

FACULTATEA DE AUTOMATICĂ ȘI
CALCULATOARE

DEPARTAMENTUL CALCULATOARE

PROIECT

la disciplina

Introducere in Baze de Date

Nume proiect: Platforma de studiu

Profesor coordonator:

Cosmina Ivan

Echipa de proiect:

Silviu Grumazescu

Tamas Dari

Andrei-Denis Alexandru

Grupa 30221, Seria A, CTI

An academic: 2021-2022

Cuprins

1. Introducere
 - 1.1. Introducere, argumente, scop si obiective specifice
2. Suportul teoretic
3. Analiza cerințelor utilizatorilor (Specificațiile de proiect)
 - 3.1. Ipotezele specifice domeniului ales pentru proiect (cerințe, constrângeri)
 - 3.2. Organizare structurata(tabelar) a cerințelor utilizator
 - 3.3. Determinarea si caracterizarea de profiluri de utilizatori
4. Modelul de date si descrierea acestuia
 - 4.1. Entități si attributele lor (descriere detaliata – **implementarea fizica**)
 - 4.2. Diagrama EER/UML pentru modelul de date complet
 - 4.3. Normalizarea datelor
5. Detalii de implementare MySQL
 - 5.1. DDL
 - 5.2. DML
 - 5.3. SQL programatic
6. Detalii de implementare Java
7. Utilizarea aplicației
8. Concluzii. Limitări si dezvoltări ulterioare
9. Bibliografie

Introducere

Secolul XXI, supranumit și secolul vitezei, a adus cu el dezvoltări exponențiale și, în unele cazuri, revoluționare, în domeniul tehnologiei informației și în domeniul comunicațiilor. Pe fondul schimbărilor rapide și progresului tehnologic înregistrat, precum și pe fondul tendinței de globalizare a educației universitare, s-au deschis noi perspective pentru sistemul educațional universitar, această fiind completat cu metode moderne de abordare a educației precum și legăturii cu studenții.

Globalizarea a adus și dezavantajul de a facilita răspândirea rapidă a bolilor și virusurilor, iar pandemia generată de virusul SARS-COV 2 a evidențiat nevoia de digitalizare a sistemelor universitare. Proiectul prezentat reprezintă o abordare minimalistă, cu foarte mult loc de dezvoltare ulterioară, a satisfacerii nevoii menționate anterior, și presupune dezvoltarea unei aplicații ce lucrează cu baze de date pentru managementul unei platforme de studiu. Aplicația oferă sprijin atât pentru interogarea bazei de date, cât și pentru manipularea informațiilor stocate în ea, sprijin oferit de interfața grafică creată cu scopul unei interacțiuni facile între utilizator și sistemul universitar.

Pentru dezvoltarea proiectului s-au folosit următoarele:

- Apache NetBeans - mediu de dezvoltare integrat pentru Java
- MySQL Workbench - tool folosit pentru crearea bazei de date, popularea inițială, crearea de proceduri și triggeri, crearea diagramei UML a tabelelor
- JDBC - pentru stabilirea conexiunii dintre aplicație și baza de date
- Apache POI - oferă biblioteci Java pure pentru citirea și scrierea fișierelor în formate Microsoft Office, cum ar fi Word, PowerPoint și **Excel**

Suportul teoretic

MySQL

MySQL este un sistem de gestionare a bazelor de date relaționale open-source care este utilizat în principal pentru aplicațiile online. MySQL poate crea și gestiona baze de date foarte utile (cum ar fi informații despre angajați, inventar și multe altele). Un SGBD relațional înseamnă că datele găzduite în structură sunt capabile să recunoască relațiile dintre informațiile stocate. Fiecare bază de date conține tabele. Fiecare tabel (denumit și o relație) conține una sau mai multe categorii de date stocate în coloane (denumite și attribute). Fiecare rând (denumit, de asemenea, o înregistrare sau „tuple”) conține o informație unică (altfel menționată ca și cheie) pentru categoriile definite în coloane.

Există un acronim popular în industria tehnologiei: LAMP și WAMP. Acest acronim înseamnă Linux/Windows Apache MySQL PHP și este unul dintre cele mai populare „web stack-uri” de pe planetă (un „stack” este o stivă de software care interacționează împreună).

