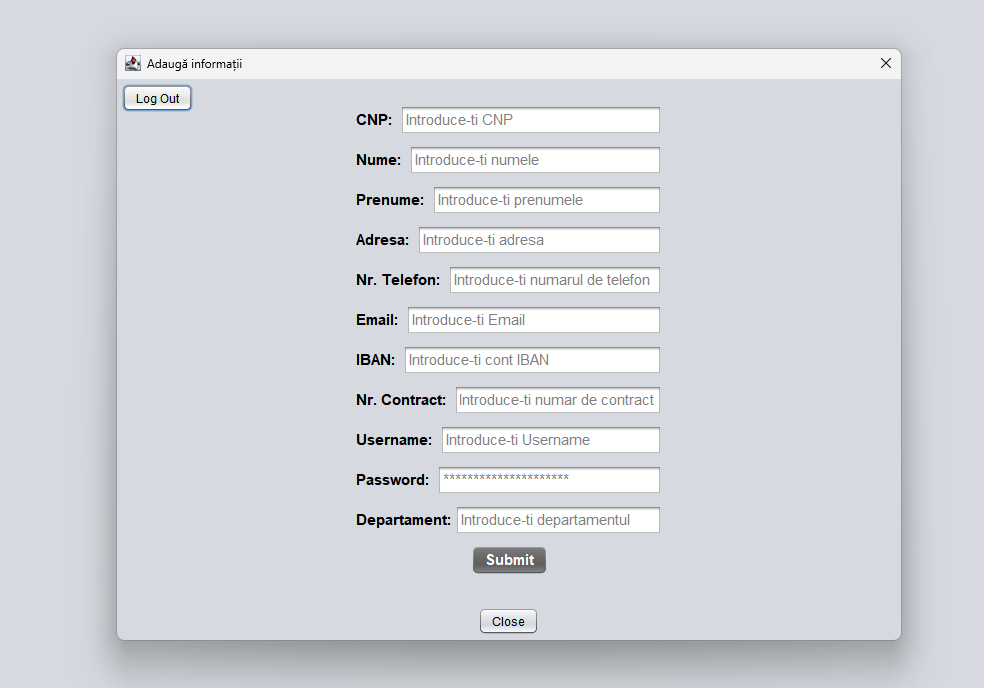


Altfel utilizatorul este șters din baza de date.

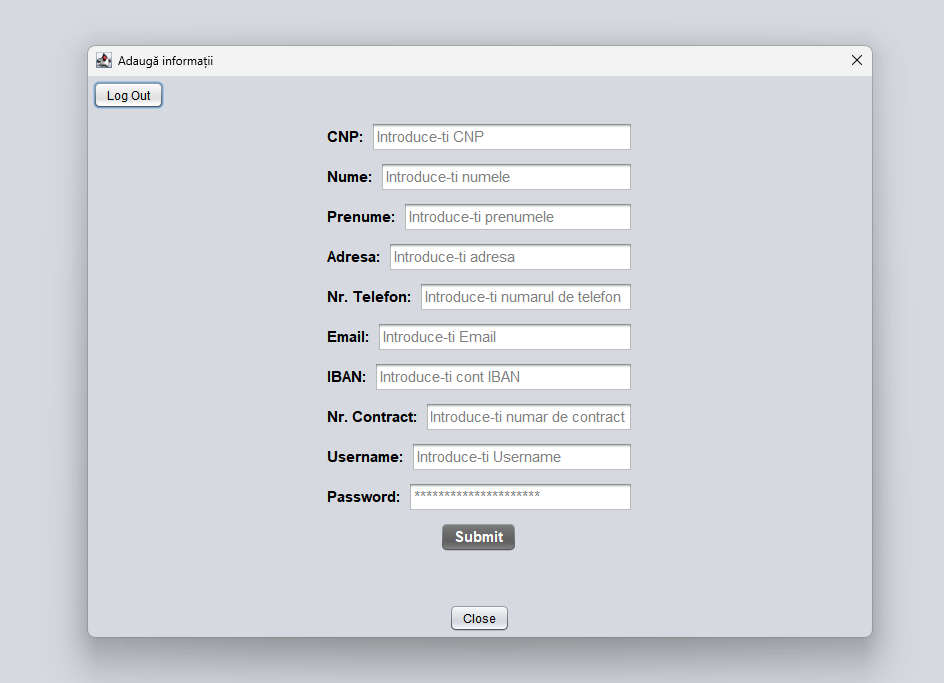
**Butonul “Adaugă Student”** deschide o fereastră ca cea de autentificare în care administratorul poate introduce date pentru a crea un cont pentru un utilizator de tip student.



