MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARECATEDRA CALCULATOARE

Proiect de Semestru
la disciplina
Introducere in Baze de Date

Administratie Facultate

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Cuprins

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

1.Introducere

Proiectul presupune dezvoltarea unei aplicatii care lucreaza cu baze de date pentru gestiunea facultatii UTCN. Scopul aplicatiei este simplificarea operatiilor cu baza de date prin oferirea unei interfete grafice pe care angajatii ,studentii si profesorii universitati sa o poata utiliza. Aplicatia ofera sprinjin atat pentru interogarea bazei de date cat si pentru manipularea acesteia.

Aplicatia vine ca raspuns la stocarea de informatii pentru gestiunea unei universitati corespunzator anului in care ne aflam deoarece informatiile se pot stoca si modifica mult mai usor virtual decat in registre, pe hartie.

Pentru dezvoltarea proiectului au fost folosite:

- MySQL Workbench 8.0 -pentru crearea bazei de date, popularea initiala, proceduri si triggere si pentru crearea diagramei UML a tabelelor
- Intellij –mediu de dezvoltare Java
- Intellij – mediu de dezvoltare JavaFx

2. Analiza cerintelor utilizatorului

2.1 Ipotezele universitatii

Aplicatia gestioneaza actiunile studentilor , profesorilor si angajatii dintr-o universitate, aceasta utilizeaza o baza de date care este supusa urmatoarelor cerinte:

- Exista patru tipuri de utilizatori : student , profesor , administrator , super-administrator
- Un utilizator este unic identificat prin Id-ul sau, acesta mai are CNP, nume, prenume, adresa, numar de telefon, email, cont IBAN, numarul de contract si parola cu care se logheaza in aplicatie
- aplicatia trebuie sa ofere și o funcționalitate pentru deautentificare
- Utilizatorul de tip administrator poate adauga, modifica si sterge informatii in baza de date , informații legate de utilizatori; va exista si un rol de super-administrator care poate opera inclusiv asupra utilizatorilor de tip administrator
- Pentru un utilizator de tip profesor se vor retine si cursurile predate, numarul minim si numarul maxim de ore pe care le poate preda si departamentul din care face parte.
- Pentru un utilizator de tip student se va retine si anul de studiu si numarul de ore pe care trebuie sa le sustina.

- Cursurile sunt predate de mai multi profesori și au una sau mai multe tipuri de activitati (curs, seminar,
- laborator), o descriere, si un numar maxim de studenti participant
- Studentii se pot inscrie la cursuri si sunt asignati profesorului cu cei mai putini studenti la data inscrierii
- Fiecare activitate se desfasoara recursiv intre doua date, pe o anumita perioada de timp
- La asignarea unui profesor la un curs se vor alege tipurile de activitati
- profesorul poate programa activitatile intr-un calendar, pe zile si ore, specificand si numarul maxim de participant
- Profesorii pot accesa un catalog, unde pot filtra studentii dupa cursuri si le pot adauga note
- studentii se pot inscrie in grupuri de studiu pentru o anumita materie, daca sunt inscrisi la materia respective
- Pe grup, studentii pot adauga activitati si sa defineasca un numar minim de participanti si o perioada in care ceilalti studenti pot sa anunte participarea. Daca numarul minim nu este atins, activitatea se anuleaza, iar studentii inscrisi la ea primesc un mesaj de informare

2.2. Determinarea si caracterizarea profilurilor de utilizator

- autentificare, deautentificare
- operatii utilizator de tip administrator / super-administrator
- implementare sistem de control al drepturilor de acces al utilizatorilor in cadrul sistemului
- informatic pe baza rolurilor
- cautare cursuri
- cautare utilizatori

Avem 2 tipuri de utilizatori fara drepturi de administrator:

1. Profesori, care au urmatoarele posibilitati in aplicatie si asupre bazei de date:
    - cautare curs
    - inscriere la curs
    - vizualizare note
    - vizualizare grupuri si membri
    - mesaje pe grup
    - vizualizare / descarcare activitati curente / din viitor
2. Studenti, care au urmatoarele posibilitati in aplicatie si asupre bazei de date:
    - adaugare si programare activitati

- gestionare ponderi note
- notare studenti
- vizualizare liste studenti si descarcare cataloage
- vizualizare / descarcare activitati curente / din viitor

3.Modelul de date si descrierea acestuia

3.1 Entitati si atributele lor

Tabelele:

Student – informatii despre studentii universitatii.

Atribute : _id,CNP,nume,prenume,adresa, nr_telefon, email, IBAN, nr_Contract, anStudiu, nrOre.

Se leaga direct de Conturi;

Profesor — informatii despre studentii universitatii.

Atribute:_id, CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract, nrMinOre, nrMaxOre, department.

Se leaga direct de Conturi;

Administrator – informatii despre administratorii universitatii (de regula acestia sunt angajati ai universitatii).

Atribute : _id, CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract, rank.

Se leaga direct de Conturi;

Activitate – informatii despre activitatile unui grup.

Atribute : _id, nume, nrMinParticipanti, termenLimita, idGrup, idProfesor, _data.

Se leaga direct de Grupuri si Participanti deoarece un grup poate avea mai multi participanti.

Activitate Didactica : informatii despre activitati;e didactice ale universitatii (Curs, laborator, seminar).

Atribute: _id, nume, idProfesor, tip, pondere, activitatedidacticacol, descriere, nrMaxStudenti

Se leaga direct de calendar deoarece fiecare activitate se desfasoara comform unui program.

Calendar – contine datele in care au loc activitatile didactice.

Atribute : _id, date, idActivitate
Se leaga direct de Activitate Didactica.

Catalog – notele la diferite activitati pentru diferiti student.
Atribute : _id, idProfesorAct, idStudentAct, idActivitate, nota, data.

Chat – Atribute : _id, date, idGrup.
Se leaga direct de Grup si Mesaje;

Conturi – contine informatile de logare ale utilizatorilor.
Atribute : _idCont, id, parola, restrictive, idUtilizator, idStudent, idProfesor, idAdministrator
Se leaga direct de cele trei tabele ce contin informatii despre utilizatori: Profesor, Student, Administrator.

Grupuri – Atribute :_id , idMaterie.
Se leaga direct de Activitati, lista Membri si Chat.

Lista Membri – Atribute : _id, idStudent, idGrup
Se leaga direct de Grup deoarece un Grup are mai multi membri.

Mesaje – Atribute : _id, mesaj, idStudent, idChat
Se leaga direct de Chat, deoarece in chat se scriu mai multe mesaje.

Participanti – Atribute : _id, idStudent, idActivitate
Se leaga direct de activitate;

