



*Facultatea de Automatică și Calculatoare*

*Departamentul Calculatoare*

## **PROIECT**

la disciplina

**BAZE DE DATE**

**Titlu**

**Gestiunea unei platforme de studiu**

**Popovici Eusebiu Ionuț**

**Grădinariu Robert-Iulian**

**An academic: 2022-2023**

# Cuprins

1. Introducere .....	1
2. Modelul de Date UML.....	2
2.1 Schema UML restransa.....	3
2.2 Schema UML desfasurata.....	4
3. Tabele distincte pentru elementele programabile.....	5
3.1 Tabel procedures.....	6
3.2 Tabel triggers.....	7
4. Model al interfetei.....	8
4.1 Meniu.....	9
4.2 Login & Sign Up.....	10
5. Nivel de normalizare.....	11
6. Interogari.....	12
6.1 Afisarea membrilor grupului respectiv.....	13
6.2 Obtinerea materiei la care predă respectivul profesor.....	14
6.3 Afisarea activitatilor la care este inscris un student.....	15
6.4 Afisarea cursurilor unui profesor.....	16
6.5 Vizualizarea grupurilor de studiu a respectivului curs dat ca parametru.....	17
7. Obiecte de tip triggere, proceduri stocate.....	18
7.1 Sterge notele de la fiecare activitate.....	19
7.3 Inserarea unui student in baza de date.....	20
8. Descrierea aplicatiei.....	21
9. Detalii de programare a aplicatiei.....	22
10. Concluzii.....	23