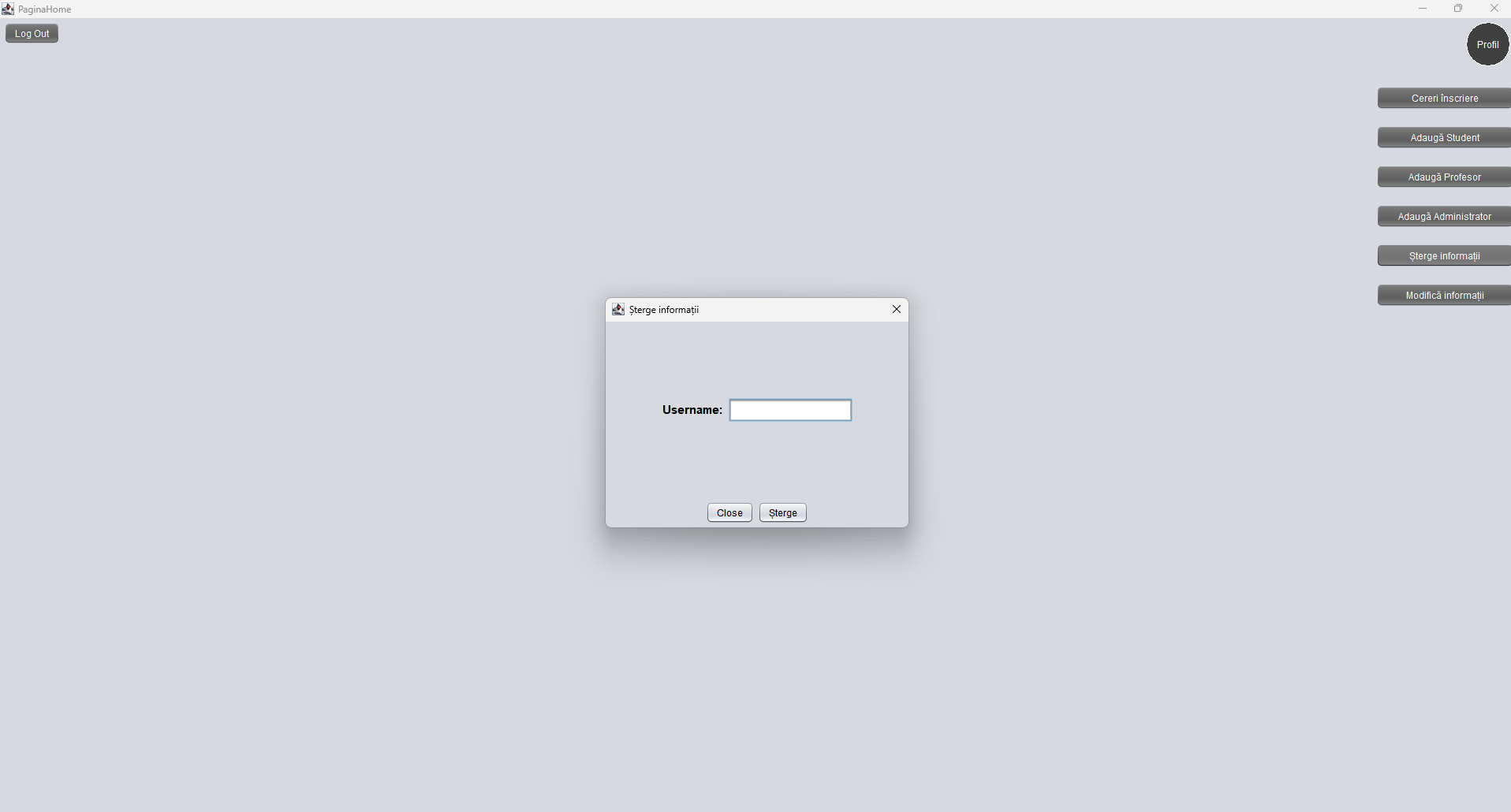
La fel și pentru **Butonul “Adaugă Profesor”**, se va deschide pagina de autentificare pentru tipul profesor.