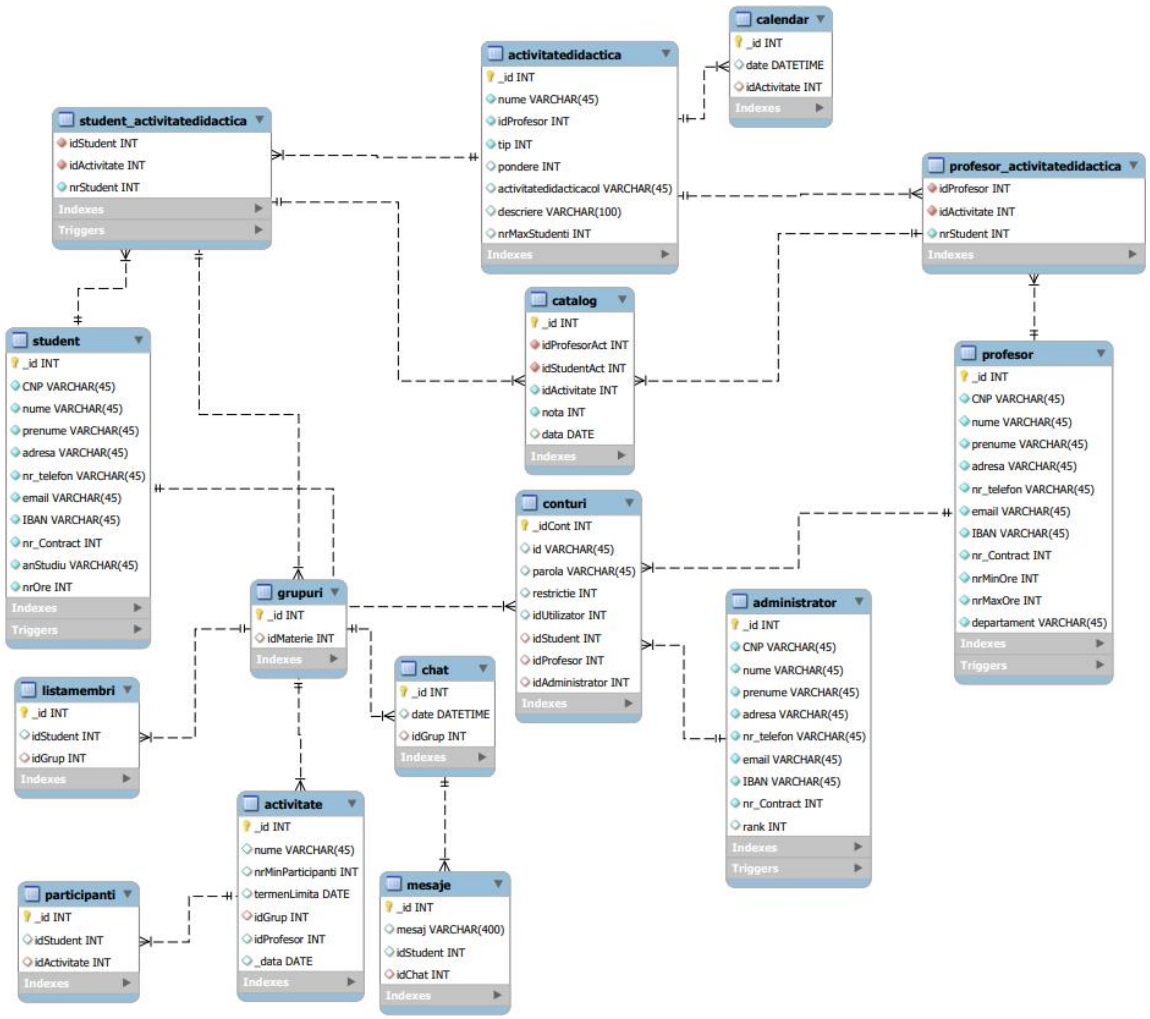
Profesor activitate didactica : idProfesor, idActivitate, nrStudent
Rezolva problema pentru mai multe legaturi many to may.

Student activitate didactica: idActivitate, nrStudent, nrStudent
Rezolva problema pentru mai multe legaturi many to may.

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

3.2 Diagrama UML pentru modelul de date complet



3.3. Proceduri, triggere

Proceduri:

1. adauga_activitate (nume varchar(45), idProfesor int, tip int, pondere int, descriere varchar(45), nrMaxSrudenti int, dataStart datetime, dataEnd datetime)
   Adauga activitate.
   Nume – numele activitatii
   idProfesor- id-ul Profesorului
   tip – tipul activitatii: Curs – 1, laborator – 2, seminar – 3
   pondere – ponderea activitatii
   descriere – o descriere a acttivitatii (aceasta poate sa lipseasca)
   nrMaxStudenti – numar maxim de stundenti care se pot inscribe la activitatea respective
   dataStart- prima data in calendar
   dataEnd – data inainte de care trebuie sa fie planificata ultima activitate

2. adaugare_nota (idProfesor int, idStudent int, idActivitate int, nota int)
   Adauga nota
   idProfesor – id-ul profesorului care adauga nota
   idActivitate – id-ul activitatii la care se adauga nota
   idStudent – id-u studentului la care se adauga nota
   nota – nota ce va fi adaugata

3. addDateInCalendar (idActiv int , dateStart datetime, dateEnd datetime)
   Adauga o planificare pentru o activitate cu id_ul idActivitate in calendar incepand cu dataStart in mod recursiv, din 7 in 7 zile, la aceeasi ora, pana la ultima data mai mica decat dataEnd

4. giveUpCourse (nameActiv varchar(45) , idStud int)
   Sterge studentul cu id-ul idStud de la toate activitatile didactice cu numele nameActiv. (implicit si din restul tabelelor care deriva de aici)

5. giveUpGroup(nameActiv varchar(45) , idStud int)
   Sterge studentul cu id-ul idStud din toate grupurile activitatilor didactice cu numele nameActiv.

6. giveUpGroupActivity(nameActiv varchar(45) , idStud int)
   Sterge studentul cu id-ul idStud de la toate activitatile grupurilor activitatilor didactice cu numele nameActiv.

7. inscriereLaActivitate (idStud int, idActiv int, nrStudenti int)

Inscrie studebdul cu id-ul idStud la activitatea idActiv si seteaza numarul de student la activitatea respective cu nrStudenti

8. delete_cont(idStud int, tip int)
   Sterge contul pentru utilizatorul de tipul tip cu id-ul id

9. delete_student_from_student(idStud int)
   Sterge studentul cu id_ul idStud din tabela Stud

10. schimba_pondere
    Seteaza ponderea activitatii idActiv la valoarea pond

11. schimbare_nota nota (idProfesor int, idStudent int, idActivitate int, nota int)
    Schimba nota
    idProfesor – id-ul profesorului care schimba nota
    idActivitate – id-ul activitatii la care se schimba nota
    idStudent – id-u studentului la care se schimba nota
    nota – nota ce va fi schimbata

12. setNrContractAdmin(id int, newNrContract int
    Seteaza valoarea contractului la newnrContract pt administratorul cu id-ul id

13. setNrContractProf (id int, newNrContract int)
    Seteaza valoarea contractului la newnrContract pt profesorul cu id-ul id

14. setNrContractStud (id int, newNrContract int)
    Seteaza valoarea contractului la newnrContract pt studentul cu id-ul id

15. setPasswordProcedure (new_parola varchar(45), idCont int)
    Seteaza parola pentru contul cu id-ul idCont

16. signInAdministratorProcedure (CNP varchar(45), nume varchar(45), prenume varchar(45), adresa varchar(45), nr_telefon varchar(45), email varchar(45), IBAN varchar(45), parola varchar(45))
    Inregistreaza un nou administrator

17. signInProfesorProcedure (CNP varchar(45), nume varchar(45), prenume varchar(45), adresa varchar(45), nr_telefon varchar(45), email varchar(45), IBAN varchar(45), nrMinOre int, nrMaxOre int, departament varchar(45), parola varchar(45))
    Inregistreaza un nou profesor

18. signInStudentProcedure (CNP varchar(45), nume varchar(45), prenume varchar(45), adresa varchar(45), nr_telefon varchar(45), email varchar(45), IBAN varchar(45), parola varchar(45))
    Inregistreaza un nou student

Triggere:

- insertIntoConturiAfterinsertIntoStudent – creaza cont pentru un student nou
- insertIntoConturiAfterinsertIntoProfesor– creaza cont pentru un profesor nou
- insertIntoConturiAfterinsertIntoAdministrator– creaza cont pentru un administrator nou
- decrementProfessorNrStud – decrementeaza numarul studentilor de la o anumita activitate didactica dupa ce un student renunta la aceasta activitate

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. 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 (cheia primară a tabelei). De exemplu, pentru tabela Student avem cheia primară _id și toate dependențele au în stânga supercheia idStudent, în tabelul Profesor această supercheie este idProfesor etc.

3.5 Interogari MySQL

```
       public static final String QUERY_CONTURI = "SELECT * FROM " +
       TABLE_CONTURI + " WHERE " + COLUMN_USERNAME + " = ? AND " +
COLUMN_PASSWORD + " = ?";
```

Utilizat pentru operatia de login.

```
public static final String QUERY_VALIDITY_DATE_ACTIVITY =
"SELECT * FROM " + TABLE_STUDENT_ACTIVITATE_DIDACTICA
        + " INNER JOIN "+ TABLE_ACTIVITATE_DIDACTICA
        + " ON " + TABLE_STUDENT_ACTIVITATE_DIDACTICA + "." +
COLUMN_ID_ACTIVITATE + " = " + TABLE_ACTIVITATE_DIDACTICA + "."
+ COLUMN_ID
        + " INNER JOIN " + TABLE_CALENDAR + " ON " +
TABLE_ACTIVITATE_DIDACTICA + "." + COLUMN_ID + " = " +
TABLE_CALENDAR + "." + COLUMN_ID_ACTIVITATE
        + " WHERE " + TABLE_STUDENT_ACTIVITATE_DIDACTICA + "." +
COLUMN_ID_STUDENT + " = ? AND " + TABLE_CALENDAR + "." +
COLUMN_ID_ACTIVITATE + " <> ? AND "
        + TABLE_CALENDAR + "." + COLUMN_DATE + " = ANY(SELECT "
+ TABLE_CALENDAR + "." + COLUMN_DATE + " FROM " + TABLE_CALENDAR
        + " WHERE " + TABLE_CALENDAR + "." +
COLUMN_ID_ACTIVITATE + " = ?)";
```

