

**FACULATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

**PROIECT DE SEMESTRU**

LA DISCIPLINA BAZE DE DATE

**PLATFORMĂ DE STUDIU**

Nume: Vultur Ana Maria

An academic 2020-2021

**Cuprins**

[**1.Introducere** 3](#_Toc61464384)

[**2. Analiza cerințelor utilizatorilor** 4](#_Toc61464385)

[2.1. Ipotezele facultații 4](#_Toc61464386)

[2.2. Organizarea structurată tabelar a cerințelor utilizator 5](#_Toc61464387)

[2.3. Determinarea și caracterizarea profilurilor de utilizator 5](#_Toc61464388)

[**3.Modelul de date și descrierea acestuia** 6](#_Toc61464389)

[3.1. Entitați și atributele lor 6](#_Toc61464390)

[3.2.Diagrama EER pentru modelul de date complet 12](#_Toc61464391)

[3.3. Proceduri, view-uri 12](#_Toc61464392)

[3.4.Normalizarea datelor 23](#_Toc61464393)

[3.5. Interogari MySQL 24](#_Toc61464394)

[**4. Detalii de implementare** 27](#_Toc61464395)

[4.1. Structura claselor in Java 27](#_Toc61464396)

[4.2. Manual de utilizare/instalare 27](#_Toc61464397)

[4.3. Elemente de Securitate a aplicatiei 28](#_Toc61464398)

[**5.Concluzii. Limitari si dezvoltati ulterioare** 28](#_Toc61464399)

# **1.Introducere**

Proiectul presupune dezvoltarea unui sistem informatic destinat gestiunii unei platforme de studiu. Scopul aplicației este simplificarea operațiilor cu baza de date prin oferirea unei interfețe grafice pe care administartorul,profesorii și studențîi să o poată utiliza. Aplicața oferaă sprinjin atât pentru interogarea bazei de date cât și pentru manipularea acesteia.

Utilizatorul de tip administrator poate adăuga, modifică şi şterge informaţii în baza de date , informații legate de utilizatori. De asemenea, va exista şi un rol de super-administrator care poate opera inclusiv asupra utilizatorilor de tip administrator. Administratorii pot să caute utilizatorii după nume și îi pot filtra după tip, pot asigna profesorii la cursuri și pot face căutare după numele cursului.

Aplicația va permite gestiunea cu ușurință a activităților didactice și astfel a interacțiunilor dintre studenți și profesori. Cursurile sunt predate de mai mulți profesoriși au una sau mai multe tipuri de activități. Studenții se pot înscrie la cursuri și sunt asignati profesorului cu cei mai puțini studenți la dată înscrierii. Aceștia sunt evaluate cu note pentru fiecare tip de activitate și primesc o notă finală că medie ponderată între tipurile de activități.

Profesorul poate programa activitățile într-un calendar, pe zile și ore, specificând și numărul maxim de participanți. Activitățile pot fi programate doar în viitor. Profesorii pot accesa un catalog, unde pot filtra studențîi după cursuri și le pot adaugă note.

Studenții se pot înscrie la cursuri, pot renunța la cursuri și își pot vedea notele. Aceștia trebuie să aleagă activitățile la care vor să participe și pot participa la ele doar dacă mai sunt locuri sau nu există o suprapunere cu o altă activitate. Totodată, studențîi se pot înscrie în grupuri de studiu pentru o anumită materie, dacă sunt înscrisi la materia respectivă.

Aplicația este utilă pentru o facultate, deoarece se pot realiza toate funcționalitățile necesare acesteia din interfața grafică ,și totodată informațiile despre studenți, profesori și administratori sunt stocate într-o bază de date.Pentru dezvoltarea proiectului am folosit:

* **MySQL Workbench**- pentru crearea bazei de date, popularea initiala, dezvltarea de viewuri, procedure
* **Eclipse, IntelliJ** – mediu de dezvoltare Java,unde am realizat interfața grafică

# **2. Analiza cerințelor utilizatorilor**

## 2.1. Ipotezele facultății

* Există 3 tipuri de utilizatori: administartorul (super-administratorul), studentul și profesorul.
* Fiecare utilizator este identificat printr-un id și mai este reținut numele, prenumele, CNP-ul, numărul de telefon, adresa ,email, IBAN-ul și numarul de contract.
* Administartorul este identificat printr-un id, are un id\_utilizator pentru a face legatura cu toate datele menționate anterior și un câmp prin care să putem vedea daca este super administrator sau nu.
* Profesorul este identificat prin id, are id\_utilizator pentru a face legatura cu utilizatorul , numărul minim și maxim de ore pe care le poate preda și departamentul.
* Studentul este identificat prin id, are id\_utilizator, anul de studiu și numarul de ore pe care trebuie să le susțină.
* Cursurile sunt identificate prin id, au descriere, numărul maxim de studenți și numărul de studenți înscriși.
* Exista o tabelă cu note unde avem id\_student și id\_curs pentru a face legatura și a ști ce student și la ce curs a obținut o nota.
* Tabela Ponderi\_note gestionează poderile activitaților (laborator, seminar și curs).
* Tabelele grupa\_studiu și activitate ajută la înscrierea studenților într-o grupă de studiu.
* Tabela mesaje\_grup ne dă posibilitatea ca studenții înscriși într-o grupă de studio sa își lase mesaje.
* Iar tabela calendar ne definește un orar pentru profesori, studenți și cursurile lor, unde avem reținută ziua, data, ora, activiatea care se desfășoară și numărul de participanți.
* Încă 3 tabele necesare pentru a realiza legatura de many to many :inscriere\_curs , curs\_profesor și membrii\_grupa.