Java

Java este un limbaj de programare OOP sau orientat-obiect, dezvoltat de James Gosling la Sun Microsystems (acum filială Oracle), la începutul anilor '90 și lansat în 1995. Limbajul împrumută o mare parte din sintaxa C și C++, dar are un model al obiectelor mai simplu. Java este un program în care se pot produce aplicații. După ce un programator dezvoltă o aplicație Java, aceasta poate rula pe majoritatea sistemelor de operare (OS), incluzând Windows, Linux și Mac OS, lucru ce face Java un limbaj versatil.

Este important de menționat că Java poate fi folosit pentru a dezvolta aplicații *complete*, care pot rula pe un singur computer sau care pot fi distribuite între servere și clienți într-o rețea. De asemenea, poate fi folosit pentru a programa miniaplicații sau *applets* care nu sunt independente, ci sunt parte a unei pagini web și facilitează interacțiunea utilizatorului cu interfața grafică.

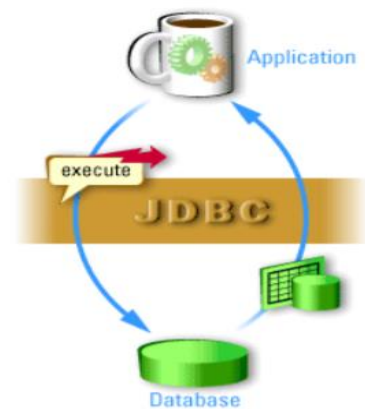
JDBC (Java Database Connectivity)

JDBC (Java Database Connectivity) este tehnologia Java de acces la baze de date relaționale. Este independentă de tipul bazei de date, fiind orientată pe utilizarea de adaptoare (driver) între client și SGBD.

Procesul de conectare la o bază de date implică înregistrarea unui driver corespunzător și realizarea unei conexiuni cu baza de date. O conexiune la o bază de date reprezintă un context prin care sunt trimise secvențe SQL către baza de date și sunt primite rezultate la nivel de aplicație client. API-ul JDBC furnizează acces orientat obiect la bazele de date, prin definirea de clase și interfețe ce modelează concepte abstracte specifice tehnologiei, semnificația celor mai importante concepte JDBC fiind următoarea:

- Driver - clasa Driver și Driver Manager.
- Conexiunea la baza de date – clasa Connection
- Interogări SQL - clasele Statement, PreparedStatement, CallableStatement
- Mulțimi rezultat al execuției - clasa ResultSet

Specificarea accesului la o bază de date creată și găzduită de un SGBD, dintr-o aplicație client Java presupune mai multe etape, ce trebuie parcurse într-o secvențiere strictă și anume: înregistrarea unui driver ce va fi utilizat pentru traducerea către baza de date a frazelor SQL, respectiv stabilirea conexiunii cu baza de date, corect specificată prin intermediul unui string de conexiune specific fiecărei baze de date.



Analiza cerințelor utilizatorilor (Specificații de proiect)

Ipotezele specifice domeniului ales pentru proiect

Se dorește implementarea unui sistem informatic destinat gestiunii unei platforme de studiu. Aplicația va folosi un sistem de gestiune pentru baze de date MySQL, iar interacțiunea cu aceasta va fi realizată doar prin interfața grafică. Funcționalitățile pe care le va oferi programul vizează operații ce țin de gestiunea studenților, profesorilor și administrarea operațiilor curente din cadrul unor programe de studiu.

Aplicația va putea fi accesată, pe baza unui proces de autentificare, de către mai multe tipuri de utilizatori: studenți, profesori, administratori. Pentru fiecare tip de utilizator se vor reține informații precum CNP, nume, prenume, adresa, număr de telefon, email, cont IBAN, numărul de contract. Fiecare utilizator își va putea vizualiza datele personale imediat după ce va accesa sistemul informatic, fără a avea însă posibilitatea de a le modifica. Totodată, programul trebuie să ofere și o funcționalitate pentru deautentificare, prin care se revine la fereastra care solicită datele de acces, astfel încât și un alt utilizator să îl poată folosi ulterior, fără a fi necesară repornirea sa.