Folosit pentru a determina daca un student se poate inscrie la un curs din punct de vedere al orarului pe care il are inainte.

```
public static final String QUERY_MATERI_BY_PROFESOR = "SELECT *
FROM " + TABLE_ACTIVITATE_DIDACTICA +
        " INNER JOIN " + TABLE_PROFESOR_ACTIVITATEDIDACTICA + "
ON "+ TABLE_ACTIVITATE_DIDACTICA + "." + COLUMN_ID + " = " +
TABLE_PROFESOR_ACTIVITATEDIDACTICA + "." + COLUMN_ID_ACTIVITATE
+
        " INNER JOIN " + TABLE_PROFESOR + " ON "+
TABLE_PROFESOR_ACTIVITATEDIDACTICA + "." + COLUMN_ID_PROFESOR +
" = " + TABLE_PROFESOR + "." + COLUMN_ID +
        " WHERE " + TABLE_PROFESOR + "." + COLUMN_NUME + " LIKE
? ";
```

Folosit pentru a determina materile unui profesor.

```
public static final String QUERY_MATERI_BY_DATE = "SELECT " +
TABLE_ACTIVITATE_DIDACTICA + "." + COLUMN_ID + ", "
+TABLE_ACTIVITATE_DIDACTICA + "." + COLUMN_NUME +
        ", " + TABLE_ACTIVITATE_DIDACTICA + "." +
COLUMN_ID_PROFESOR + ", " + TABLE_CALENDAR + "." + COLUMN_DATE
+ " FROM " + TABLE_CALENDAR +
        " INNER JOIN " + TABLE_ACTIVITATE_DIDACTICA + " ON " +
TABLE_CALENDAR + "." + COLUMN_ID_ACTIVITATE + " = " +
TABLE_ACTIVITATE_DIDACTICA + "." + COLUMN_ID +
        " WHERE " + TABLE_CALENDAR + "." + COLUMN_DATE + " LIKE
?";
```

Folosit pentru a determina materii in functie de data.

```
public static final String QUERY_MATERI_BY_STUDENT = "SELECT *
FROM " + TABLE_ACTIVITATE_DIDACTICA +
        " INNER JOIN " + TABLE_STUDENT_ACTIVITATE_DIDACTICA + "
ON "+ TABLE_ACTIVITATE_DIDACTICA + "." + COLUMN_ID + " = " +
TABLE_STUDENT_ACTIVITATE_DIDACTICA + "." + COLUMN_ID_ACTIVITATE
+
        " INNER JOIN " + TABLE_STUDENT + " ON "+
TABLE_STUDENT_ACTIVITATE_DIDACTICA + "." + COLUMN_ID_STUDENT + "
= " + TABLE_STUDENT + "." + COLUMN_ID +
        " WHERE " + TABLE_STUDENT + "." + COLUMN_NUME + " LIKE ?
";
```

Folosit pentru a determina materii la care este inregistrat un anume student.

```
public static final String QUERY_AMINISTRATOR_BY_ID = "SELECT *
FROM " + TABLE_ADMINISTRATOR +
        " WHERE " + COLUMN_ID + " = ?";
```

Folosit pentru a determina datele unui administrator in functie de id.

```
public static final String QUERY_PROFESOR_BY_ID = "SELECT * FROM
" + TABLE_PROFESOR +
        " WHERE " + COLUMN_ID + " = ?";
```

Folosit pentru a determina datele unui profesor in functie de id.

```
public static final String QUERY_STUDENT_BY_ID = "SELECT * FROM
" + TABLE_STUDENT +
        " WHERE " + COLUMN_ID + " = ?";
```

Folosit pentru a determina datele unui student in functie de id.

```
public static final String QUERY_GRUP_BY_MATERIE = "SELECT " +
TABLE_GRUPURI + "." + COLUMN_ID + ", "  + " FROM " +
TABLE_GRUPURI +
        " WHERE " + TABLE_GRUPURI + "." + COLUMN_ID_MATERIE + "
= ? ";
```

Folosit pentru a determina grupuri in functie de idMaterie.

```
public static final String QUERY_CHAT_BY_GRUP = "SELECT " +
TABLE_MESAJE + "." + COLUMN_MESAJ + ", " + TABLE_MESAJE + "." +
COLUMN_ID_STUDENT + " FROM " + TABLE_MESAJE +
        " INNER JOIN " + TABLE_CHAT + " ON " + TABLE_MESAJE +
"." + COLUMN_ID_CHAT + " = " + TABLE_CHAT + "." + COLUMN_ID +
        " WHERE " + COLUMN_ID_CHAT + " = ?";
```

Folosit pentru a determina mesajele din chat ul unui grup.