## 2.2. Organizarea structurată tabelar a cerințelor utilizator

Baza de date trebuie să stocheze urmatoarele informații:

* utilizatori
* cursuri
* note
* orar
* grupe de studiu

Mai mult, trebuie sa permită si urmatoarele operatii:

* Ștergerea, adăugarea și schimbarea informațiilor despre utilizatori
* Vizualizare informații cursuri, informații utilizatori
* Vizualizare note, grupe studiu, activitati curente/viitoare
* Înscrierea unui student la curs
* Adăugare si gestionare activități
* Notare studenți
* Gestionarea ponderilor notelor
* Vizualizare studenți înscriși la curs și descărcare cataloage

## 2.3. Determinarea și caracterizarea profilurilor de utilizator

Aplicația are 3 tipuri de utiliztori:

1. Administratorul:

* Autentificarea în aplicație
* Vizualizare informații personale
* Căutare utilizatori dupa nume
* Căutare curs
* Adaugare,ștergere,modificare informații despre utilizatori
* Aceste operații făcute pe administrator,nu se pot realiza decât daca este super-administrator

1. Profesorul:

* Autentificarea în aplicație
* Vizualizare informații personale
* Adăugare si gestionare activități
* Notare studenți
* Gestionarea ponderilor notelor
* Vizualizare studenți înscriși la curs și descărcare cataloage

1. Studentul:

* Autentificarea în aplicație
* Vizualizare informații personale
* Vizualizare note,grupe studiu,activitati curente/viitoare
* Înscrierea unui student la curs
* Înscriere in / parasire grup de studiu

# **3.Modelul de date și descrierea acestuia**

## 3.1. Entitați și atributele lor

* **Utilizator**:id,Cnp,nume,prenume,adresă,email,numar de telefon,IBAN,numar contract.Se leaga de administrator,professor și student.
* **Administartor**:id,id\_utilizator,super\_administrator.Se leaga de utiliztor.
* **Profesor**: id, id\_utilizator, număr minim ore, număr maxim, departamentul, număr toatal studenți. Se leaga de utilizator.
* **Student**: id, id\_utilizator, an studiu, număr ore. Se leaga de utilizator.
* **Cursuri**: id, număr student inscriși, număr maxim studenți, descriere.
* **Ponderi note**: id,pondere laborator,seminar,curs.
* **Note**:id,id\_student,id\_curs,note.Se leaga de student,curs si ponderi note.
* **Grupa studiu**:id,id\_curs,nume.Se leaga la curs.
* **Mesaje grup:** id,mesaj,id\_student,id\_grupa.Se leaga la grupa studiu.
* **Activitati**: id, id\_grupa, denumire, data, ora, număr studenți maxim, student inscriși, durata. Se leaga la grupa de studio.
* **Calendar**:id, id\_profesor, id\_curs, activitate, data, oră, zi, durată, număr participanți. Se leaga la professor și curs.
* **Înscriere curs**: id, id\_student, id\_curs, data. Se legă la student si cursuri.
* **Curs\_profesor**: id, id\_curs, id\_profesor. Se leagă la profesor și curs.
* **Membrii grupa**: id, id\_grupa, id\_student. Se leagă la grupa de studiu și student.

**Codul pentru crearea tabelelor:**

create database proiect;

use proiect;

create table if not exists utilizator(

utilizator\_id INT primary key auto\_increment,

CNP VARCHAR(25),

nume VARCHAR(25),

prenume VARCHAR(25),

adresa VARCHAR(25),

IBAN VARCHAR(25),

numar\_telefon VARCHAR(25),

numar\_contract INT,

email VARCHAR(25)

);

create table if not exists student(

ID\_utilizator INT,

ID\_student INT primary key auto\_increment,

an\_studiu INT,

ore INT,

constraint foreign key (ID\_utilizator) references utilizator(utilizator\_id)

);

CREATE table if not exists profesor(

ID\_utilizator INT,

ID\_profesor INT PRIMARY KEY auto\_increment,

numar\_minim\_ore INT,

numar\_maxim\_ore INT,

departament VARCHAR(25),

numar\_total\_studenti INT,

constraint foreign key (ID\_utilizator) references utilizator(utilizator\_id)

);

create table if not exists administrator(

ID\_utilizator INT,

ID\_administrator INT PRIMARY KEY auto\_increment,

super\_administrator BOOLEAN,

constraint foreign key (ID\_utilizator) references utilizator(utilizator\_id)

);

create table if not exists cursuri(

ID\_curs INT PRIMARY KEY AUTO\_INCREMENT,

id\_profesor INT,

descriere VARCHAR(25),

numar\_maxim\_studenti INT,

numar\_studenti\_inscrisi INT

);

create table if not exists ponderi\_note(

ID\_curs INT PRIMARY KEY,

pondere\_laborator int,

pondere\_seminar int,

pondere\_curs int

);

create table if not exists note(

id\_nota INT PRIMARY KEY AUTO\_INCREMENT,

id\_student INT,

ID\_curs INT,

nota\_curs decimal(4,2),

nota\_laborator decimal(4,2),

nota\_seminar decimal(4,2),

medie decimal(4,2),

constraint foreign key (ID\_student) references student(ID\_student),

constraint foreign key (ID\_curs) references cursuri(ID\_curs),

constraint foreign key(ID\_curs) references ponderi\_note(ID\_curs)

);

create table if not exists grupa\_studiu(

ID\_grupa INT primary key auto\_increment,

materie\_ID int,

nume\_grup VARCHAR(25),

constraint foreign key (materie\_ID) references cursuri(ID\_curs)

);

create table if not exists mesaje\_grup(

id\_mesaj INT PRIMARY KEY auto\_increment,

mesaj VARCHAR(100),

id\_student INT,

id\_grupa INT,

constraint foreign key (id\_grupa) references grupa\_studiu(ID\_grupa)

);

create table if not exists activitate(

id\_activitate int primary key auto\_increment,

ID\_grupa int,

denumire varchar(25),

data\_organizare DATE,

ora VARCHAR(25),

numar\_minim\_participanti INT,

numar\_studenti\_inscrisi INT,

durata\_expirare INT,

data\_creare datetime,

constraint foreign key (ID\_grupa) references grupa\_studiu(ID\_grupa)

);

create table if not exists calendar(

ID\_activitate INT PRIMARY KEY auto\_increment,

ID\_profesor int,

id\_curs INT,

activitate VARCHAR(25),

data\_inceput DATE,

data\_incheiere DATE,

ora\_inceput VARCHAR(5),

ora\_incheiere VARCHAR(5),

zi VARCHAR(9),

numar\_participanti int,

constraint foreign key (ID\_profesor) references profesor(ID\_profesor),

constraint foreign key (ID\_curs) references cursuri(ID\_curs)

);

create table if not exists inscriere\_curs(

ID\_inscriere INT PRIMARY KEY AUTO\_INCREMENT,

ID\_student int,

ID\_curs int,

data\_inscriere DATE,

data\_renuntare DATE,

constraint foreign key (ID\_student) references student(ID\_student),

constraint foreign key (ID\_curs) references cursuri(ID\_curs)