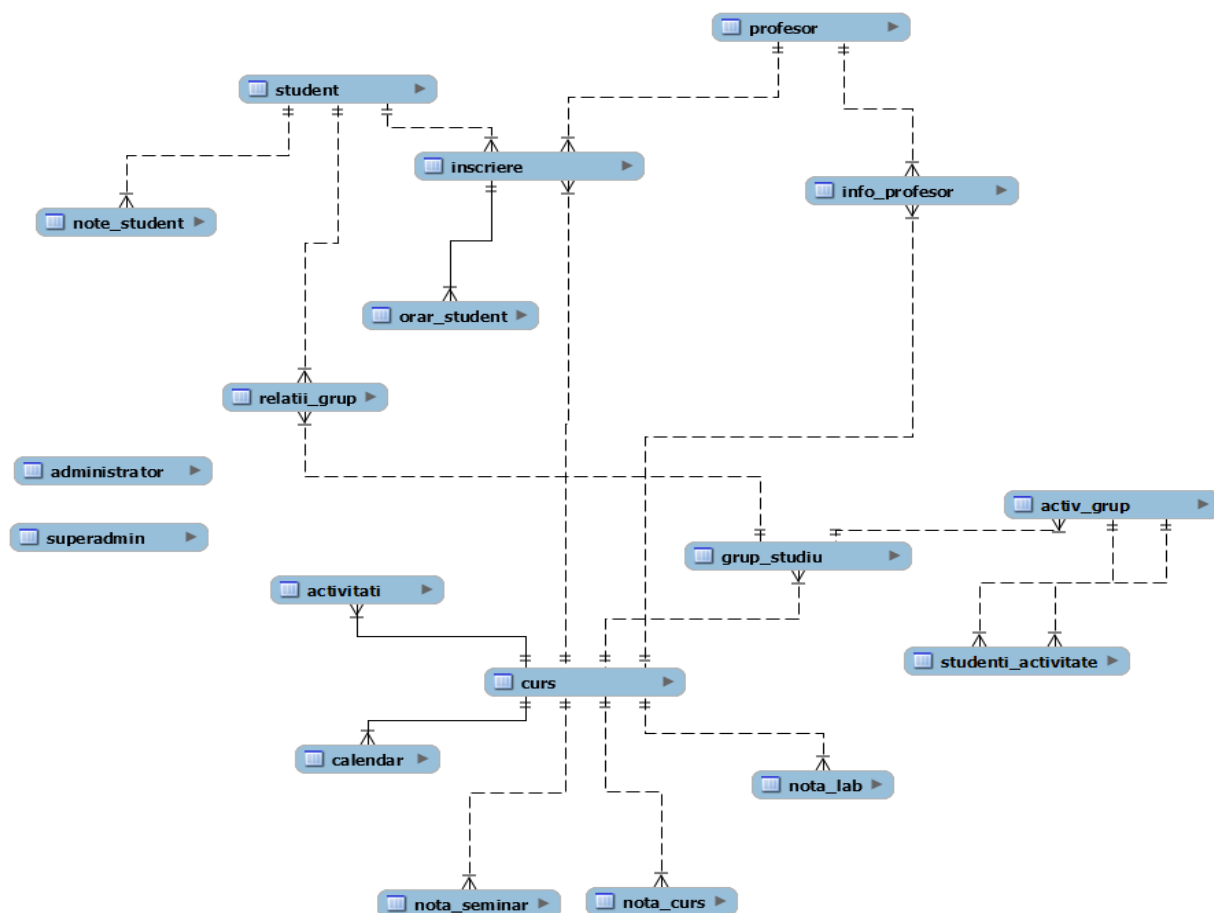
# **1. Introducere**

Aplicatia care urmeaza a fi prezentata isi propune gestionarea unei platforme de studiu, de tipul platformei Websinu, a proiectelor si evenimentelor in cadrul universitatii UTCN, precum si alte date absolut necesare pentru viitor. Creata pentru a fi de a folos studentilor, profesorilor si adminilor de retea, aplicatia se adreseaza tuturor celor care isi doresc sa si administreze si vizualizeze situatia profesionala ( cea din cadrul facultatii).

## 2. Modelul de Date – UML

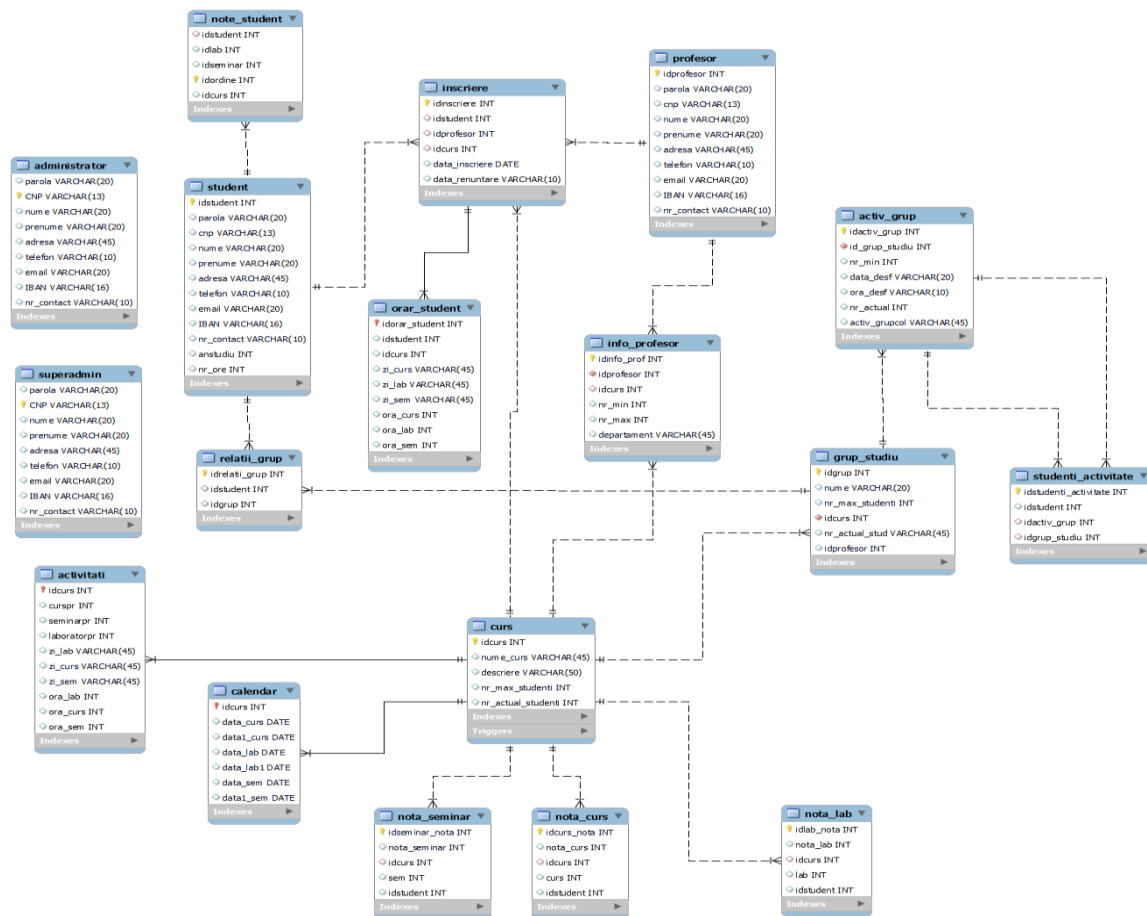
### 2.1 Schema UML restransa

Modelul de date, proiectat prin MySQL, are urmatoarea schema restransa:



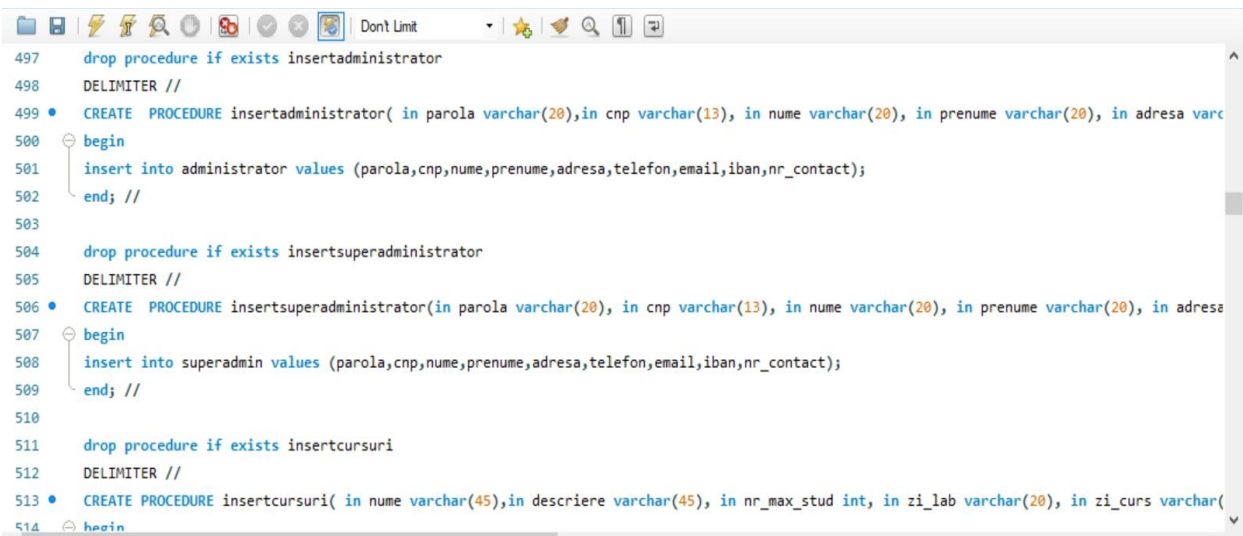
## 2.2 Schema UML desfasurata

Modelul de date, proiectat in MySQL, are urmatoarea schema desfasurata:



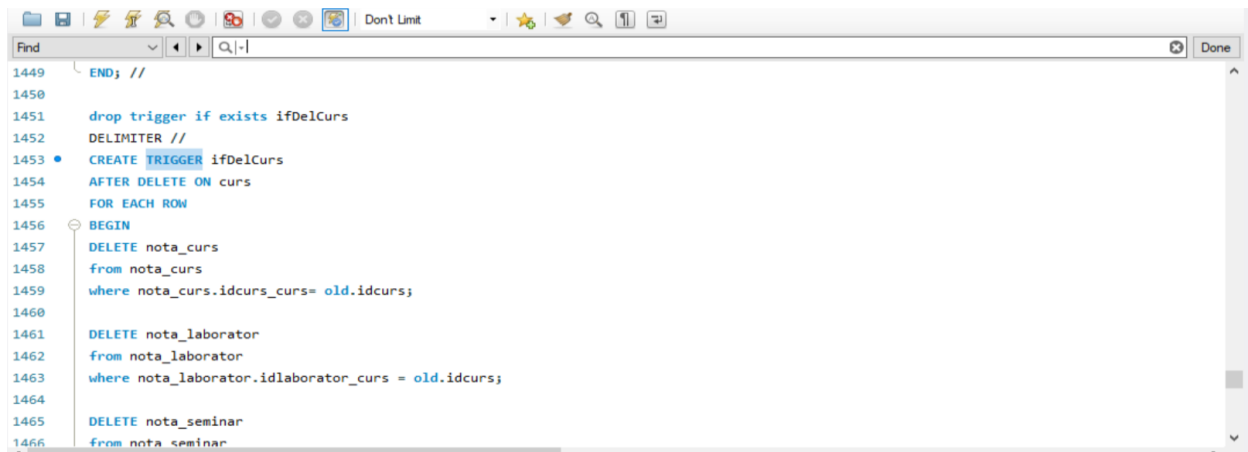
## 3. Tabele distincte pentru elementele programabile