3.6 Cod MySQL

3.6.1 Cod pentru crearea bazei de date si a tabelelor

- Schema mydb

```
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema administratiefacultate
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema administratiefacultate
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `administratiefacultate` DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `administratiefacultate` ;

-- -----------------------------------------------------
-- Table `administratiefacultate`.`student`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`student` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `CNP` VARCHAR(45) NOT NULL,
```

```
  `nume` VARCHAR(45) NOT NULL,
  `prenume` VARCHAR(45) NOT NULL,
  `adresa` VARCHAR(45) NOT NULL,
  `nr_telefon` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `IBAN` VARCHAR(45) NOT NULL,
  `nr_Contract` INT NOT NULL,
  `anStudiu` VARCHAR(45) NOT NULL,
  `nrOre` INT NOT NULL,
  PRIMARY KEY (`_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`activitatedidactica`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`activitatedidactica` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `nume` VARCHAR(45) NOT NULL,
  `idProfesor` INT NOT NULL,
  `tip` INT NOT NULL,
  `pondere` INT NULL DEFAULT NULL,
  `activitatedidacticacol` VARCHAR(45) NULL DEFAULT NULL,
  `descriere` VARCHAR(100) NULL DEFAULT NULL,
  `nrMaxStudenti` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`student_activitatedidactica`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`student_activitatedidactica` (
  `idStudent` INT NOT NULL,
  `idActivitate` INT NOT NULL,
  `nrStudent` INT NOT NULL,
  INDEX `idStudent` (`idStudent` ASC) VISIBLE,
```

```
  INDEX `idActivitate` (`idActivitate` ASC) VISIBLE,
  CONSTRAINT `student_activitatedidactica_ibfk_1`
    FOREIGN KEY (`idStudent`)
    REFERENCES `administratiefacultate`.`student` (`_id`),
  CONSTRAINT `student_activitatedidactica_ibfk_2`
    FOREIGN KEY (`idActivitate`)
    REFERENCES `administratiefacultate`.`activitatedidactica` (`_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`grupuri`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`grupuri` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `idMaterie` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_id`),
  UNIQUE INDEX `idMaterie_UNIQUE` (`idMaterie` ASC) VISIBLE,
  CONSTRAINT `grupuri_ibfk_1`
    FOREIGN KEY (`idMaterie`)
    REFERENCES `administratiefacultate`.`student_activitatedidactica` (`idActivitate`))
ENGINE = InnoDB
AUTO_INCREMENT = 6
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`activitate`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`activitate` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `nume` VARCHAR(45) NULL DEFAULT NULL,
  `nrMinParticipanti` INT NULL DEFAULT NULL,
  `termenLimita` DATE NULL DEFAULT NULL,
  `idGrup` INT NULL DEFAULT NULL,
  `idProfesor` INT NULL DEFAULT NULL,
  `_data` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`_id`),
  INDEX `_id_idx` (`idGrup` ASC) VISIBLE,
  CONSTRAINT `activitate_ibfk_1`
```

```
    FOREIGN KEY (`idGrup`)
    REFERENCES `administratiefacultate`.`grupuri` (`_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`administrator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`administrator` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `CNP` VARCHAR(45) NOT NULL,
  `nume` VARCHAR(45) NOT NULL,
  `prenume` VARCHAR(45) NOT NULL,
  `adresa` VARCHAR(45) NOT NULL,
  `nr_telefon` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `IBAN` VARCHAR(45) NOT NULL,
  `nr_Contract` INT NOT NULL,
  `rank` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`calendar`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`calendar` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `date` DATETIME NULL DEFAULT NULL,
  `idActivitate` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_id`),
  INDEX `_id_idx` (`idActivitate` ASC) VISIBLE,
  CONSTRAINT `calendar_ibfk_1`
    FOREIGN KEY (`idActivitate`)
    REFERENCES `administratiefacultate`.`activitatedidactica` (`_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 157
DEFAULT CHARACTER SET = utf8mb4
```

```
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`profesor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`profesor` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `CNP` VARCHAR(45) NOT NULL,
  `nume` VARCHAR(45) NOT NULL,
  `prenume` VARCHAR(45) NOT NULL,
  `adresa` VARCHAR(45) NOT NULL,
  `nr_telefon` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `IBAN` VARCHAR(45) NOT NULL,
  `nr_Contract` INT NOT NULL,
  `nrMinOre` INT NOT NULL,
  `nrMaxOre` INT NOT NULL,
  `departament` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 11
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`profesor_activitatedidactica`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`profesor_activitatedidactica` (
  `idProfesor` INT NOT NULL,
  `idActivitate` INT NOT NULL,
  `nrStudent` INT NOT NULL,
  INDEX `idProfesor` (`idProfesor` ASC) VISIBLE,
  INDEX `idActivitate` (`idActivitate` ASC) VISIBLE,
  CONSTRAINT `profesor_activitatedidactica_ibfk_1`
    FOREIGN KEY (`idProfesor`)
    REFERENCES `administratiefacultate`.`profesor` (`_id`),
  CONSTRAINT `profesor_activitatedidactica_ibfk_2`
    FOREIGN KEY (`idActivitate`)
    REFERENCES `administratiefacultate`.`activitatedidactica` (`_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
```

```
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`catalog`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`catalog` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `idProfesorAct` INT NOT NULL,
  `idStudentAct` INT NOT NULL,
  `idActivitate` INT NOT NULL,
  `nota` INT NOT NULL,
  `data` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`_id`),
  INDEX `idProfesor_idx` (`idProfesorAct` ASC) VISIBLE,
  INDEX `idStudent_idx` (`idStudentAct` ASC) VISIBLE,
  CONSTRAINT `idProfesor`
    FOREIGN KEY (`idProfesorAct`)
    REFERENCES `administratiefacultate`.`profesor_activitatedidactica` (`idProfesor`),
  CONSTRAINT `idStudent`
    FOREIGN KEY (`idStudentAct`)
    REFERENCES `administratiefacultate`.`student_activitatedidactica` (`idStudent`))
ENGINE = InnoDB
AUTO_INCREMENT = 41
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`chat`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`chat` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `date` DATETIME NULL DEFAULT NULL,
  `idGrup` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_id`),
  INDEX `_id_idx` (`idGrup` ASC) VISIBLE,
  CONSTRAINT `_id`
    FOREIGN KEY (`idGrup`)
    REFERENCES `administratiefacultate`.`grupuri` (`_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 4
DEFAULT CHARACTER SET = utf8mb4
```

```
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`conturi`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`conturi` (
  `_idCont` INT NOT NULL AUTO_INCREMENT,
  `id` VARCHAR(45) NULL DEFAULT NULL,
  `parola` VARCHAR(45) NULL DEFAULT NULL,
  `restrictie` INT NULL DEFAULT NULL,
  `idUtilizator` INT NULL DEFAULT NULL,
  `idStudent` INT NULL DEFAULT NULL,
  `idProfesor` INT NULL DEFAULT NULL,
  `idAdministrator` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_idCont`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  INDEX `_id_idx` (`idStudent` ASC) VISIBLE,
  INDEX `_id_idx1` (`idProfesor` ASC) VISIBLE,
  INDEX `_id2_idx` (`idAdministrator` ASC) VISIBLE,
  CONSTRAINT `_idd`
    FOREIGN KEY (`idStudent`)
    REFERENCES `administratiefacultate`.`student` (`_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `_id1`
    FOREIGN KEY (`idProfesor`)
    REFERENCES `administratiefacultate`.`profesor` (`_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `_id2`
    FOREIGN KEY (`idAdministrator`)
    REFERENCES `administratiefacultate`.`administrator` (`_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB
AUTO_INCREMENT = 31
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`listamembri`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`listamembri` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `idStudent` INT NULL DEFAULT NULL,
  `idGrup` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_id`),
  INDEX `_id_idx` (`idGrup` ASC) VISIBLE,
  CONSTRAINT `listamembri_ibfk_1`
    FOREIGN KEY (`idGrup`)
    REFERENCES `administratiefacultate`.`grupuri` (`_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 12
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`mesaje`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`mesaje` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `mesaj` VARCHAR(400) NULL DEFAULT NULL,
  `idStudent` INT NULL DEFAULT NULL,
  `idChat` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_id`),
  INDEX `_id_idx` (`idChat` ASC) VISIBLE,
  CONSTRAINT `mesaje_ibfk_1`
    FOREIGN KEY (`idChat`)
    REFERENCES `administratiefacultate`.`chat` (`_id`))
ENGINE = InnoDB
AUTO_INCREMENT = 77
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `administratiefacultate`.`participanti`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `administratiefacultate`.`participanti` (
  `_id` INT NOT NULL AUTO_INCREMENT,
  `idStudent` INT NULL DEFAULT NULL,
  `idActivitate` INT NULL DEFAULT NULL,
  PRIMARY KEY (`_id`),
```

```
  INDEX `_id_idx` (`idActivitate` ASC) VISIBLE,
  CONSTRAINT `participanti_ibfk_1`
    FOREIGN KEY (`idActivitate`)
    REFERENCES `administratiefacultate`.`activitate` (`_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

3.6.2 Inserari date existente

```
use administratiefacultate;

-- populare baza de date

-- populare tabela student
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("1263650640303", "Avram", "Vasile","Zalau, str. Calea Clujului", "0722343767",
"Avaram.Vasile@utcluj.ro", "RO69RNCB1971959919719599", '1', '2', '30');
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("1433451243363", "Margin", "Ioan","Zalau, str. Simion Barnutiu", "0774611180",
"Margin.Ioan@utcluj.ro", "RO69RNCB21386417113864171", '2', '1', '30');
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("1592079592079", "Eminovici", "Cristian","Arad, str. Observatorului", "0785262130",
"Eminovici.Cristian@utcluj.ro", "RO69RNCBRO69RNCB138641713864171", '3', '3', '30');
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("1564384126561", "Rusu", "Ioan","Cluj-Napoca, str. Unirii", "0722343375",
"Rusu.Ioan@utcluj.ro", "RO69RNCB3779554937795549", '4', '1', '30');
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("1126561259275", "Pop", "Valentin","Cluj-Napoca, str. Simion Barnutiu",
"0745890609", "Pop.Valentin@utcluj.ro", "RO69RNCB4998369349983693", '5', '2', '30');
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("1858318210436", "Bulgarean", "Andrei","Zalau, str. 21 Decembrie", "0764942898",
"Bulgarean.Andrei@utcluj.ro", "RO69RNCB8235827382358273", '6', '1', '30');
```

```
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("2359730242894", "Pestean", "Camelia","Oradea, str. Primaverii", "0713343008",
"Pestean.Camelia@utcluj.ro", "RO69RNCB8223113582231135", '7', '3', '30');
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("2127894799285", "Ardelean", "Denisa","Iasi, str. Ciresilor", "0713343008",
"Ardelean.Denisa@utcluj.ro", "RO69RNCB6796563567965635", '8', '2', '30');
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("2940639318301", "Puscas", "Rodica","Craiova, str. Traian Vuia", "0721465009",
"Puscas.Rodica@utcluj.ro", "RO69RNCB6865715768657157", '9', '1', '30');
insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values ("2946577864080", "Margin", "Alina","Bucuresti, str. Simion Barnutiu", "0725106422",
"Margin.Alina@utcluj.ro", "RO69RNCB2510642225106422", '10', '2', '30');