Utilizatorul de tip **administrator** poate adăuga, modifica și șterge informații în baza de date, informații legate de utilizatori. De asemenea, va exista și 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. La căutarea unui curs se afișează și numele profesorilor de la acel curs și un buton care permite vizualizarea tuturor studenților înscriși la cursul respectiv.

Pentru un utilizator de tip **profesor** se vor reține și cursurile predate, numărul minim și numărul maxim de ore pe care le poate preda și departamentul din care face parte.

Pentru un utilizator de tip **student** se va reține și anul de studiu și numărul de ore pe care trebuie să le susțină.

Aplicația va permite gestiunea cu ușurință a activităților didactice și astfel a interacțiunilor dintre studenți și profesori. Cursurile sunt predate de mai mulți profesori și au una sau mai multe tipuri de activități (curs, seminar, laborator), o descriere, și un număr maxim de studenți participanți. Studenții se pot înscrie la cursuri și sunt asignați profesorului cu cei mai puțini studenți la data înscrierii. Aceștia sunt evaluați cu note pentru fiecare tip de activitate și primesc o notă finală ca medie ponderată între tipurile de activități. Profesorul stabilește din interfața grafică împărțirea procentuală pe tipurile de activități (ex. 20% seminar, 35% laborator, 45% curs/examenul de la curs).



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Fiecare activitate se desfășoară recursiv între două date, pe o anumită perioadă de timp. La asignarea unui profesor la un curs se vor alege tipurile de activități. De exemplu, profesorul X predă cursul Y cu activitățile: curs –săptămânal, laborator –săptămânal. Ulterior, profesorul poate programa activitățile (curs, seminar, laborator, colocviu, examen) 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 studenții după cursuri și le pot adăuga note. Cataloagele pot fi descărcate sub forma de fișier.

La logare, studenții și profesorii pot să își vadă activitățile din ziua curentă sau pot accesa o pagină cu toate activitățile la care sunt asigurați / înscriși. Aceste liste pot fi descărcate din sistem sub forma unor fișiere.

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 (de exemplu, studentul dorește să participe la laboratorul de BD marți la ora 10. Se înscrie la acea activitate, iar înscrierea este validă doar dacă nu are deja o altă activitate marți la ora 10 sau dacă mai sunt locuri disponibile. În caz contrar, se afișează un mesaj de eroare).

Totodată, studenții se pot înscrie în grupuri de studiu pentru o anumită materie, dacă sunt înscriși la materia respectivă. Aceștia pot să vadă toți membrii grupului și să lase mesaje. Pe grup, studenții pot adăuga activități și să definească un număr minim de participanți și o perioadă în care ceilalți studenți pot să anunțe participarea (de exemplu, un student adăugă o activitate de aprofundare a cursului pentru data de 12.12.2020, ora 16:00, 2 ore, cu număr minim de participanți 5 și timp de expirare 2 ore). Dacă numărul minim nu este atins, activitatea se anulează, iar studenții înscriși la ea primesc un mesaj de informare.

Organizare structurată a cerințelor utilizatorilor

Baza de date trebuie să stocheze informații despre:

- Utilizatori (super-admin, admini, profesori, studenți)
- Cursuri
- Profesorii asigurați la un curs
- Participanții la un curs
- Grupurile de studiu
- Activitățile fiecărui curs și ale grupelor de studiu
- Participanții la o activitate
- Mesajele de pe grupurile de studiu
- Membrii unui grup
- Invitațiile fiecărui utilizator

Mai mult trebuie sa permită si următoarele operații:

- Adăugarea de utilizatori noi daca nu există
- Adăugarea de cursuri daca nu există
- Adăugarea de activități
- Adăugarea de grupuri
- Ștergerea automata a invitațiilor
- Ștergerea automata a activităților de grup
- Calcularea notei finale a unui student la un curs
- Verifica suprapunerea a doua activități in momentul asignării unui student la o materie
- Calcularea numărului total de participanți la o activitate de grup
- Găsirea profesorului cu numărul minim de participant

Aplicația Java trebuie sa permită prelucrarea informațiilor din baza de date, căutarea, ștergerea, modificarea si afișarea informațiilor si conectarea utilizatorilor înregistrați.