### 3.1 Tabel procedures



```
497 drop procedure if exists insertadministrator
498 DELIMITER //
499 • CREATE PROCEDURE insertadministrator( in parola varchar(20), in cnp varchar(13), in nume varchar(20), in prenume varchar(20), in adresa varc
500 begin
501     insert into administrator values (parola,cnp,nume,prenume,adresa,telefon,email,iban,nr_contact);
502 end; //
503
504 drop procedure if exists insertsuperadministrator
505 DELIMITER //
506 • CREATE PROCEDURE insertsuperadministrator(in parola varchar(20), in cnp varchar(13), in nume varchar(20), in prenume varchar(20), in adresa
507 begin
508     insert into superadmin values (parola,cnp,nume,prenume,adresa,telefon,email,iban,nr_contact);
509 end; //
510
511 drop procedure if exists insertcursuri
512 DELIMITER //
513 • CREATE PROCEDURE insertcursuri( in nume varchar(45), in descriere varchar(45), in nr_max_stud int, in zi_lab varchar(20), in zi_curs varchar(
514 begin
```

## 3.2 Tabel triggers



The screenshot shows a SQL editor window with a toolbar at the top containing icons for file operations, execution, and search. Below the toolbar is a 'Find' bar with a search input field and a 'Done' button. The main area of the editor displays a SQL script with line numbers on the left. The script defines a trigger named 'ifDelCurs' that fires after a delete operation on a table named 'curs'. The trigger body contains three delete statements: one for 'nota\_curs', one for 'nota\_laborator', and one for 'nota\_seminar', all filtered by a condition on 'idcurs'.

```
1449 END; //
```

```
1450
```

```
1451 drop trigger if exists ifDelCurs
```

```
1452 DELIMITER //
```

```
1453 • CREATE TRIGGER ifDelCurs
```

```
1454 AFTER DELETE ON curs
```

```
1455 FOR EACH ROW
```

```
1456 BEGIN
```

```
1457 DELETE nota_curs
```

```
1458 from nota_curs
```

```
1459 where nota_curs.idcurs_curs= old.idcurs;
```

```
1460
```

```
1461 DELETE nota_laborator
```

```
1462 from nota_laborator
```

```
1463 where nota_laborator.idlaborator_curs = old.idcurs;
```

```
1464
```

```
1465 DELETE nota_seminar
```

```
1466 from nota_seminar
```

## 4. Model al interfetei

### 4.1 Sign Up & Log In

Interfata a fost proiectata pentru a oferi acces la datele personale nu doar a studentilor, ci si a profesorilor, adminilor si a superadminului.

Designul este unul prietenos, usor de folosit si inteles, care faciliteaza gestiunea datelor.



The image shows a screenshot of a web application's user registration interface. The interface is contained within a light blue window with standard window controls (minimize, maximize, close) in the top right corner. At the top left of the form area, it says "Select user:" followed by a dropdown menu currently showing "Student". Below this, the title "Introduceti datele:" is centered. The form consists of several labeled input fields stacked vertically: "Nume:", "Prenume:", "CNP:", "Adresa:", "Telefon:", "Email:", "IBAN:", "Nr Contra...", and "Parola:". At the bottom right of the form, there are two buttons: "Back" and "Sign Up".

Select user: Student ▼

**Introduceti datele:**

Nume:

Prenume:

CNP:

Adresa:

Telefon:

Email:

IBAN:

Nr Contra...

Parola:

Back Sign Up



## 4.2 Meniu

Fiecare utilizator are diverse privilegii, in functie de statutul profesional. Studentul poate naviga prin acest meniu pentru a vedea date legate de profilul personal, a se inscrie la cursuri, grupuri, activitati ale grupurilor si sa vizualizeze detalii despre acestea, totodata sa renunte la cele mentionate, sa-si vada orarul si mesajele de pe grupurile de activitati.