-- populare tabela profesor
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values ("1291467040303", "Pojar", "Eduard","Craiova, str. Traian Vuia", "0722343767",
"Pojar.Eduard@utcluj.ro", "RO69RNCB1971959919719599", '11', '4', '35', "Matematica");
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN,
nr_Contract,nrMinOre, nrMaxOre, departament)
 values ("1731183228693", "Barnutiu", "Denis","Iasi, str. Ciresilor", "0722343767",
"Barnutiu.Denis@utcluj.ro", "RO69RNCB1971959919719599", '12', '18', '39', "Calcuatoare");
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values ("1263650640303", "Opris", "Nelu","Zalau, buevardul Mihai Viteazu", "0722343767",
"Opris.Nelu@utcluj.ro", "RO69RNCB1971959919719599", '13', '15', '21', "Fizica");
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values ("1960469142215", "Lingurar", "Ionut","Zalau, str. Calea Clujului", "0722343767",
"Lingurar.Ionut@utcluj.ro", "RO69RNCB1971959919719599", '14', '20', '34', "Calcuatoare");
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values ("1985849858411", "Jula", "Eusebiu","Arad, str. Observatorului", "0722343767",
"Jula.Eusebiu@utcluj.ro", "RO69RNCB1971959919719599", '15', '3', '38', "Calcuatoare");
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values ("1887307887307", "Ghiurco", "Vlad","Suceava, str. Calea Vinului", "0722343767",
"Ghiurco.Vlad@utcluj.ro", "RO69RNCB1971959919719599", '16', '8', '35', "Limbi straine");
```

```
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values ("2840774840774", "Barbanta", "Aexandra","Braila, str. Crinului", "0722343767",
"Barbanta.Aexandra@utcluj.ro", "RO69RNCB1971959919719599", '17', '7', '32',
"Matematica");
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values ("2697303573176", "Alexa", "Petruta","Constanta, str. Emil Cioran", "0722343767",
"Alexa.Petruta@utcluj.ro", "RO69RNCB1971959919719599", '18', '2', '20', "Calcuatoare");
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values ("2263650640303", "Dunca", "Daniela","Zimbor, str. Principala", "0722343767",
"Dunca.Daniela@utcluj.ro", "RO69RNCB1971959919719599", '19', '20', '39', "Calcuatoare");
insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN,
nr_Contract,nrMinOre, nrMaxOre, departament)
 values ("2640336758103", "Goia", "Ioana","Huedin str. Mesteacanului", "0722343767",
"Goia.Ioana@utcluj.ro", "RO69RNCB1971959919719599", '20', '17', '29', "Fizica");


 -- popuare tabela administrator
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1265979760303", "Matioc", "Bogdan","Craiova, str. Traian Vlad", "0721773120",
"Matioc.Bogdan@utcluj.ro", "RO69RNCB9090721959721959", '21');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1263650597976", "Tibrea", "Andrei","Arad, str. Traian Vuia", "0733742259",
"Tibrea.Andrei@utcluj.ro", "RO69RNCB6156153536135361", '22');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1704905704905", "Pop", "Marius","Timisoara, str. Tineretului", "0711487804",
"Pop.Marius@utcluj.ro", "RO69RNCB7303973039623623", '23');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1387132387132", "Sigoiu", "Razvan","Satu Marea, str. Odorheiului", "0782884551",
"Sigoiu.Razvan@utcluj.ro", "RO69RNCB6239623963496349", '24');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1260303519601", "Madar", "George","Baia Mare, str. Traian Vuia", "0775825201",
"Madar.Razvan@utcluj.ro", "RO69RNCB5661545661547979", '25');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1266990266990", "Patac", "Daniela","Carei, str. Ciucas", "0729633751",
"Patac.Daniela@utcluj.ro", "RO69RNCB8924389243772772", '26');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1363603240324", "Stoica", "Ramona","Craiova, str. Calin Popescu", "0789613718",
"Stoica.Ramona@utcluj.ro", "RO69RNCB2901429014022022", '27');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
```

```
 values ("1884608846033", "Lobont", "Mara","Tasnad, str. Garii", "0791473511",
"Lobont.Mara@utcluj.ro", "RO69RNCB4548863454886366", '28');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1499324499324", "Dunca", "Calin","Medias, str. Fabricii", "0755819495",
"Dunca.Calin@utcluj.ro", "RO69RNCB5473547319841984", '29');
insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract)
 values ("1966978473972", "Pojar", "Alina","Sibiu, str. Libertatii", "0794839629",
"Pojar.Alina@utcluj.ro", "RO69RNCB7483490474834904", '30');


 -- POPULARE TABELA activitatedidactica + profesor+activitateDidactica + calendar

call adauga_activitate("BD", 1, 3, 20, "", 30, "2021-01-04 10:00:00", "2021-07-04 10:00:00");
call adauga_activitate("BD", 1, 2, 40, "", 30, "2021-01-05 10:00:00", "2021-07-04 10:00:00");
call adauga_activitate("BD", 1, 1, 40, "", 30, "2021-01-06 10:00:00", "2021-07-04 10:00:00");
call adauga_activitate("POO", 6, 1, 50, "", 30, "2021-01-07 10:00:00", "2021-07-04 10:00:00");
call adauga_activitate("POO", 6, 2, 50, "", 30, "2021-01-08 10:00:00", "2021-07-04 10:00:00");
call adauga_activitate("AF", 7, 3, 20, "", 30, "2021-01-04 10:00:00", "2021-07-04 12:00:00");
call adauga_activitate("CAN", 3, 1, 70, "", 30, "2021-01-05 10:00:00", "2021-07-04 12:00:00");
call adauga_activitate("AM", 10, 1, 90, "", 30, "2021-01-06 10:00:00", "2021-07-04 12:00:00");
call adauga_activitate("PC", 5, 3, 10, "", 30, "2021-01-07 10:00:00", "2021-07-04 12:00:00");
call adauga_activitate("PC", 5, 2, 45, "", 30, "2021-01-04 10:00:00", "2021-07-04 14:00:00");


-- populare tabela student_activitatedidactica

call inscriereLaActivitate(1, 1, 1);
call inscriereLaActivitate(2, 5, 1);
call inscriereLaActivitate(2, 4, 1);
call inscriereLaActivitate(1, 2, 1);
call inscriereLaActivitate(6, 6, 1);
call inscriereLaActivitate(1, 3, 1);
call inscriereLaActivitate(7, 8, 1);
call inscriereLaActivitate(4, 7, 1);
call inscriereLaActivitate(5, 10, 1);
call inscriereLaActivitate(5, 9, 1);

-- populare tabela catalog
call adaugare_nota(1, 1, 1, 5);
call adaugare_nota(1, 1, 2, 7);
call adaugare_nota(1, 1, 3, 6);
call adaugare_nota(6, 2, 5, 10);
call adaugare_nota(6, 2, 4, 7);
```

```
call adaugare_nota(10, 7, 8, 5);

insert into grupuri(idMaterie) values(6);
insert into grupuri(idMaterie) values(7);
insert into grupuri(idMaterie) values(9);
insert into grupuri(idMaterie) values(4);
insert into grupuri(idMaterie) values(5);

insert into listamembri(idStudent, idGrup) values(6, 1);
insert into listamembri(idStudent, idGrup) values(4, 2);
insert into listamembri(idStudent, idGrup) values(5, 3);
insert into listamembri(idStudent, idGrup) values(2, 4);
insert into listamembri(idStudent, idGrup) values(2, 5);

insert into chat(date, idGrup) values(current_timestamp(), 1);
insert into chat(date, idGrup) values(current_timestamp(), 2);
insert into chat(date, idGrup) values(current_timestamp(), 3);
insert into chat(date, idGrup) values(current_timestamp(), 4);
insert into chat(date, idGrup) values(current_timestamp(), 5);

insert into mesaje(mesaj, idStudent, idChat) values("Hello", 6, 1);
insert into mesaje(mesaj, idStudent, idChat) values("Hello", 4, 2);
insert into mesaje(mesaj, idStudent, idChat) values("Hello", 5, 3);
insert into mesaje(mesaj, idStudent, idChat) values("Hello", 2, 4);
insert into mesaje(mesaj, idStudent, idChat) values("Hello", 2, 5);

insert into activitate(nume, nrMinParticipanti, termenLimita, idGrup, idProfesor, _data)
values("fizica", 3, "2021-01-23", 1, 7,  "2021-02-04");
insert into activitate(nume, nrMinParticipanti, termenLimita, idGrup, idProfesor, _data)
values("lectura", 12, "2021-01-13", 2, null,  "2021-02-04");
insert into activitate(nume, nrMinParticipanti, termenLimita, idGrup, idProfesor, _data)
values("anteprenoriat", 5, "2021-05-23", 3, null, "2021-06-04");
insert into activitate(nume, nrMinParticipanti, termenLimita, idGrup, idProfesor, _data)
values("volunatriat", 4, "2021-01-12", 4, 6,  "2021-02-04");
insert into activitate(nume, nrMinParticipanti, termenLimita, idGrup, idProfesor, _data)
values("competitie-sah", 3, "2021-01-17", 5, null, "2021-02-04");

insert into participanti(idStudent, idActivitate) values(6, 1);
insert into participanti(idStudent, idActivitate) values(4, 2);
insert into participanti(idStudent, idActivitate) values(5, 3);
insert into participanti(idStudent, idActivitate) values(2, 4);
insert into participanti(idStudent, idActivitate) values(2, 5);
```

3.6.3 Cod proceduri