);

create table if not exists curs\_profesor(

ID\_curs int,

id INT primary key auto\_increment,

ID\_profesor int,

constraint foreign key (ID\_profesor) references profesor(ID\_profesor),

constraint foreign key (ID\_curs) references cursuri(ID\_curs)

);

create table if not exists membrii\_grupa(

ID\_grupa INT,

ID\_membru INT primary key AUTO\_INCREMENT,

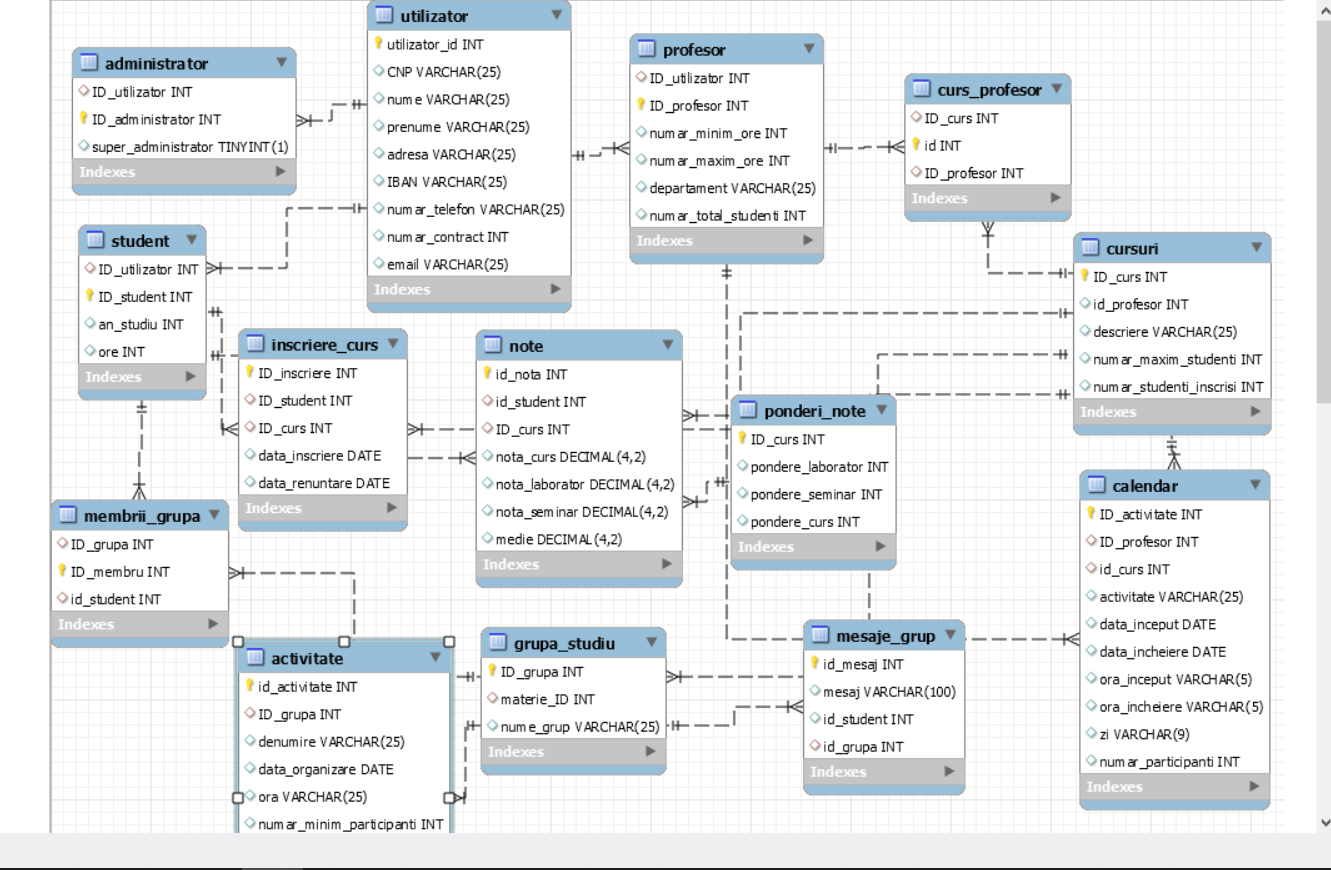
id\_student INT,

constraint foreign key(ID\_grupa) references grupa\_studiu(ID\_grupa),

constraint foreign key(id\_student) references student(ID\_student)

);

## 3.2.Diagrama EER pentru modelul de date complet



## 3.3. Proceduri, view-uri

* **Proceduri administrator:**

1. **insert\_utilizator**(CNP VARCHAR(25),nume VARCHAR(25),prenume VARCHAR(25),adresa VARCHAR(25),IBAN VARCHAR(25),numar\_telefon VARCHAR(25),numar\_contract INT,email VARCHAR(25))

Insereaza un utilizator.

1. **insert\_student**(ID\_utilizator INT,an\_studiu INT,ore INT)

Insereaza un student.

1. **insert\_profesor**(ID\_utilizator INT,numar\_minim\_ore INT,numar\_maxim\_ore INT, departament VARCHAR(25),numar\_total\_studenti INT)

Insereaza un professor.

1. **insert\_administrator**(ID\_utilizator1 INT,super\_administrator1 BOOLEAN,nume1 varchar(25),prenume1 varchar(25))

Insereaza un administrator.

*-- super-administratorul poate sa insereze administratori*

*delimiter //*

*create procedure insert\_administrator(ID\_utilizator1 INT,super\_administrator1 BOOLEAN,nume1 varchar(25),prenume1 varchar(25))*

*begin*

*declare ids boolean;*

*set ids=(select super\_administrator from administrator,utilizator where utilizator\_ID=ID\_utilizator and utilizator.nume=nume1 and utilizator.prenume=prenume1);*

*if ids=true*

*then*

*insert into administrator(ID\_utilizator,super\_administrator) values (ID\_utilizator1,super\_administrator1);*

*end if;*

*end;//*

1. **delete\_student**(nume varchar(25),prenume varchar(25),cnp varchar(25))

Sterge un student;

1. **delete\_profesor**(nume varchar(25),prenume varchar(25),cnp varchar(25))
2. delete\_administrator(nume varchar(25),prenume varchar(25),nume1 varchar(25),prenume1 varchar(25),cnp varchar(25))

Sterge un professor.