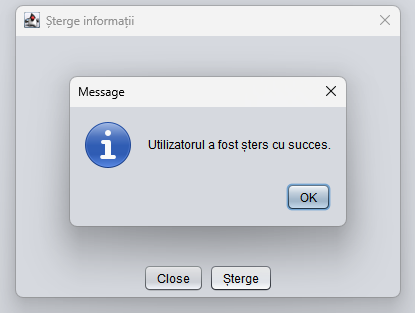
**Butonul “Adaugă Administrator”** va fi vizibil doar pentru Super Administratori. Acestea doar au acces la datele altor administratori și pot sa adauge sau sa modifice date.



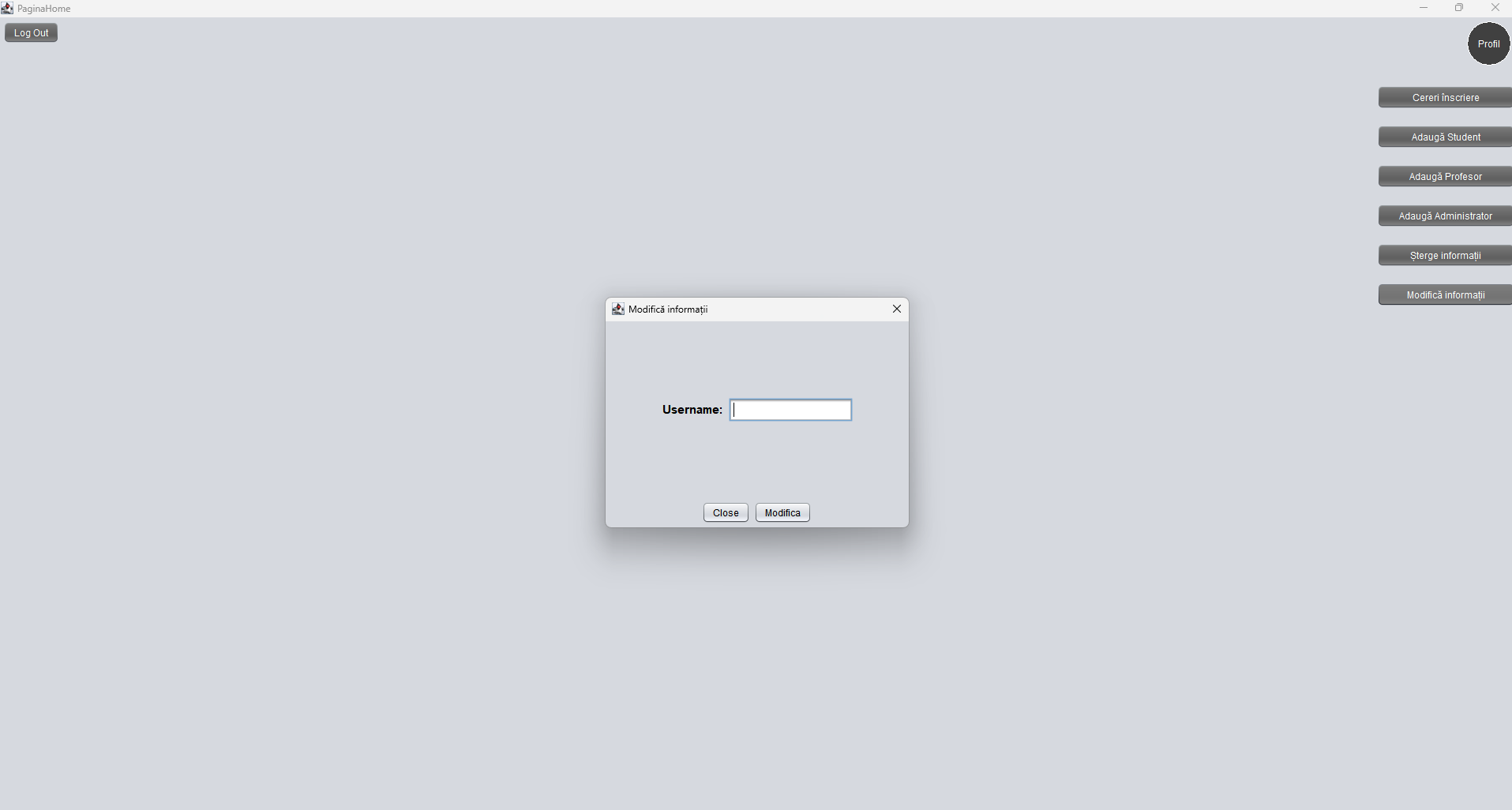
**Butonul “Șterge Informații”** va deschide o fereastră unde administratorul poate introduce username-ul unui utilizator la datele căruia are acces pe care dorește să-l șteargă din baza de date.