The screenshot shows a web application window titled "Bun venit, Gradinarariu!". The window has a light blue background and a white header bar. The header bar contains a navigation menu with the following items: "Afisare grupuri", "Afisare activitati", "Renuntare curs", "Renuntare grup", "Afisare cursuri", "Detalii", "Note", "Inscriere Curs", "Inscriere grupuri", and "Orar Student". The "Inscriere Curs" item is currently selected and highlighted in blue. Below the header bar, the main content area is divided into two sections. The top section contains three input fields labeled "Alege cursul:", "Profesorul:", and "CNP:", followed by a button labeled "Asignare curs". The bottom section contains two input fields labeled "An de studiu:" and "Numar ore:", followed by a button labeled "Adaugare date".

Bun venit, Gradinarariu!

Afisare grupuri Afisare activitati Renuntare curs Renuntare grup Afisare cursuri

Detalii Note Inscriere Curs Inscriere grupuri Orar Student

Alege cursul: Profesorul: CNP:

An de studiu: Numar ore:

Bun venit, Gradinariu!

Afisare grupuri Afisare activitati Renuntare curs Renuntare grup Afisare cursuri

Detalii Note Inscriere Curs Inscriere grupuri Orar Student


Alege cursul: Profesorul: CNP:

An de studiu: Numar ore:

Asignare curs

Adaugare date

Profesorul are mai multe roluri pe care le poate indeplini din interfata grafica, cum ar fi asignarea la cursuri, grupuri de activitati, sa creeze grupuri de studiu, sa selecteze procentele activitatilor cursului caruia ii apartine ( curs, seminar, laborator sau orice combinatie dintre cele trei), sa vizualizeze listele de elevi care sunt asignati la cursul lui, la grup, la activitati sau in acelasi timp sa paraseasca un curs, grup sau activitate, sa si vizualizeze orarul sau si al studentului.

— □ ×

Cursuri

Creare Grup

Selectie procente

Adaugare note

Grup

Detalii prof

Materie:

Procent Curs:

Procent Lab :

Procent seminar:

Adaugare procente

— □ ×

Cursuri

Creare Grup

Selectie procente

Adaugare note

Grup

Detalii prof

Nume grup:

Nr max stud:

Cursul:

Nume prof:

Prenume prof:

Creare Grup

idprofesor	idkurs	departament
4	2	Programare pe obiecte

Nume student:

Prenume student:

Nume curs:

Nota curs:

Nota seminar:

Nota lab:

Adminul are abilitatile de a modifica detalii despre studenti, profesori, cursuri, grupuri, activitati, sa stearga din baza de date cele 3 campuri mentionate mai sus, sa vizualizeze listele de studenti si profesori si restul de abilitati pe care le poate avea un profesor.

—

□

×

Modificare Procente

Detalii admin

Lista studenti

Lista profesori

Delete curs

Delete Student

Delete Profesor

Modificare date profesor

Modificare Grup

Creare Curs

Modificare curs

Modificare date student

Nume student:

Prenume student:

CNP:

Stergere student

—

□

×

Modificare Procente

Detalii admin

Lista studenti

Lista profesori

Delete curs

Delete Student

Delete Profesor

Modificare date profesor

Modificare Grup

Creare Curs

Modificare curs

Modificare date student

Introduceti datele profesorului:

Numele:

Prenume:

CNP:

Adresa noua:

Nr Tel nou:

Email nou:

IBAN nou:

Nr Contract...

Parola noua:

Modificare date



## 5. Nivelul de normalizare

Baza de date se afla in nivelul de normalizare Boyce-Codd.

Conform cercetatorului Chris Date, acest nivel de normalizare poate fi rezumat astfel: “Fiecare atribut trebuie sa reprezinte o informatie despre cheie, intreaga cheie si nimic in afara cheii”.

1. Fiecare atribut este atomic si nu depinde de alte attribute.
2. Fiecare atribut reprezinta o singura informatie.
3. Exista attribute cheie, reprezentate de id-ul unic al tabelelor.
4. Cheia primara este suficienta ca sa determine celelalte attribute ale tabelelor.
5. Cheia primara este singurul atribut care determina celelalte attribute, nu exista un alt atribut care poate determina in mod unic o tupla.