1. **delete\_administrator**(nume varchar(25),prenume varchar(25),nume1 varchar(25),prenume1 varchar(25),cnp varchar(25))

Sterge un professor.

1. **update\_adresa**(nume varchar(25),prenume varchar(25),adresa varchar(25),cnp varchar(25))

Schimba adresa unui utilizator.

1. **update\_email**(nume varchar(25),prenume varchar(25),email varchar(25),cnp varchar(25))

Schimba email ul unui utilizator.

1. **update\_telefon(**nume varchar(25),prenume varchar(25),numar\_telefon varchar(25),cnp varchar(25))

Schimba telefonul unui utilizator.

*delimiter //*

*create procedure update\_telefon(nume varchar(25),prenume varchar(25),numar\_telefon varchar(25),cnp varchar(25))*

*begin*

*declare id int;*

*set id=(select utilizator\_id from utilizator where utilizator.nume=nume and utilizator.prenume=prenume and utilizator.CNP=cnp);*

*update utilizator set numar\_telefon=numar\_telefon*

*where utilizator.utilizator\_id=id;*

*end;//*

1. **update\_contract**(nume varchar(25),prenume varchar(25),numar\_contract varchar(25),cnp varchar(25))

Schimba nr contractului unui utilizator.

1. **update\_iban**(nume varchar(25),prenume varchar(25),IBAN varchar(25),cnp varchar(25))

Schimba adresa unui utilizator.

1. **update\_nrore**(nume varchar(25),prenume varchar(25),numar\_maxim\_ore INT,cnp varchar(25))

Schimba numarul de ore unui utilizator.

1. **update\_an**(nume varchar(25),prenume varchar(25),an\_studiu INT,cnp varchar(25))

Schimba anul de studiu unui utilizator.

1. **cauta\_nume**(nume varchar(25),prenume varchar(25))

Cauta un utilizator dupa nume

1. **filtare\_studenti**()

Filtreaza toti studentii;

1. **filtare\_profesori**()

Filtreaza toti profesorii.

1. **filtare\_administratori**()

Filtreaza tori adminisatratorii.

1. **asignare\_prof**(nume1 varchar(25),prenume1 varchar(25),numecurs varchar(25))

Asigneaza un professor la un curs.

*delimiter //*

*create procedure asignare\_prof(nume1 varchar(25),prenume1 varchar(25),numecurs varchar(25))*

*begin*

*declare id int;*

*declare idcurs int;*

*declare idu int;*

*set id=(select ID\_profesor from profesor,utilizator where utilizator\_ID=ID\_utilizator and utilizator.nume=nume1 and utilizator.prenume=prenume1);*

*set idcurs=(select ID\_curs from cursuri where cursuri.descriere=numecurs LIMIT 1);*

*set idu=(select id from curs\_profesor where id\_profesor=id and id\_curs=idcurs);*

*if idu is null*

*then*

*insert into curs\_profesor(ID\_curs,ID\_profesor) values (idcurs,id);*

*update cursuri set id\_profesor=id where id\_curs=idcurs;*

*end if;*

*end;//*

1. **cauta\_curs**(nume varchar(25))

Cauta profesorii care predau cursul respective;

1. **cauta\_studenti**(nume1 varchar(25))

Cauta studentii care sunt inscrisi la un curs.

* **Proceduri professor:**

1. **adaugaActivitate**(nume\_activitate varchar(25), cnpp varchar(25), dataInit date, dataFinal date, numar int, nume\_curs varchar(25),ora\_i varchar(5),ora\_s varchar(5),zi varchar(9))

Adauga o activitate in orar.

*DELIMITER |*

*CREATE PROCEDURE adaugaActivitate(nume\_activitate varchar(25), cnpp varchar(25), dataInit date, dataFinal date, numar int, nume\_curs varchar(25),ora\_i varchar(5),ora\_s varchar(5),zi varchar(9))*

*BEGIN*

*declare ID\_c, ID\_p, ID\_u,ID\_a int;*

*set ID\_u=(select utilizator\_id from utilizator where (cnp=cnpp));*

*set ID\_p=(select id\_profesor from profesor where profesor.id\_utilizator=ID\_u);*

*set ID\_c=(select id\_curs from cursuri where descriere=nume\_curs and id\_profesor=id\_p);*

*set ID\_a=(select ID\_activitate from calendar where calendar.activitate=nume\_activitate and calendar.ID\_profesor=id\_p and calendar.id\_curs=ID\_c);*

*if ID\_a is not NULL*

*then*

*update calendar*

*set data\_inceput=dataInit,*

*data\_incheiere=dataFinal,*

*ora\_inceput=ora\_i,*

*ora\_incheiere=ora\_s,*

*calendar.zi=zi,*

*numar\_participanti=numar*

*where calendar.ID\_profesor=ID\_p and calendar.ID\_activitate=ID\_a and calendar.id\_curs=ID\_c;*

*else*

*insert into calendar values(NULL,ID\_p,ID\_c,nume\_activitate,dataInit,dataFinal,ora\_i,ora\_s,zi,numar);*

*end if;*

*END;*

*|*

1. **GestionarePonderi**(pLab int,pCurs int,pSem int,numeCurs varchar(25),idu int)

Caluleaza nota in functie de ponderi.

1. **utId(**cnpp varchar(25))

Selecteaza id-ul utilizatorului cu acel cnp.

1. **notare**(num varchar(25),prenum varchar(25),notaS numeric(4,2),notaL numeric(4,2),notaC numeric(4,2),numeCurs varchar(25))

Adauga note in tabela de note.