```
-- -----------------------------------------------------
-- procedure adauga_activitate
-- -----------------------------------------------------

DELIMITER //
CREATE DEFINER=`root`@`localhost` PROCEDURE `adauga_activitate`(nume varchar(45), idProfesor int, tip int, pondere int, descriere varchar(45), nrMaxSrudenti int, dataStart datetime, dataEnd datetime)
BEGIN
	insert into activitatedidactica(nume, idProfesor, tip, pondere, descriere, nrMaxStudenti) values (nume, idProfesor, tip, pondere, descriere, nrMaxStudenti);
	select max(_id) into @newIdActivitate from activitatedidactica;
   insert into profesor_activitatedidactica(idProfesor, idActivitate, nrStudent) values(idProfesor, @newIdActivitate,0);
   call addDateInCalendar(@newIdActivitate, dataStart, dataEnd);
END
// DELIMITER ;

-- -----------------------------------------------------
-- procedure adaugare_nota
-- -----------------------------------------------------

DELIMITER $$
USE `administratiefacultate`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `adaugare_nota`(idProfesor int, idStudent int, idActivitate int, nota int)
BEGIN
	insert into catalog(idProfesorAct, idStudentAct, idActivitate, nota, data) values (idProfesor, idStudent, idActivitate, nota, current_date());
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure addDateInCalendar
-- -----------------------------------------------------

DELIMITER $$
USE `administratiefacultate`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `addDateInCalendar`(idActiv int , dateStart datetime, dateEnd datetime)
```

```
BEGIN
   declare dateInsert datetime default dateStart;
   r: while datediff(dateEnd, dateInsert) > 0 do
      insert into calendar(idActivitate, date) values (idActiv, dateInsert);
      set dateInsert = adddate(dateInsert, 7);
   end while r;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure schimba_pondere
-- -----------------------------------------------------

DELIMITER $$
USE `administratiefacultate`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `schimba_pondere`(idActivitate int,
pond int)
BEGIN
        update activitatedidactica set pondere = pond where _id = idActivitate;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure schimbare_nota
-- -----------------------------------------------------

DELIMITER $$
USE `administratiefacultate`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `schimbare_nota`(idProfesor int,
idStudent int, idActiv int, new_nota int)
BEGIN
        update catalog set nota = new_nota where idProfesorAct = idProfesor and idStudentAct =
idStudent and idActivitate = idActiv;
END$$


drop procedure if exists giveUpGroupActivity;
DELIMITER //
CREATE PROCEDURE giveUpGroupActivity(nameActiv varchar(45) , idStud int)
BEGIN
        declare id int ;
```

```
   select min(participanti._id)  into id from participanti, activitate
   where participanti.idStudent = idStud and participanti.idActivitate = activitate._id and
   activitate.idGrup = ANY( select grupuri._id from grupuri, activitatedidactica
   where grupuri.idMaterie = activitatedidactica._id and activitatedidactica.nume =
nameActiv);

   while id is not null do
                delete from participanti where _id = id;

                select min(participanti._id)  into id from participanti, activitate
                where participanti.idStudent = idStud and participanti.idActivitate =
activitate._id and
                activitate.idGrup = ANY( select grupuri._id from grupuri, activitatedidactica
                where grupuri.idMaterie = activitatedidactica._id and
activitatedidactica.nume = nameActiv);

   end while;
END;
// DELIMITER ;

drop procedure if exists giveUpGroup;
DELIMITER //
CREATE PROCEDURE giveUpGroup(nameActiv varchar(45) , idStud int)
BEGIN
        declare id int ;

        select min(listamembri._id)  into id from listamembri where listamembri.idStudent
= idStud and
   listamembri.idGrup = ANY( select grupuri._id from grupuri, activitatedidactica
   where grupuri.idMaterie = activitatedidactica._id and activitatedidactica.nume =
nameActiv);

   while id is not null do
                delete from listamembri where _id = id;

                select min(listamembri._id)  into id from listamembri where
listamembri.idStudent = idStud and
                listamembri.idGrup = ANY( select grupuri._id from grupuri,
activitatedidactica
                where grupuri.idMaterie = activitatedidactica._id and
activitatedidactica.nume = nameActiv);
```

```
    end while;

END;
// DELIMITER ;

drop procedure if exists giveUpCourse;
DELIMITER //
CREATE PROCEDURE giveUpCourse(nameActiv varchar(45) , idStud int)
BEGIN
        declare id int default null;

        call giveUpGroupActivity(nameActiv, idStud);
        call giveUpGroup(nameActiv, idStud);

    SET FOREIGN_KEY_CHECKS = 0;
        delete from student_activitatedidactica where idActivitate = any(select _id from
activitatedidactica where nume = nameActiv) and idStudent = idStud;
        update student_activitatedidactica set nrStudent = nrStudent - 1 where
idActivitate = any(select _id from activitatedidactica where nume = nameActiv);
    SET FOREIGN_KEY_CHECKS = 1;

END;
// DELIMITER ;


drop procedure if exists inscriereLaActivitate;
DELIMITER //
CREATE PROCEDURE inscriereLaActivitate(idStud int, idActiv int, nrStudenti int)
BEGIN
    insert into student_activitatedidactica values (idStud, idActiv, nrStudenti);
    update student_activitatedidactica set nrStudent = nrStudenti where idActivitate = idActiv;
        update profesor_activitatedidactica set nrStudent = nrStudenti where idActivitate =
idActiv;
END;
// DELIMITER ;


drop procedure if exists setNrContractAdmin;
DELIMITER //
CREATE  PROCEDURE `setNrContractAdmin`(id int, newNrContract int)
BEGIN
        update administrator set nr_Contract = newNrContract where _id = id;
END;
```

```
// DELIMITER ;


drop procedure if exists setNrContractProf;
DELIMITER //
CREATE PROCEDURE `setNrContractProf`(id int, newNrContract int)
BEGIN
	update profesor set nr_Contract = newNrContract where _id = id;
END;
// DELIMITER ;


drop procedure if exists setNrContractStud;
DELIMITER //
CREATE PROCEDURE `setNrContractStud`(id int, newNrContract int)
BEGIN
	update student set nr_Contract = newNrContract where _id = id;
END;
// DELIMITER ;


drop procedure if exists setPasswordProcedure;
DELIMITER //
CREATE PROCEDURE setPasswordProcedure(new_parola varchar(45), idCont int)
BEGIN
	update conturi set parola = new_parola where _idCont = idCont;
END;
// DELIMITER ;


drop procedure if exists signInStudentProcedure;
DELIMITER //
CREATE PROCEDURE `signInStudentProcedure`(CNP varchar(45), nume varchar(45),
prenume varchar(45), adresa varchar(45), nr_telefon varchar(45), email varchar(45), IBAN
varchar(45), parola varchar(45))
BEGIN
	insert into student(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
anStudiu, nrOre)
 values (CNP, nume, prenume, adresa, nr_telefon, email, IBAN, 0, 1, 0);
	select max(_id) into @id from student;
	select max(_idCont) into @cont from conturi;
   call setNrContractStud(@id, @cont);
	call setPasswordProcedure(parola, @cont);
```