## 6. Interogari

In continuare se prezinta o serie de interogari ce demonstreaza cum se foloseste baza de date:

### 6. 1. Afisarea membrilor grupului respectiv

```
DELIMITER //
```

```
CREATE PROCEDURE membersGroup( in nume_grup varchar(20))
```

```
BEGIN
```

```

set @idgrup = (SELECT idgrup_studiu from grup_studiu where grup_studiu.num
= nume_grup );
select student.num, student.prenume
from student,relatii_grup
where student.idstudent = relatii_grup.idstudent
and relatii_grup.idgrup = @idgrup;
END; //

```

## 6. 2. Obținerea materiei la care predă respectivul profesor

```

DELIMITER //
CREATE PROCEDURE obtinmaterie(in nume_prof varchar(45))
BEGIN
SELECT curs.num_curs FROM curs,profesor,info_profesor WHERE nume_prof
= profesor.num AND profesor.idprofesor = info_profesor.idprofesor AND
curs.idcurs = info_profesor.idcurs;
end; //

```

## 6. 3. Afisarea activitatilor la care este inscris un student

```

DELIMITER //
CREATE PROCEDURE showActivitati( in nume_stud varchar(20), in
prenume_stud varchar(20), in CNP_stud varchar(13))
begin
SET @idstudent1 = (SELECT student.idstudent
from student
where student.num = nume_stud
and student.prenume = prenume_stud
and student.cnp = CNP_stud);

```



```
select nume_curs from curs, inscriere where inscriere.idstudent=@idstudent1 and  
curs.idcurs=inscriere.idcurs;  
end; //
```

#### **6. 4. Afisarea cursurilor unui profesor**

```
DELIMITER //  
CREATE PROCEDURE showCourses( in nume1 varchar(20))  
begin  
select nume_curs, nume from curs, profesor, info_profesor  
where curs.nume_curs=nume1  
and profesor.idprofesor=info_profesor.idprofesor  
and curs.idcurs=info_profesor.idcurs;  
end; //
```

#### **6.5. Vizualizarea grupurilor de studiu a respectivului curs dat ca parametru**

```
DELIMITER //  
CREATE PROCEDURE showGroups( in nume_curs varchar(45))  
begin  
SELECT grup_studiu.nume, grup_studiu.nr_max_studenti,curs.nume_curs  
from grup_studiu,curs  
where curs.idcurs = (SELECT curs.idcurs  
FROM curs  
WHERE curs.nume_curs = nume_curs )  
and grup_studiu.idcurs = curs.idcurs;  
end; //
```

## 6. 6. Vizualizarea grupurilor de studiu din care face parte studentul

DELIMITER //

```
CREATE PROCEDURE showGrupuriStud( in nume_stud varchar(20), in
prenume_stud varchar(20), in CNP_stud varchar(13))
begin
SET @idstudent1 = (SELECT student.idstudent
                    from student
                    where student.nume = nume_stud
                    and student.prenume = prenume_stud
                    and student.cnp = CNP_stud);

select nume from grup_studiu, relatii_grup where
relatii_grup.idstudent=@idstudent1 and grup_studiu.idgrup=relatii_grup.idgrup;
end; //
```

## 6. 7. Vizualizarea notelor studentului

DELIMITER //

```
CREATE PROCEDURE showNotes(in nume_stud varchar(20))
begin
SET @idstud = (SELECT student.idstudent
               from student
               where student.nume = nume_stud
               );

select distinct nume_curs, nota_curs, nota_lab, nota_seminar from curs, nota_curs,
nota_lab, nota_seminar, student, note_student, inscriere
where @idstud=student.idstudent and student.idstudent = note_student.idstudent
```

```

and student.idstudent = inscriere.idstudent
and note_student.idlab=nota_lab.idcurs
and note_student.idcurs=nota_curs.idcurs
and note_student.idseminar=nota_seminar.idcurs
and note_student.idcurs=curs.idcurs;
end; //
```

## 6. 8. Vizualizarea orar profesor

```
DELIMITER //
```

```
CREATE PROCEDURE showOrarProf(in nume varchar(20), in prenume
varchar(20), in cnp varchar(20), in zi varchar(10))
```

```
begin
```

```
select ora_lab from activitati join (select idcurs from info_profesor where
info_profesor.idprofesor=(SELECT profesor.idprofesor
```

```
from profesor
```

```
where profesor.nume = nume
```

```
and profesor.prenume = prenume
```

```
and profesor.cnp = cnp) ) as curs on activitati.idcurs=curs.idcurs where
activitati.zi_lab=zi;
```

```
select ora_sem from activitati join (select idcurs from info_profesor where
info_profesor.idprofesor=(SELECT profesor.idprofesor
```

```
from profesor
```

```
where profesor.nume = nume
```

```
and profesor.prenume = prenume
```

```

        and profesor.cnp = cnp) ) as curs on activitati.idcurs=curs.idcurs where
activitati.zi_sem=zi;

select ora_curs from activitati join (select idcurs from info_profesor where
info_profesor.idprofesor=(SELECT profesor.idprofesor

        from profesor

        where profesor.num = nume

        and profesor.prenume = prenume

        and profesor.cnp = cnp) ) as curs on activitati.idcurs=curs.idcurs where
activitati.zi_curs=zi;

end; //
```