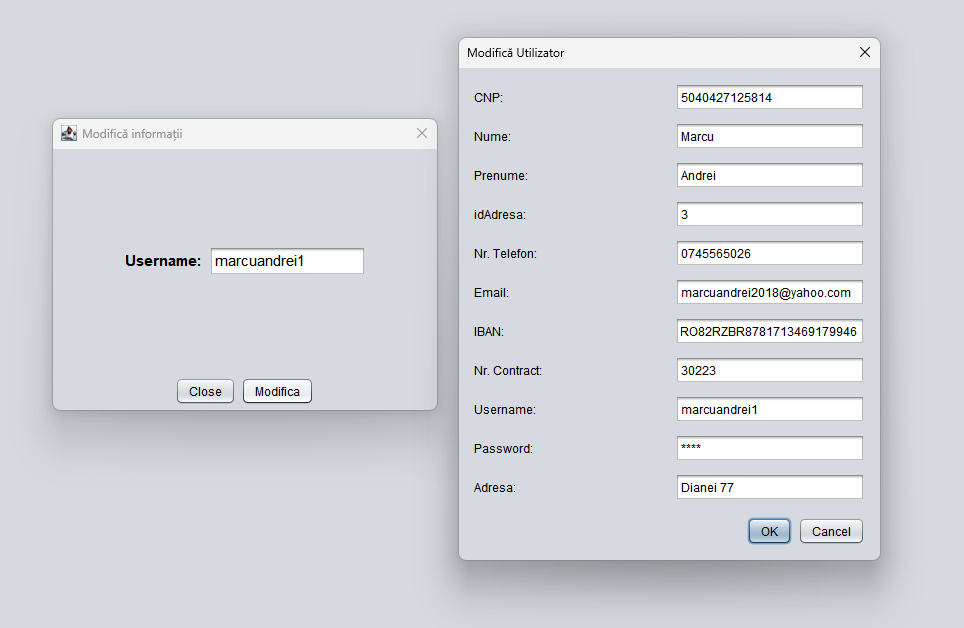
După apăsarea butonului “Șterge” se va primi o confirmare dacă ștergerea a fost făcută cu succes.