```
END;
// DELIMITER ;


drop procedure if exists signInProfesorProcedure;
DELIMITER //
CREATE PROCEDURE `signInProfesorProcedure`(CNP varchar(45), nume varchar(45),
prenume varchar(45), adresa varchar(45), nr_telefon varchar(45), email varchar(45), IBAN
varchar(45), nrMinOre int, nrMaxOre int, departament varchar(45), parola varchar(45))
BEGIN
	insert into profesor(CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_Contract,
nrMinOre, nrMaxOre, departament)
 values (CNP, nume, prenume, adresa, nr_telefon, email, IBAN, 0, nrMinOre, nrMaxOre,
departament);
   select max(_id) into @id from profesor;
	select max(_idCont) into @cont from conturi;
    call setNrContractProf(@id, @cont);
	call setPasswordProcedure(parola, @cont);
END;
// DELIMITER ;

drop procedure if exists signInAdministratorProcedure;
DELIMITER //
CREATE PROCEDURE `signInAdministratorProcedure`(CNP varchar(45), nume varchar(45),
prenume varchar(45), adresa varchar(45), nr_telefon varchar(45), email varchar(45), IBAN
varchar(45), parola varchar(45))
BEGIN
	insert into administrator(CNP, nume, prenume, adresa, nr_telefon, email, IBAN,
nr_Contract)
 values (CNP, nume, prenume, adresa, nr_telefon, email, IBAN, 0);
	select max(_id) into @id from administrator;
	select max(_idCont) into @cont from conturi;
    call setNrContractAdmin(@id, @cont);
	call setPasswordProcedure(parola, @cont);
END;
// DELIMITER ;
```

3.6.4 Cod Triggere

```
DELIMITER $$
USE `administratiefacultate`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `administratiefacultate`.`insertIntoConturiAfterinsertIntoStudent`
AFTER INSERT ON `administratiefacultate`.`student`
FOR EACH ROW
begin
        -- if _id = new._id then
                select _id into @newidUtilizator from student where _id = new._id;
    select email into @new_id from student where _id = new._id;
    select nume into @newparola from student where _id = new._id;

    insert into conturi(id, parola, restrictie, idUtilizator) values(@new_id,  @newparola, 1,
@newidUtilizator);
        -- end if;
end$$

USE `administratiefacultate`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `administratiefacultate`.`insertIntoConturiAfterinsertIntoAdministrator`
AFTER INSERT ON `administratiefacultate`.`administrator`
FOR EACH ROW
begin
        -- if _id = new._id then
                select _id into @newidUtilizator from administrator where _id = new._id;
    select email into @new_id from administrator  where _id = new._id;
    select nume into @newparola from administrator  where _id = new._id;

    insert into conturi(id, parola, restrictie, idUtilizator) values(@new_id,  @newparola, 3,
@newidUtilizator);
        -- end if;
end$$

USE `administratiefacultate`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `administratiefacultate`.`insertIntoConturiAfterinsertIntoProfesor`
AFTER INSERT ON `administratiefacultate`.`profesor`
FOR EACH ROW
begin
```

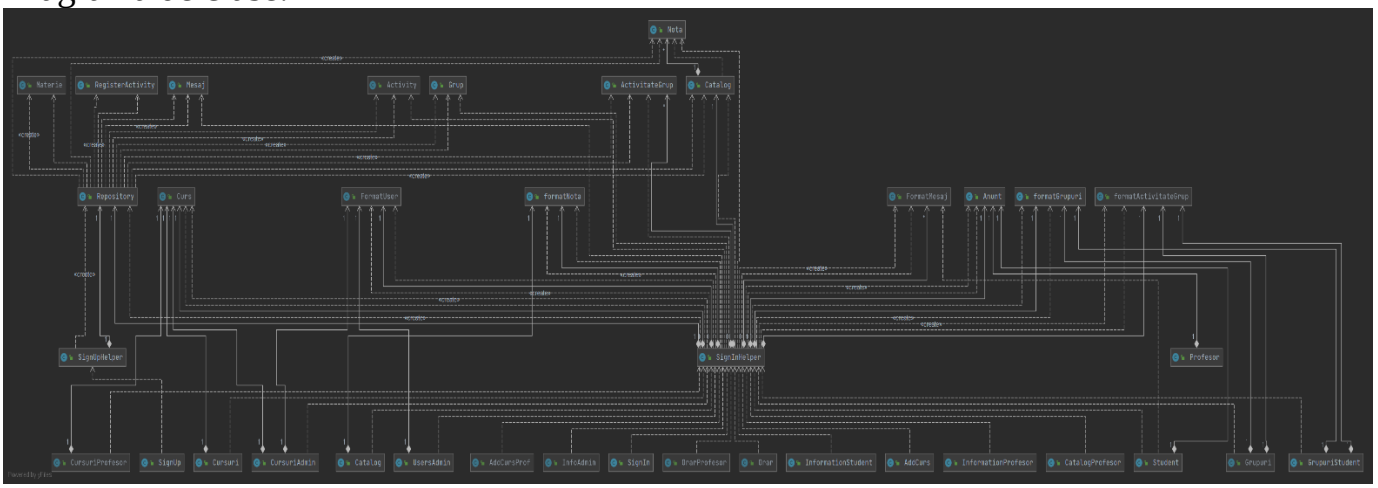
```
        -- if _id = new._id then
                select _id into @newidUtilizator from profesor where _id = new._id;
        select email into @new_id from profesor  where _id = new._id;
        select nume into @newparola from profesor  where _id = new._id;