Determinarea si caracterizarea de profiluri de utilizatori

Orice utilizator deține următoarele funcționalități:

- Autentificare/Deautentificare
- Vizualizare date personale
- Vizualizare activități într-o zi anume sau in toate zilele
- Vizualizare cursuri
- Vizualizare invitații
- Participare la o activitate didactica sau a unei grupe de studiu

Student	Profesor	Administrator
Înscriere/Părăsire/Căutare curs	Programare activități didactice	Gestionare conturi (Vizualizare, Modificare, Ștergere)
Înscriere activități didactice	Setare ponderi activități	Creare cursuri
Creare/Înscriere/Vizualizare grup de studio	Vizualizare studenți si înscriere note	Asignarea profesorilor la cursuri
Invitare colegi		
Creare/Înscriere activitate de grup		
Chat cu grupul de studio		
Vizualizare note		

Modelul de date si descrierea acestuia

Entități si atributele lor

Users - informații generale despre utilizatori. Atribute: username (PK), password, CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_contract, role.

Studenți - informații suplimentare despre student. Atribute: username (PK, FK), an_studiu, nr_ore.

Profesori - informații suplimentare despre profesor. Atribute: username (PK, FK), departament, min_ore, max_ore.

Cursuri - informații despre cursuri. Atribute: nume_curs (PK), descriere, nr_max_participanti.

Asignare Profesori la Cursuri - informații despre profesorii asignați unor discipline. Atribute: id_asignare_prof_curs (PK), username (FK), nume_curs (FK), pondere_curs, pondere_laborator, pondere_seminar, bool_curs, bool_laborator, bool_seminar.

Asignare Studenți la Cursuri - informații despre studenții înscriși la cursuri. Atribute: id_asignare_student_prof_curs (PK), username (FK), id_asignare_prof_curs (FK), nota_finala, nota_curs, nota_laborator, nota_seminar.

Activitate - informații despre activități. Atribute: id_activ (PK), start_date, final_date, zi, ora, durata, tip_activitate, locuri_max, id_asignare_prof_curs (FK), id_activ_grupa (FK), nr_part_curent.

Participanți activitate - informații despre participanții la o activitate. Atribute: id_part (PK), id_activ (FK), username (FK), nota_activitate.

Grupa Studiu - informații generale despre o grupa de studio. Atribute: id_grupa (PK), nume_curs (FK).

Participanți grupa - informații despre participanții la o activitate de grupa. id_part_grupa(PK), username (FK), id_grupa (FK).

Activitati grupa - informații despre activitățile grupelor de studiu. Atribute: id_activ_grupa (PK), participant, id_grupa (FK), data_expirare.

Chat grupa - stochează mesajele scrise pe chat-urile grupelor. Atribute: id_mesaj (PK), username (FK), textmesaj, id_grupa (FK).

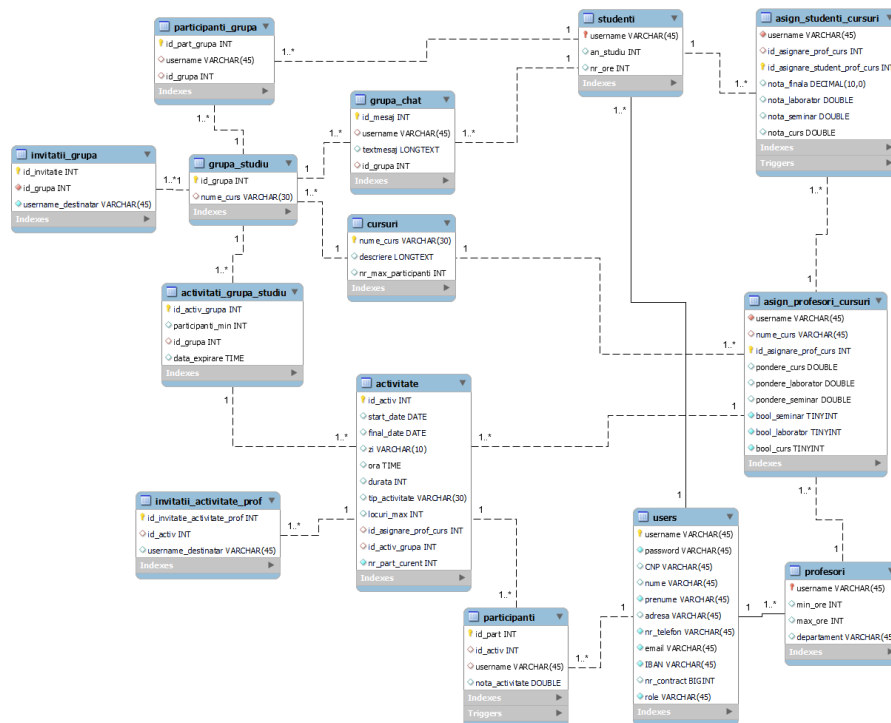
Invitații profesori la activitate de grupa - invitațiile la o activitate către profesori. Atribute: id_invitatie_activitate_prof (PK), id_activ (FK), username_destinatar.