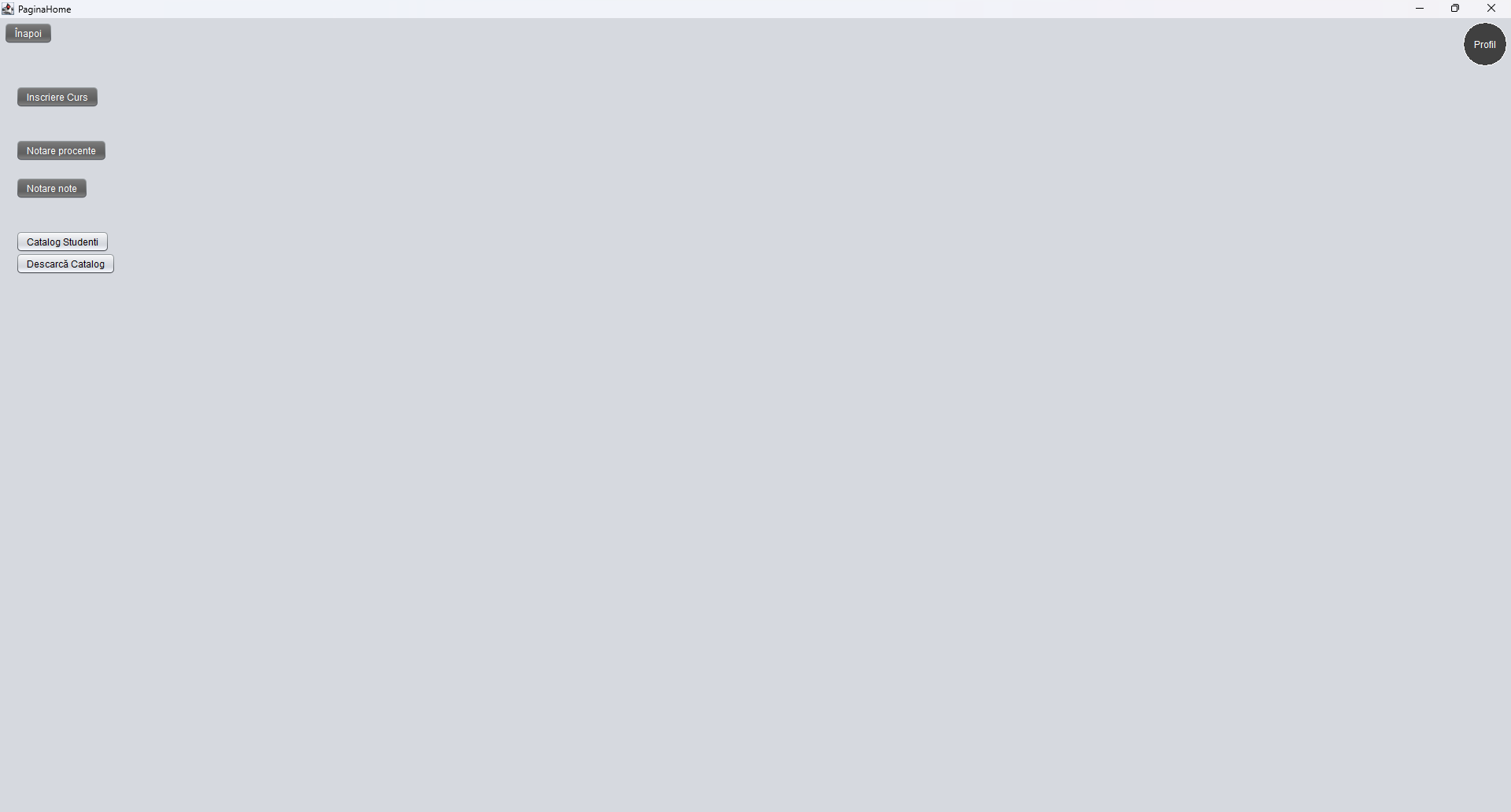
**Butonul “Modifică Informații”** va deschide o fereastră unde administratorul poate introduce username-ul unui utilizator la datele căruia are acces.



Dacă acesta este valid se va deschide o nouă fereastra în care va vedea datele utilizatorului, care vor putea fi modificate și apoi salvate.



### **Pagina Profesorului:**



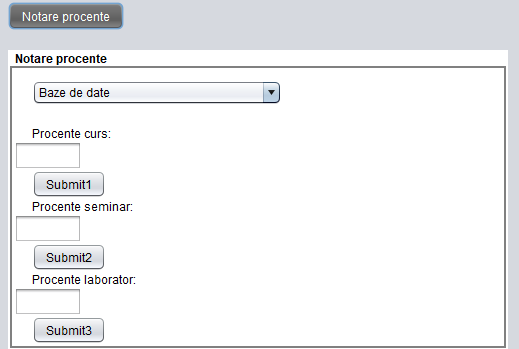
Pentru **butonul de înscriere curs**:

Fiecare profesor se poate înscrie la una din materiile din baza de date.