*DELIMITER |*

*CREATE PROCEDURE notare(num varchar(25),prenum varchar(25),notaS numeric(4,2),notaL numeric(4,2),notaC numeric(4,2),numeCurs varchar(25))*

*BEGIN*

*DECLARE id int;*

*DECLARE idC int;*

*DECLARE pS,pC,pL,idv int;*

*set id=(SELECT ID\_student from student inner join utilizator on utilizator.utilizator\_id=student.ID\_utilizator and utilizator.nume=num and utilizator.prenume=prenum);*

*set idC=(select cursuri.id\_curs from cursuri inner join inscriere\_curs on (inscriere\_curs.id\_student=id and inscriere\_curs.id\_curs=cursuri.id\_curs)where cursuri.descriere=numeCurs);*

*set pS=(SELECT pondere\_seminar from ponderi\_note where ID\_curs=idC);*

*set pC=(SELECT pondere\_curs from ponderi\_note where ID\_curs=idC);*

*set pL=(SELECT pondere\_laborator from ponderi\_note where ID\_curs=idC);*

*set idV=(SELECT ID\_nota from note where id\_student=id and id\_curs=idC);*

*if(idV is NULL)*

*then*

*if pL=0*

*then*

*if pS=0*

*then insert into note values(NULL,id,idC,notaC,0,0,0.00);*

*else*

*insert into note values(NULL,id,idC,notaC,0,notaS,0.00);*

*end if;*

*else*

*if pS=0*

*then*

*insert into note values(NULL,id,idC,notaC,notaL,0,0.00);*

*else*

*insert into note values(NULL,id,idC,notaC,notaL,notaS,0.00);*

*end if;*

*end if;*

*update note set medie=((nota\_seminar\*pS/100)+(nota\_laborator\*pL/100)+(nota\_curs\*pC/100)) where id\_student=id and id\_curs=idC;*

*else*

*if pL=0*

*then*

*if pS=0*

*then*

*update note set nota\_laborator=0, nota\_seminar=0;*

*else*

*update note set nota\_laborator=0;*

*end if;*

*else*

*if pS=0*

*then*

*update note set nota\_seminar=0;*

*else*

*update note set nota\_curs=notaC, nota\_laborator=notaL, nota\_seminar=notaS, medie=0.00 where id\_nota=idV;*

*end if;*

*end if;*

*update note set medie=((nota\_seminar\*pS/100)+(nota\_laborator\*pL/100)+(nota\_curs\*pC/100))where id\_student=id and id\_curs=idC;*

*end if;*

*update note set medie=((nota\_seminar\*pS/100)+(nota\_laborator\*pL/100)+(nota\_curs\*pC/100)) where id\_student=id and id\_curs=idC;*

*END;*

1. *|***viz\_note**(num varchar(25),prenum varchar(25),numeCurs varchar(25))

Poti vedea notele.

1. **viz\_inf\_as\_profesor**(cnpp varchar(25))

Vede informatii despre el

1. **lista\_studenti**(nume\_curs varchar(25), CNPP varchar(25))

Vede studentii de la un curs.

*DELIMITER |*

*CREATE PROCEDURE lista\_studenti(nume\_curs varchar(25), CNPP varchar(25))*

*BEGIN*

*DECLARE idP,idU,idC int;*

*set idU=(select utilizator\_id from utilizator where CNP=CNPP);*

*set idP=(select id\_profesor from profesor where profesor.id\_utilizator=idU);*

*set idC=(select id\_curs from cursuri where descriere=nume\_curs and id\_profesor=idP);*

*SELECT utilizator.nume, utilizator.prenume*

*from utilizator*

*inner join student on utilizator.utilizator\_id=student.ID\_utilizator*

*inner join inscriere\_curs on student.id\_student=inscriere\_curs.id\_student*

*where inscriere\_curs.ID\_curs=idC*

*order by utilizator.nume;*

*END;*

1. *|***vizualizare\_activitati\_curente\_AS\_profesor**(cnpp varchar(25))

Vede activitatile curente.

1. **vizualizare\_activitati\_viitoare\_AS\_profesor**(cnpp varchar(25))

Vede activitatile viitoare.

*DELIMITER |*

*CREATE PROCEDURE vizualizare\_activitati\_viitoare\_AS\_profesor(cnpp varchar(25))*

*BEGIN*

*declare idP,idU int;*

*set idU=(select utilizator\_id from utilizator where (cnp=cnpp));*

*set idP=(select id\_profesor from profesor where profesor.id\_utilizator=idU);*

*SELECT utilizator.nume, utilizator.prenume, calendar.activitate as 'ACTIVITATI VIITOARE',calendar.data\_inceput as "DATA INCEPUT", calendar.ora\_inceput, calendar.ora\_incheiere, calendar.zi*

*from utilizator*

*inner join profesor on utilizator.utilizator\_id=profesor.ID\_utilizator*

*inner join calendar on profesor.ID\_profesor=calendar.ID\_profesor*

*where calendar.data\_inceput>current\_date and profesor.id\_profesor=idP;*

*END;*

*|*

1. **vizualizare\_cursuri**(cnpp varchar(25))

Vede cursurile unui professor.

1. **viz\_cnp**(num varchar(25), prenum varchar(25))

Selecteaza cnp-ul unui utilizator cu acel nume

1. **viz\_id**(num varchar(25), prenum varchar(25))

Selecteaza id-ul unui utilizator cu acel nume

1. **viz\_ore**(CNP varchar(25), stud\_id int)

Selecteaza numarul de ore ale unui student

1. **viz\_zile**(CNP varchar(25), stud\_id int)

Selecteaza zilele din orar

* **Proceduri student:**

1. **cautare\_cursuri**()

Cauta cursurile

1. **cautare\_grupuri**()

Cauta grupurile

1. **cautare\_activitati**(id\_grupa int)

Cauta activitatile unei grupe

1. **inscriere\_curs**(nume VARCHAR(25), id\_stud INT, nr\_studenti INT)

Inscrie un student la curs

1. **vizualizare\_info** (nume VARCHAR(25), CNP VARCHAR(25))

Vizualizeaza informatii despre el

1. **vizualizare\_note** (CNP VARCHAR(25))

Isi vede notele

*DELIMITER //*

*CREATE PROCEDURE vizualizare\_note (CNP VARCHAR(25))*

*BEGIN*

*SELECT cursuri.descriere, note.nota\_curs , note.nota\_seminar, note.nota\_laborator, note.medie*

*FROM utilizator*

*JOIN student ON utilizator.utilizator\_id = student.id\_utilizator*

*JOIN inscriere\_curs ON student.id\_student = inscriere\_curs.id\_student*

*JOIN cursuri ON inscriere\_curs.id\_curs = cursuri.id\_curs*

*JOIN note ON cursuri.id\_curs = note.id\_curs*

*WHERE CNP = utilizator.CNP*

*GROUP BY cursuri.descriere;*

*END; //*

1. **determinare\_id\_student**(CNP VARCHAR(25))

Selecteaza id-ul studentului cu acel cnp

1. **vizualizare\_activitati\_curente\_AS\_student** (CNP VARCHAR(25), stud\_id INT)

Vede activitatile curente.