        insert into conturi(id, parola, restrictie, idUtilizator) values(@new_id,  @newparola, 2,
@newidUtilizator);
            -- end if;
end$$
```

4. Detalii de implementare

4.1. Structura claselor in Java

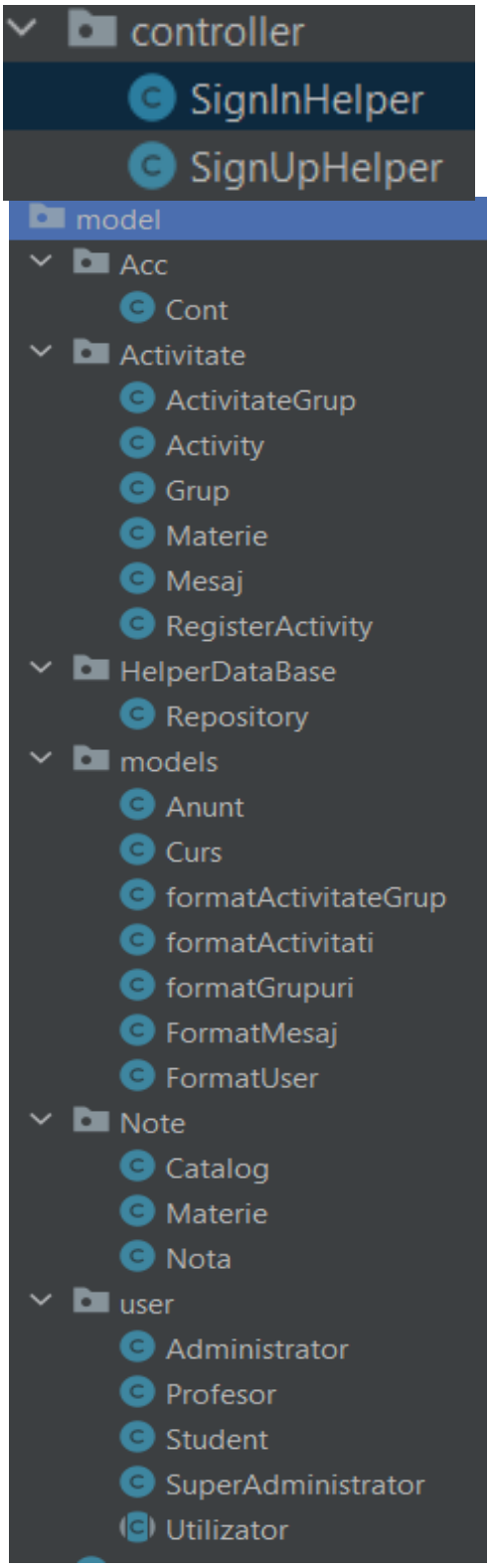
Diagrama de clase:



Clasele sunt impartite in 3 pachete: controller, model and sample si database. In pachetul sample sunt clasele de interfata grafica, frame-uri si dialog-uri. Pachetul controller contine clasele legate de logica de functionare a aplicatiei. Pachetul model contine alte 6 pachete: Acc, Activitate, HelperDataBase, models, Note, user. Pachetele Acc, Activitate, Note si user contine clase folosite pentru a prelua informatile din tabelele bazei de date. Pachetul models contine modelele de tabele ce trebuie afisate in aplicatie, iar pachetul HelperDataBase contine clasa Repository care contine logica de cocnectare la baza de date si logica de interogare si manipulare a datelor din baza de date.

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA



-controleaza logarea
-controleaza inregistratrea

-clasa pentru stocarea sub forma de obiect a informatilor din tabela Conturi

-clasa pentru stocarea sub forma de obiect a informatilor din tabela Activitati
-clasa pentru stocarea sub forma de obiect a informatilor din tabela Activitate didactica
-clasa pentru stocarea sub forma de obiect a informatilor din tabela Grup
-clasa pentru stocarea sub forma de obiect a informatilor din tabela Activitate Didactica
-clasa pentru stocarea sub forma de obiect a informatilor din tabela
-clasa pentru stocarea sub forma de obiect a informatilor din tabela Conturi

- contine logica de conectare la baza dedate si logica de operare cu baza de date

-clasa contine logica pentru notificarile ce vor fi afisate in interfata
-clasa contine logica pentru cursurile ce vor fi afisate in interfata
-clasa contine formatul pentru interfata pentru Activitati Grup
-clasa contine formatul pentru interfata pentru Activitati didactice
-clasa contine formatul pentru interfata pentru Mesaj
-clasa contine formatul pentru interfata pentru Grup
-clasa contine formatul pentru interfata pentru User

-clasa pentru stocarea sub forma de obiect a unei liste de note
-clasa pentru stocarea sub forma de obiect a informatilor din tabela Conturi
-clasa pentru stocarea sub forma de obiect a informatilor din tabela catalog

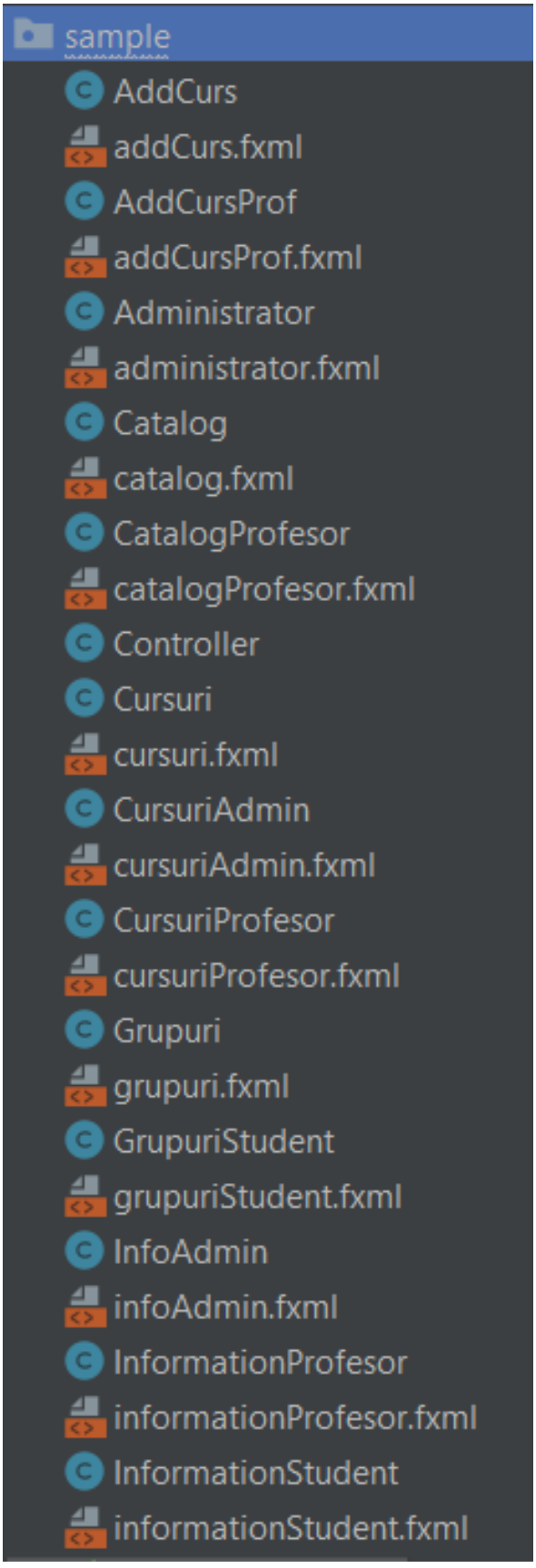
-clasa pentru stocarea sub forma de obiect a informatilor din tabela Administrator

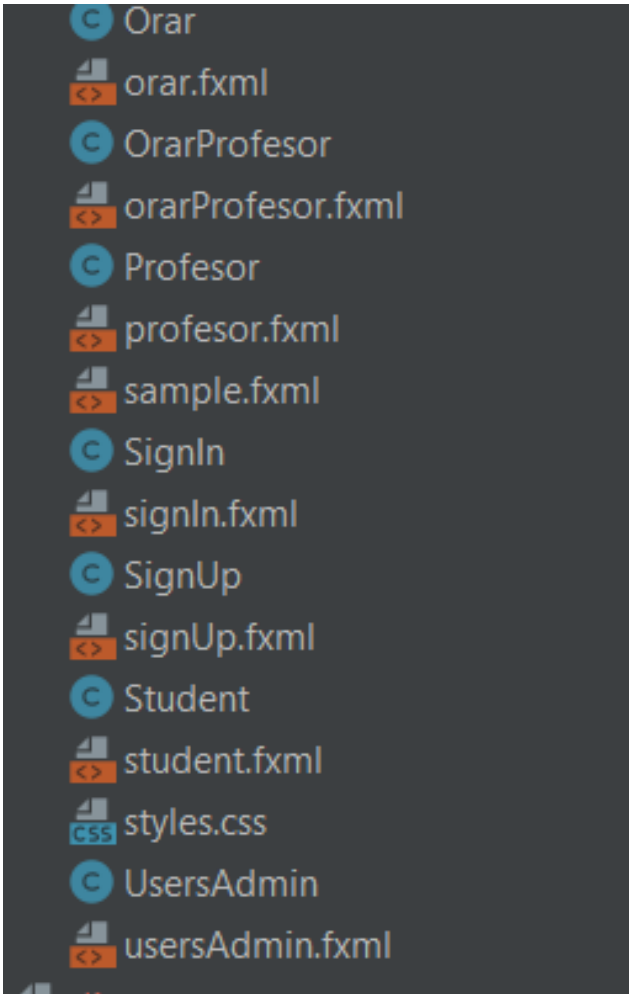
-clasa pentru stocarea sub forma de obiect a informatilor din tabela Profesor

-clasa pentru stocarea sub forma de obiect a informatilor din tabela Student

-clasa pentru stocarea sub forma de obiect a informatilor din tabela SuperAdministrator

-clasa abstracta parinte pentru celalte clase din pachetul user





- Aceste clase sunt folosite petru interfata grafica.

4.2. Manual de utilizare/instalare

La lansarea in executie a aplicatiei suntem intampinati de fereastra “Welcome” de unde putem
alege ca ce tip de utilizator vrem sa ne autentificam pentru a folosi aplicatia, daca alegem sa ne conectam
ca cititor si datele de logare sunt corecte, se va deschide fereastra principala a cititorului care permite
diferite operatii pe care un cititor le-ar putea face la o biblioteca. Daca alegem sa ne conectam ca bibliotecar iar datele de logare sunt corecte o fereastra corespunzatoare pentru un bibliotecar va fi
deschisa ar aceasta ii va permite acestuia diferie posibilitati de manipulare a datelor din baza de date.

Pentru conectarea ca student se pot folosi: “ Avaram.Vasile@utcluj.ro” si parola “Avram”.
Pentru conectarea ca profesor se pot folosi: “ Pojar.Eduard@utcluj.ro” cu parola “Pojar”.
Pentru conectarea ca profesor se pot folosi: “ Matioc.Bogdan@utcluj.ro” cu parola „Matioc”.
Pentru conectarea ca profesor se pot folosi: “ Sigoiu.Razvan@utcluj.ro” cu parola „Sigoiu”

5. Concluzii. Limitari si dezvoltati ulterioare

Toate cerintele au fost respecate si indeplinite, acestea ducand la un mediu placut prin care student si profesori pot interactiona cu o universitate modera si la un mod convenabil de gestiune a datelor de catreprofesori si restul angajatilor universitatii..
Aplicatia ofera suport deplin pentru clientii pentru a-si opera toate interactiunile cu universitatea. Aplicatia nu ofera control si acces deplin profesorilor asupra datelor, deoarece aceastea sunt oferite unui utilizator de tip administrator sau super-administrator, un utilizator care sa stie sa gestioneze cu atentie datele pentru a nu le altera.
Ca dezvoltare ulterioara mentionez dezvoltarea functilor pe grup, cum ar fi: posibilitatea creari unui apel video pentru a se sustine cursuri online sau incarcarea de fisiere.