## 6.9. Vizualizare orar student

```

DELIMITER //

CREATE PROCEDURE showOrarStud( in nume varchar(20), in prenume
varchar(20), in cnp varchar(20), in ziua varchar(45))

begin
```

```
SELECT orar_student.ora_curs, curs.ume_curs
FROM orar_student,curs
WHERE orar_student.idstudent = (SELECT student.idstudent from student where
student.ume = ume and student.prenume = prenume and student.cnp = cnp)
AND orar_student.zi_curs = ziua
AND curs.idcurs = orar_student.idcurs;
```

```
SELECT orar_student.ora_lab, curs.ume_curs
FROM orar_student,curs
WHERE orar_student.idstudent = (SELECT student.idstudent from student where
student.ume = ume and student.prenume = prenume and student.cnp = cnp)
AND orar_student.zi_lab = ziua
AND curs.idcurs = orar_student.idcurs;
```

```
SELECT orar_student.ora_sem, curs.ume_curs
FROM orar_student,curs
WHERE orar_student.idstudent = (SELECT student.idstudent from student where
student.ume = ume and student.prenume = prenume and student.cnp = cnp)
AND orar_student.zi_sem = ziua
AND curs.idcurs = orar_student.idcurs;
```

```
end; //
```

## 7. Obiecte de tip triggere, proceduri stocate

- Triggere

### 7.1 Sterge notele de la fiecare activitate

```
DELIMITER //
CREATE TRIGGER ifDelCurs
AFTER DELETE ON curs
FOR EACH ROW
BEGIN
DELETE nota_curs
from nota_curs
where nota_curs.idcurs_curs= old.idcurs;
DELETE nota_laborator
from nota_laborator
where nota_laborator.idlaborator_curs = old.idcurs;
DELETE nota_seminar
from nota_seminar
where nota_seminar.idseminar_curs = old.idcurs;
DELETE inscriere
from inscriere
where inscriere.id_curs = old.idcurs;
```

```
DELETE inf_profesor  
from inf_profesor  
where inf_profesor.id_curs = old.idcurs;  
END; //
```

## **7.2 Stergerea unui grup de studiu si relatiile acestuia**

```
delimiter //  
CREATE TRIGGER Delgrup  
AFTER DELETE ON grup_studiu  
for each row  
begin  
DELETE relatii_grup  
from relatii_grup  
where relatii_grup.id_grup = old.idgrup;  
end; //  
drop trigger Delgrup;
```

- **Proceduri**

## **7.3 Inserarea unui student in baza de date**

DELIMITER //

```
create procedure insertstudent(in parola varchar(20),in cnp varchar(13),in nume
varchar(20),in prenume varchar(20),in adresa varchar(45),in telefon varchar(10),in
email varchar(20),in iban varchar(16),in nr_contact varchar(10),in anstudiu int,in
nr_ore int)
```

```
begin
```

```
set @ID=( SELECT MAX(student.idstudent) FROM student) + 1;
```

```
if @ID IS NULL then
```

```
set @ID=1;
```

```
end if;
```

```
insert into student
```

```
values      (@ID,parola,cnp,nume,prenume,adresa,telefon,email,iban,nr_contact,
anstudiu, nr_ore);
```

```
end; //
```

## **7.4 Adaugarea unui profesor la grupul de studiu**

DELIMITER //

```
CREATE PROCEDURE addTeacherToGroup(in nume_grup varchar(35), in
nume_prof varchar(25))
```

```
begin
```

```
set @id=(select idgrup from grup_studiu where grup_studiu.nume=nume_grup);
```

```
set @idProf=(select idprofesor from profesor where profesor.nume=nume_prof );
```

```
update grup_studiu set idprofesor = @idProf where grup_studiu.idgrup=@id;
```

```
end; //
```