*DELIMITER //*

*CREATE PROCEDURE vizualizare\_activitati\_curente\_AS\_student (CNP VARCHAR(25), stud\_id INT)*

*BEGIN*

*SELECT cursuri.descriere, calendar.activitate , utilizator.nume, utilizator.prenume, calendar.zi, calendar.ora\_inceput, calendar.ora\_incheiere, calendar.data\_inceput, calendar.data\_incheiere*

*from inscriere\_curs*

*join cursuri on inscriere\_curs.id\_curs = cursuri.id\_curs*

*join calendar on cursuri.id\_curs = calendar.id\_curs*

*join profesor on calendar.id\_profesor = profesor.id\_profesor*

*join utilizator on profesor.id\_utilizator = utilizator.utilizator\_id*

*where calendar.data\_inceput <= current\_date and calendar.data\_incheiere >= current\_date*

*and inscriere\_curs.id\_student = stud\_id;*

*END; //*

1. **vizualizare\_activitati\_viitoare\_AS\_student** (CNP VARCHAR(25), stud\_id INT)

Vede activitatile viitoare.

1. **vizualizare\_activitati\_zi\_curenta\_AS\_**student (CNP VARCHAR(25), stud\_id INT)

Vede activitatile din acea zi

1. **parasire\_curs** (CNP varchar(25), nume\_curs varchar(25))

Un student care vrea sa paraseasca un curs

*DELIMITER //*

*CREATE PROCEDURE parasire\_curs (CNP varchar(25), nume\_curs varchar(25))*

*begin*

*set @id := null;*

*select @id:=utilizator\_id from utilizator where CNP = utilizator.CNP;*

*set @id\_stud := null;*

*select @id\_stud:=student.id\_student, @id\_curs:=cursuri.id\_curs from student, inscriere\_curs, cursuri*

*where student.id\_student = inscriere\_curs.id\_student*

*and @id = student.id\_utilizator*

*and inscriere\_curs.id\_curs = cursuri.id\_curs*

*and cursuri.descriere = nume\_curs;*

*delete from inscriere\_curs where id\_student = @id\_stud and id\_curs = @id\_curs;*

*delete from note WHERE id\_curs = @id\_curs and id\_student = @id\_stud;*

*UPDATE cursuri SET numar\_studenti\_inscrisi = numar\_studenti\_inscrisi - 1 WHERE cursuri.id\_curs = @id\_curs;*

*end; //*

1. **scrie\_mesaje** (mesaj VARCHAR(100), id\_student INT, nume VARCHAR(25))

Studentii pot sa scrie mesaje in grupul de studiu

1. **afiseaza\_mesaje** (nume\_grupa VARCHAR(25))

Se afiseaza mesajele scrise

1. **vizualizare\_membrii\_grup** (nume\_grup VARCHAR(25))

Se vad membrii unei grupe de studiu

1. **inscriere\_grup(**nume VARCHAR(25), CNP VARCHAR(25))

Un student se poate inscrie la un grup de studio

*DELIMITER //*

*CREATE PROCEDURE inscriere\_grup(nume VARCHAR(25), CNP VARCHAR(25))*

*BEGIN*

*SET @id\_gr := NULL;*

*SELECT @id\_gr:=grupa\_studiu.ID\_grupa from grupa\_studiu WHERE nume\_grup = nume;*

*SET @id\_stud := NULL;*

*SELECT @id\_stud:=id\_student from student, utilizator WHERE student.id\_utilizator = utilizator.utilizator\_id and utilizator.CNP = CNP;*

*insert into membrii\_grupa(id\_grupa, id\_student) VALUES (@id\_gr, @id\_stud);*

*END; //*

1. **parasire\_grup** (id\_student INT, nume varchar(25))

Un student poate parasi un grup de studiu

1. **adaugare\_activitate\_grup**(ID\_grupa int, denumire varchar(25), data\_organizare DATE, ora VARCHAR(25), numar\_minim\_participanti INT, durata\_expirare INT, nume\_prof VARCHAR(25))

Se adauga activitati in grupul de studiu

1. **inscriere\_activitate\_grup**(id int)

Se incrementeza numarul de student inscrisi.

1. **sugestii\_grup**(nume VARCHAR(25))

Apar sugestii de grupuri.

*DELIMITER //*

*CREATE PROCEDURE sugestii\_grup(nume VARCHAR(25))*

*begin*

*Set @id\_curs:=(select ID\_curs from cursuri where descriere=nume);*

*SELECT utilizator.nume , utilizator.prenume*

*FROM utilizator*

*INNER JOIN student ON utilizator.utilizator\_id = student.id\_utilizator*

*WHERE student.ID\_student in (select ID\_student from inscriere\_curs where @id\_curs=ID\_curs)*

*and student.ID\_student not in (select ID\_student from membrii\_grupa*

*inner join grupa\_studiu on (materie\_ID=@id\_curs and grupa\_studiu.ID\_grupa=membrii\_grupa.ID\_grupa));*

*end; //*

* **View-uri:**

*CREATE VIEW vizualizare\_note AS*

*SELECT utilizator.nume, utilizator.prenume, cursuri.descriere, note.nota\_curs , note.nota\_seminar, note.nota\_laborator*

*FROM utilizator*

*JOIN student ON utilizator.utilizator\_id = student.id\_utilizator*

*JOIN inscriere\_curs ON student.id\_student = inscriere\_curs.id\_student*

*JOIN cursuri ON inscriere\_curs.id\_curs = cursuri.id\_curs*

*JOIN note ON cursuri.id\_curs = note.id\_curs*

*GROUP BY utilizator.utilizator\_id;*

*CREATE VIEW vizualizare\_grupuri AS*

*SELECT grupa\_studiu.nume\_grup*

*FROM grupa\_studiu*

*GROUP BY nume\_grup;*

*CREATE VIEW vizualizare\_mesaje AS*

*SELECT mesaje\_grup.mesaj*

*FROM mesaje\_grup;*

* **Event:**

*DELIMITER //*

*CREATE EVENT verificare\_expirare\_activitate*

*ON SCHEDULE AT CURRENT\_TIMESTAMP + INTERVAL 1 MINUTE*

*DO*

*BEGIN*

*SET @id\_act:=(select id\_activitate from activitate where (TIMEDIFF(current\_time,activitate.data\_creare) > activitate.durata\_expirare) and activitate.numar\_studenti\_inscrisi < activitate.numar\_minim\_participanti);*

*SET @id\_gr:=(select id\_grupa from activitate where id\_activitate=@id\_act);*

*DELETE from activitate where id\_activitate = @id\_act;*

*INSERT INTO mesaje\_grup (mesaj, id\_grupa) values ("Activitate anulata", @id\_gr);*

*END //*

Se afiseaza un mesaj de “Activiate anulata”,tururor studentiilor inscrisi intr-un curs daca nu sunt suficienti pentru a tine activitatea.