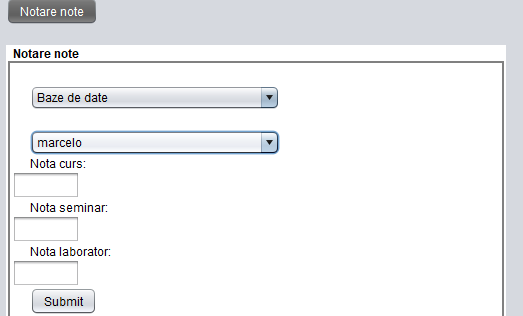
**Butonul de notare procente**:

Fiecare profesor poate să își selecteze pe rând una din materiile la care preda din ComboBox-ul de mai jos și își poate introduce, in functie de curs, seminar si laborator, procentul dorit pentru a calcula notele studenților.



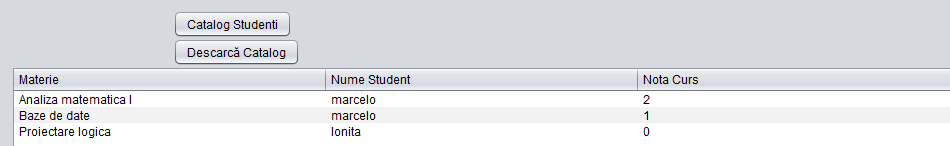
**Butonul de notare note:**

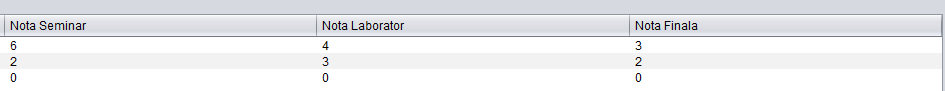
Profesorul trebuie sa selecteze prima data una din materiile predate, dupaia poate selecta unul din studenții pe care îi are la materia respectivă și poate dupaia sa introduca notele pentru curs, seminar si laborator. În urma butonului de submit vor apărea în catalog notele la fiecare disciplina pentru fiecare student.



**Pentru catalog:**

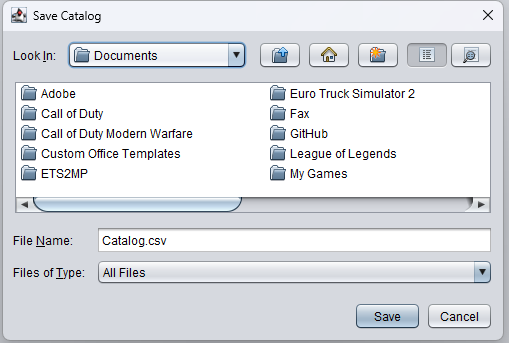
Fiecare student a profesorului în funcție de materie apare in catalog. Acești studenți au în dreptul lor notele de la curs, seminar si laborator si nota finala la materia resprectiva.

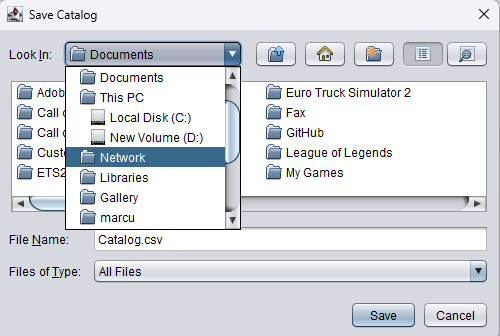




La apăsarea butonului “Descarca catalog” se va descarca un document de tip .csv si va putea fi downloadat și deschis in excel sau alt document.

**Salvarea documentului:**





### **Pagina Studentului:**

## 5.2 Detalii de implementare funcționalități specifice

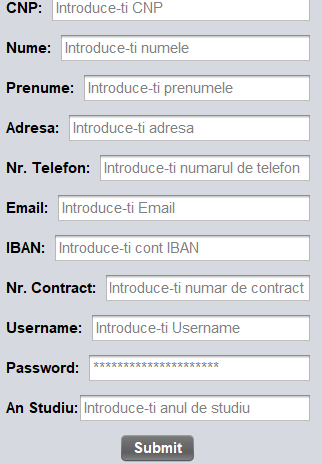
Pentru implementarea interfeței grafice am folosit libraria java swing care ne-a pus la dispoziție toate componentele și clasele necesare pentru crearea unei interfețe simple și ușor de înțeles pentru utilizatori.