## **7.5 Crearea unei activitati de grup**



DELIMITER //

```
CREATE PROCEDURE CreateActivityGroup(in nume_grup varchar(20), in
nume_activ varchar(60), in data_desf varchar(20), in ora_exp varchar(10), in
nr_min int)
```

```
begin
```

```
set @ID=( SELECT MAX(activ_grup.idactiv_grup) FROM activ_grup) + 1;
```

```
if @ID IS NULL then
```

```
set @ID=1;
```

```
end if;
```

```
set @idgrup =( select idgrup from grup_studiu where
grup_studiu.nume=nume_grup);
```

```
insert into activ_grup values (@ID, @idgrup, nr_min, data_desf, ora_exp,0,
nume_activ);
```

```
end; //
```

## **7.6 Adaugarea unui curs in calendar (orar)**

DELIMITER //

```
CREATE PROCEDURE addToCalendar(in nume_curs varchar(10), in data_sem
date, in data1_sem date, in data_lab date, in data1_lab date, in data_curs date, in
data1_curs date)
```

```
begin
```

```
set @idcurs = ( SELECT curs.idcurs
```

```
from curs
```

```
where curs.nume_curs = nume_curs );
```

```
insert into calendar values(@idcurs, data_curs, data1_curs, data_lab, data1_lab,
data_sem, data1_sem);
```

end; //

## 8. Descrierea aplicatiei

### 8.1 Operatii posibile

In scopul utilizarii mai usoare a bazei de date am dezvoltat o aplicatie demonstrative prin care se pot efectua urmatoarele operatii:

- Conectare la o baza de date ( in mod implicit localhost )

**Atentie!** Programul functioneaza doar cu o baza de date care are in structura tabelele si procedurile mentionate in capitolele anterioare. Pentru generarea structurii se foloseste script-ul ce s-a obtinut din exportarea bazei de date.

- Vizualizarea datelor personale.
- Logarea ca utilizator si abilitatea de a participa la activitati de grup, in calitate de student.
- Crearea unui nou student, profesor sau admin, pe baza procedurilor stocate din baza de date.
- Afisarea orarului studentului, profesorului
- Afisarea unor sugestii de participare la grupurile de studii pentru studenti
- Posibilitatea de adaugare a unui cadru didactic la grupul de studiu

### 8.1 Tehnologii folosite

Pentru realizarea aplicatiei am folosit platforma Swing in IDE ul Eclipse si WorkBenchMySQL pentru baza de date.

### 8.2. Conectare

Conectarea se realizeaza in urma specificarii urmatorilor parametri: adresa bazei de date ( hostname si port ), username ul acesteia si parola pe care am setat-o la inceput pentru a se realiza conexiunea.

Exista, desigur, si optiunea de deconectare, pentru a se putea sa ne conectam la alta baza de date.

### **8.3. Logare**

Aici putem sa ne logam ca utilizatori. Cu ajutorul SQL, obtinem hash-ul corespunzator username-ului introdus, il comparam cu hash-ul obtinut din criptarea parolei introduse si in caz de potrivire se actualizeaza ID ul clientului.

### **8.4 Inregistrare**

In tab ul Register putem sa autentificam un user, student, profesor sau admin, folosind toate cazurile posibile ale procedurilor stocate de creare.

### **8.5 Clase**

Proiectul are urmatoarea structura:

- Login\_interface.java
- RegisterInterface.java
- Admininterface.java
- Profinterface.java
- Sadmininterface.java
- Studentinterface.java

## 9. Detalii de programare a aplicatiei

In majoritatea cazurilor, multe dintre utilitati au fost implementate folosind SQL, fie prin clasa Statement, fie prin CallableStatement, avantajul celei de-a doua este ca se specifica forma interogari si argumentele se specifica pe urma. Pentru interogari de tip SELECT am folosit metoda executeQuery si pt UPDATE si INSERT am folosit metoda executeUpdate.

Cu ajutorul Java ului, am implementat procedurile si am pus conditii de verificare pentru datele de intrare, sa nu fie eronate si sa nu fie permis accesul din greseala in interfata.

## 10. Concluzii

In concluzie, dezvoltarea aplicatiilor folosind Java si MySQL este posibila, dar este o varianta de implementare a aplicatiilor intre multe variante.

**No table of contents entries found.**

Legat de aplicatia de fata, dupa cum am mai mentionat este demonstrativa, dar metodele folosite aici se pot folosi usor pentru a implementa o aplicatie reala.