## 3.4.Normalizarea datelor

Definitia formei normale Boyce-Codd (FNBC) este: Fie R o schemă de relație si F multimea de dependențe funcționale asociată. Se spune ca R este în forma normală Boyce-Codd dacă și numai dacă oricare ar fi o dependență netrivială X →Y din F , atunci X este supercheie pentru R.

Baza de date respectă aceasta definitie. Atributele fiecărui tabel nu depind de alte atribute. Fiecare tabel are o singură cheie primară după care sunt identificate înregistrările și este suficientă pentru a identifica în mod unic orice înregistrare din baza de date. În fiecare tabel avem doar o cheie și toate dependențele au în partea stângă o supercheie.

De exemplu,pentru tabela utilizator:

Un set complet de dependente functionale:

F={utilizator\_id->CNP

utilizator\_id->nume

utilizator\_id->prenume

utilizator\_id->adresa

utilizator\_id->IBAN

utilizator\_id->numar\_telefon

utilizator\_id->numar\_contract

utilizator\_id->email}

Datorita reuniunii=>utilizator\_id->CNP,nume,prenume,adresa,IBAN,numar\_telefon,

numar\_contract,email

Deci utilizator\_id este supercheie. Aceasta se afla in partea stanga a dependentelor funcionale deci se afla in FNBC.

Pentru tabela profesor:

Un set complet de depandenta functionale:

F={id\_profesor->utilizator\_id

Id\_profesor->numar\_minim\_ore

Id\_profesor->numar\_maxim\_ore

Id\_profesor->departamentul

Id\_profesor->numar\_total\_studenti}

Datorita reuniunii =>id\_profrsor->id\_utilizator,numar\_minim\_ore,numar\_maxim\_ore,

departamentul ,numar\_total\_studenti.Deci Id\_profesor este supercheie.Iar acest table se afla in FNBC.

Pentru tabela administrator:

F={id-adminisatrtor->id\_utilizator

Id\_adminisatrator->super\_administrator}

Datorita reuniunii=>id\_adminisatrtor->id\_utilizator,super\_adminisatrator.Deci id\_adminisatrtor este supercheie,si se afla in FNBC.

## 3.5. Interogări în algebra relațională

-- folosita la procedura cautare\_cursuri pentru a selecta denumirea tuturor cursurilor

**SELECT descriere FROM cursuri;**

**π{descriere} σ{} (Cursuri)**

-- folosita la procedura viz\_inf\_as\_profesor pentru a selecta toate informatiile unui

-- profesor(dat ca parametru prin nume si prenume) din tabela de utilizator

**SELECT \* FROM utilizator WHERE utilizator.nume=nume AND utilizator.prenume=prenume;**

**π{\*} σ{utilizator.nume = nume**

**^ utilizator.prenume = prenume} (utilizator)**

-- folosita la procedura filtrare\_profesori pentru a selecta CN-ul, numele si prenumele

-- unui profesor dat ca parametru prin id\_utilizator

**SELECT CNP,nume,prenume FROM utilizator,profesor WHERE utilizator\_ID = ID\_utilizator;**

**π{CNP,nume,prenume} σ{utilizator\_ID = ID\_utilizator} (utilizator x profesor)**

-- folosita la procedura notare pentru a selecta id-ul studentului cu numele si prenumele date ca parametrii

**SELECT ID\_student FROM student**

**INNER JOIN utilizator ON utilizator.utilizator\_id = student.ID\_utilizator**

**AND utilizator.nume = num**

**AND utilizator.prenume = prenum;**

**π{ID\_student} σ{ utilizator.utilizator\_id = student.ID\_utilizator**

**^ utilizator.nume = num**

**^ utilizator.prenume = prenum } (utilizator) ⋈ student)**

-- folosita la procedura cauta\_studenti pentru a selecta numele si prenumele studentilor inscrisi la un anumit curs

**SELECT nume, prenume FROM utilizator, student, inscriere\_curs**

**WHERE utilizator.utilizator\_id = student.ID\_utilizator**

**AND student.ID\_student = inscriere\_curs.ID\_student**

**AND inscriere\_curs.ID\_curs = id;**

**π{nume, prenume} σ{utilizator.utilizator\_id = student.ID\_utilizator**

**^ student.ID\_student = inscriere\_curs.ID\_student**

**^ inscriere\_curs.ID\_curs = id} (utilizator x student x inscriere\_curs)**

-- folosita la inscriere\_curs pentru a selecta id-urile cursurilor care au descrierea nume(este un parametru dat procedurii) si

-- care au numarul de studenti inscrisi egal cu nr\_studenti(dat ca parametru)

**SELECT cursuri.ID\_curs from cursuri WHERE descriere = nume AND cursuri.numar\_studenti\_inscrisi = nr\_studenti;**

**π{ID\_curs} σ{descriere = nume**

**^ numar\_studenti\_inscrisi = nr\_studenti} (Cursuri)**

# **Detalii de implementare**

## 4.1. Structura claselor in Java

A picture containing text, monitor, indoor, screenshot

Description automatically generated

Acestea sunt clasele noaste din interfata grafica.Avem clase pentru logare,paginile principale pentru fiecare tip de utilizator.Iar apoi clase pentru Cursuri.Si taote clasele pentru functionalitatile necesare fiecarui utilizator.