# **Manual de Utilizare**

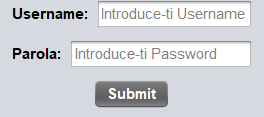
## 6.1 Înregistrare și Autentificare

Pe pagina inițială a aplicației se afișează 2 opțiuni: “Autentificare” si “Conectare”. În dreptul butonului de autentificare se poate selecta opțiunea tipului de cont care se poate crea.

Odata apasat pe butonul de “autentificare” va aparea o fereastra cu campurile necesare de completat:



Datele introduse trebuie sa fie valide în funcție de câmp pentru a putea crea contul.

Iar pentru butonul de logare vor apărea câmpurile:

Trebuie introduse datele unui cont deja creat: un “username” valid și parola respectivă contului.

6.2 Gestionarea Informațiilor per utilizatori

Pentru fiecare fel de cont (student, profesor, admin) exista butonul de “Profil” care afișează informații specifice fiecărui utilizator.

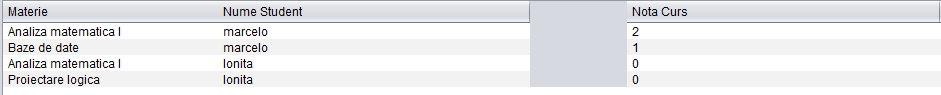


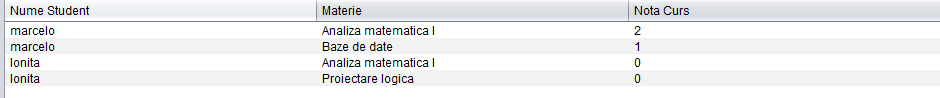
Informațiile sunt stocate în baza de date în mai multe tabele. Toate datele comune tuturor utilizatorilor sunt puse într-un tabel numit “Utilizator”. Adresele sunt puse într-un alt tabel care este legat de tabelul utilizator prin indexul adresei. Iar datele specifice pentru un rol al utilizatorului sunt puse în 3 tabele separate: profesor, student, administrator, care sunt legate de tabela utilizator prin username.

# **Concluzii și dezvoltării ulterioare**

## 7.1 Analiza funcționalităților curente

**Catalog:**

Pentru catalogul studentilor se pot modifica ordinea coloanelor cu mouse-ul: 



Prin “drag and drop” se pot muta coloanele între ele.

**Autentificare:**

În pagina de autentificarea, câmpurile trebuie completate cu date valide, altfel contul nu va putea fi creat iar utilizatorul va primi un feedback:



Exemplu (funcția de validare IBAN):

public static void verificareIban(String s, JTextField text) throws IOException {

if (s.equals("Introduce-ti cont IBAN") || s.length() < 15 || s.length() > 34) {

Color lightRed = new Color(255, 102, 102);

text.setBackground(lightRed);

}

s = s.replaceAll("\\s+", "").toUpperCase();

String countryCode = s.substring(0, 2);

Integer expectedLength = 24;//pt RO

if (expectedLength == null || s.length() != expectedLength) {

Color lightRed = new Color(255, 102, 102);

text.setBackground(lightRed);

throw new IOException("IBAN invalid!");

}

String rearranged = s.substring(4) + s.substring(0, 4);

StringBuilder numericIBAN = new StringBuilder();

for (char c : rearranged.toCharArray()) {

if (Character.*isLetter*(c)) {

numericIBAN.append(c - 'A' + 10);

} else {

numericIBAN.append(c);

}

}

BigInteger ibanNumber = new BigInteger(numericIBAN.toString());

if (ibanNumber.mod(BigInteger.*valueOf*(97)).intValue() != 1){

Color lightRed = new Color(255, 102, 102);

text.setBackground(lightRed);

throw new IOException("IBAN invalid!");