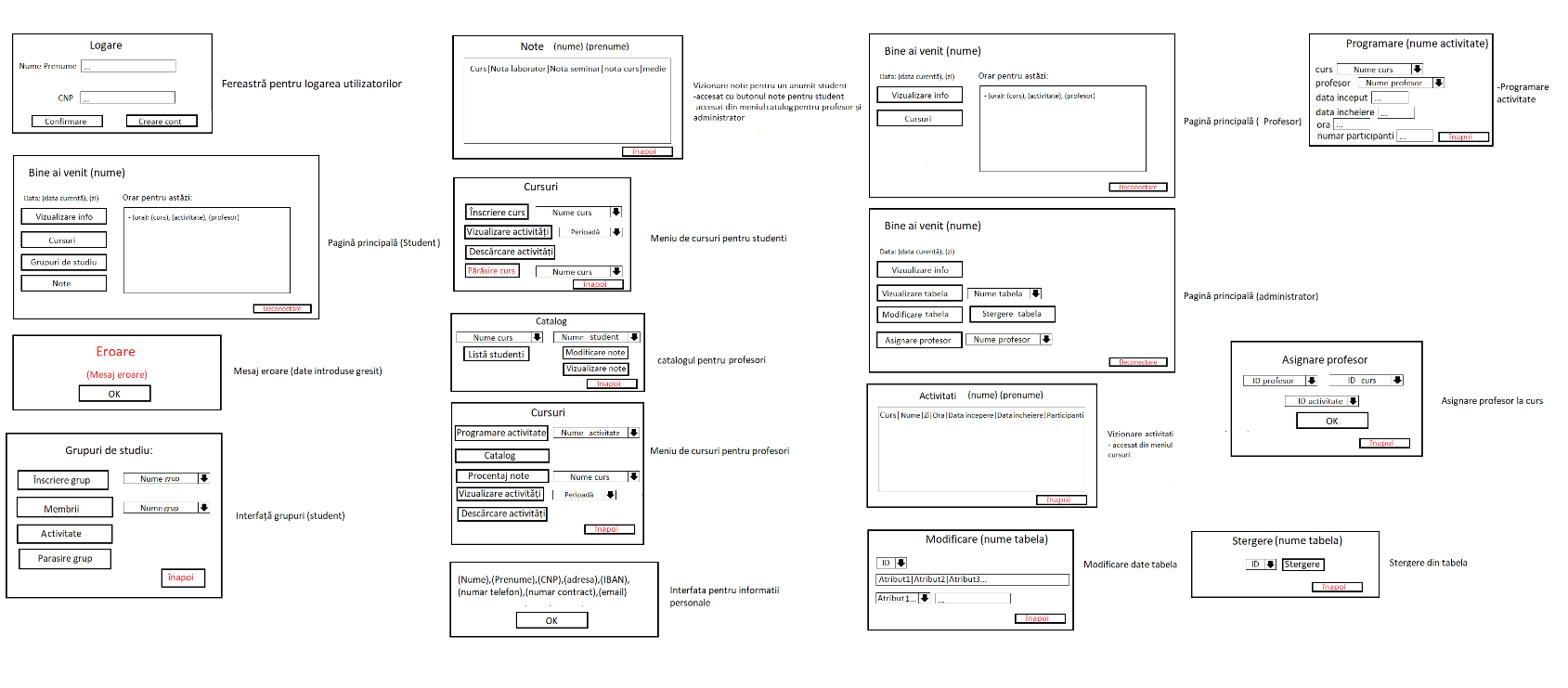
## 4.2. Manual de utilizare/instalare

La lansarea în execuție a aplicatiei suntem intampinati de fereastra de logare.Daca ne-am logat ca si un adminisatrtor,ni se deschide o pagina principla de unde putem sa executam toate functionalitatie unui adnimistartor:sa adauge,sa stearga utilizatori,sa asigneze un professor la un curs,sa vada profesorul de la un curs si studentii de la acel curs,sa vada informatii despre orice utilizator,sa vada toti studentii,profesorii si administratorii.

Daca ne-am logat ca si professor ne apare pagina principala pentru professor cu orarul pe ziua curenta,poate sa vada informatii despre el, sa adăuge si gestioneze activități,sa noteze studenți,sa gestioneze ponderile notelor,sa vizualizeze studenți înscriși la curs și sa descarcarce cataloage

Daca ne-am logat ca si student,ne apare orarul pe ziua curenta,poate sa vada informatii despre el,sa vizualizeze note,grupe studiu,activitati curente/viitoare,sa se inscrie la curs,sa se inscrie in / parasire grup de studiu.

* Pentru conectarea ca super-adminisatrator:Pop Vasile,CNP: 6001202018263
* Pentru conectarea ca administrator:Moldovan Andrei,CNP: 5001202028293
* Pentru conectarea ca professor:Costin Ana,CNP: 5000105017666
* Pentru conectarea ca student:Vultur Ana,CNP: 6001202018293



## 4.3. Elemente de Securitate a aplicatiei

Funcția pricipală a sistemului de securitate al Sistemului de gestiune a bazelor de date MySQL este aceea de a autentifica și autoriza utilizatorii conectați pentru a accesa datele stocate. Autorizarea se referă la permisiunea de a rula interogări/modificări ale datelor precum SELECT, INSERT, UPDATE sau DELETE. O clasă aparte de privilegii se referă la drepturile de administrare și de interacțiune cu sistemul de operare. Termenul privilegiu denotă în general un drept al unui utilizator de a acționa într-un anumit fel asupra unui obiect al bazei de date.

Aplicatia prezinta un nivel de securitate in momentul conectarii la baza de date dar si in momentul logarii propriu zise.In aplicatia noasta depinde cu ce rol te conectezi,si astfel o a ti se deschida pagina corespunzatoare si nu o sa ai decat acele functionalitati.

# **5.Concluzii. Limitari si dezvoltati ulterioare**

Am implementat toate cerintele,si toate functionalitatile fiecarui tip de utilizator.Consideram ca este o aplicatie foarte buna si poate sa fie utilizata cu succes de catre o universitate.

Profesorii au toate functonalitatile necesare sa poata comunica cu studentii si sa isi programeze orarul,sa noteze studentii.

Studentii pot sa vada informatiile legate de ei ,sa se inscrie la cursuri sau la grupe de studiu.Chiar pot sa si comunice intre ei prin intermediul mesajelor din grupele de studiu.

Administratorii si super-administratorii pot sa opereze pe utilizatori.Si sa aiba acces la majoriatea informatiilor din baza de date.

Ca dezvoltare ulterioara am putea sa avem o admitere,si rezulatatul la admitere sa determine inserarea unui nou student in baza de date.Si la profesori,sa observam cate cursuri nu au profesori,si sa inseram in baza de date doar numarul necesar de profesori.Am mai putea adauga la student posibilitatea de a putea plati taxele online,taxele depinzand de cate examene au picate,si dupa platirea taxei apare posibilitatea de a fi notat din nou la cursul respectiv.