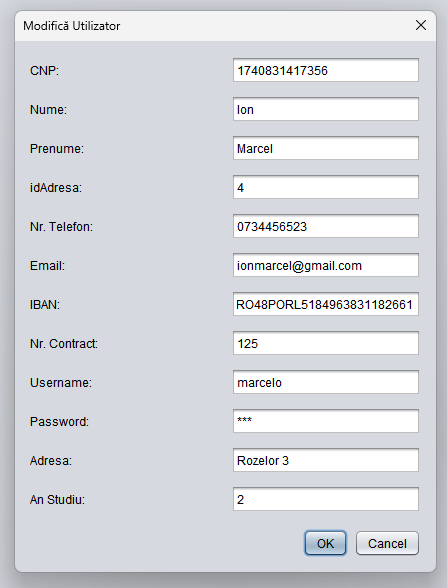
}

text.setBackground(Color.*WHITE*);

}

**Modificare Date:**

Când un administrator dorește sa modifice datele unui utilizator se deschide fereastra “Modifica Utilizator” care are un formular ca cel de la autentificare, dar care are datele utilizatorului completate. Datele pot fi modificate iar dacă sunt valide sunt introduse în baza de date în locul celor vechi.



**Algoritmul de programare a orarului:**

Se folosește algoritmul “Graph coloring” pentru a putea organiza materiile pentru studenți astfel incat sa nu se suprapună atât cursurile și activitățile, cat și orele de la care sunt programate în funcție de studenți și profesori.



try{

String url="jdbc:mysql://139.144.67.202:3306/lms?user=lms&password=WHlQjrrRDs5t";

Connection conn= DriverManager.*getConnection*(url);

PreparedStatement stmt=conn.prepareStatement("SELECT idCurs as ID,idProfesor,Nume,'Curs'as type from curs join disciplina using(idDisciplina)UNION ALL\n" +

"SELECT idSeminar,idProfesor,Nume,'Seminar'as type from seminar join disciplina using(idDisciplina) UNION ALL\n" +

"SELECT idLaborator,idProfesor,Nume,'Laborator'as type from laborator join disciplina using(idDisciplina);");

ResultSet rs=stmt.executeQuery();

List<GraphNode> nodes=new ArrayList<>();

List<Integer> idProfesor=new ArrayList<>();

while(rs.next()){

GraphNode(0,rs.getInt(1),index,rs.getString(4),rs.getString(3)));

idProfesor.add(rs.getInt(2));

index++;

}

for (int i = 0; i < idProfesor.size(); i++) {

for (int j = i + 1; j < idProfesor.size(); j++) {

// If both nodes have the same idProfesor, add an edge

if (idProfesor.get(i).equals(idProfesor.get(j))) {

graph.addEdge(nodes.get(i), nodes.get(j)); }

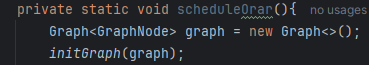
}

}

tmt=conn.prepareStatement("SELECT GROUP\_CONCAT(nota.idStudent ORDER BY nota.idStudent ASC SEPARATOR ',') AS student\_ids FROM curs JOIN disciplina USING(idDisciplina) JOIN nota ON nota.idDisciplina = curs.idDisciplina AND nota.idProfesor = curs.idProfesor GROUP BY curs.idCurs UNION ALL\n" +

"SELECT GROUP\_CONCAT(nota.idStudent ORDER BY nota.idStudent ASC SEPARATOR ',') AS student\_ids FROM seminar JOIN disciplina USING(idDisciplina) JOIN nota ON nota.idDisciplina = seminar.idDisciplina AND nota.idProfesor = seminar.idProfesor GROUP BY seminar.idSeminar UNION ALL\n" +

"SELECT GROUP\_CONCAT(nota.idStudent ORDER BY nota.idStudent ASC SEPARATOR ',') AS student\_ids FROM laborator JOIN disciplina USING(idDisciplina) JOIN nota ON nota.idDisciplina = laborator.idDisciplina AND nota.idProfesor = laborator.idProfesor GROUP BY laborator.idLaborator;");





## 7.2 Dezvoltări ulterioare

* Adaugare imagini pe grupurile de chat
* Adaugare poze (poze de profil, de aplicatie, de grupuri)
* Creare video-chat-uri
* Adaugarea de rapoarte pentru studenți
* Îmbunătățirea securității și a validării datelor
* Crearea pentru aplicații mobile