Invitații student in grupa - invitațiile in grupe către studenti. Atribute: id_invitatie (PK), id_grupa (FK), username_destinatar.



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Diagrama EER/UML pentru modelul de date complet



Normalizarea datelor

Normalizare bazelor de date este un proces de optimizare a bazei de date prin care se încearcă minimizarea redundanței datelor, și a anomaliilor de introducere, actualizare și ștergere.

Pentru a identifica corect o pereche formată dintr-un profesor și cursul pe care îl predă, sau între student și cursul la care este înscris, și de asemenea, pentru a rezolva relațiile many-to-many (un student se poate înscrie la mai multe cursuri & un curs poate avea mai mulți studenți înscriși și un profesor poate predă mai multe cursuri & un curs poate fi predat de mai mulți profesori), am introdus tabelele auxiliare **assign_profesori_cursuri** și **assign_profesori_cursuri**. Fiecare pereche profesor-curs, respectiv sintaxa “student X înscris la cursul Y predate de profesorul Z”, are un ID unic prin care este identificată.

Baza noastră de date respectă **BCNF (forma normală Boyce-Codd)**. Atributele fiecărui tabel nu depind de alte atribute, și totodată respecta principiul atomicității. 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.



Detalii de implementare MySQL

DDL (Data Definition Language)

- Crearea bazei de date

```
CREATE SCHEMA IF NOT EXISTS `mydb` ;  
USE `mydb` ;
```

- Crearea unui table

```
CREATE TABLE IF NOT EXISTS `mydb`.`users` (  
  `username` VARCHAR(45) NOT NULL,  
  `password` VARCHAR(45) NOT NULL,  
  `CNP` VARCHAR(45) NULL DEFAULT NULL,  
  `nume` VARCHAR(45) NULL DEFAULT NULL,  
  `prenume` VARCHAR(45) NOT NULL,  
  `adresa` VARCHAR(45) NULL DEFAULT NULL,  
  `nr_telefon` VARCHAR(45) NOT NULL,  
  `email` VARCHAR(45) NOT NULL,  
  `IBAN` VARCHAR(45) NOT NULL,  
  `nr_contract` BIGINT NULL DEFAULT NULL,  
  `role` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`username`))
```

- Adăugarea cheilor străine si a constrângerilor

```
CREATE TABLE IF NOT EXISTS `mydb`.`profesori` (  
  `username` VARCHAR(45) NOT NULL,  
  `min_ore` INT NULL DEFAULT NULL,  
  `max_ore` INT NULL DEFAULT NULL,  
  `departament` VARCHAR(45) NULL DEFAULT NULL,  
  PRIMARY KEY (`username`),  
  UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE,  
  CONSTRAINT `username`  
    FOREIGN KEY (`username`)  
    REFERENCES `mydb`.`users` (`username`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb3;
```



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

DML (Data Manipulation Language)

- Inserarea datelor in tabele (Popularea)

```
insert into users (username, password, CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_contract, role) values
('daritamas', '1234', '123456789', 'Tamas', 'Dari', 'Sarmasag, Salajului 72', '0746965428', 'dari.tamas@yahoo.com', 'RO01BCR12345', 1234, 'student'),
('silviugrumazescu', '1234', '123456788', 'Silviu', 'Grumazescu', 'Cluj, Prezan 73', '0746965430', 'silviugrumazescu@yahoo.com', 'RO01BCR12346', 1235, 'student'),
('denisalexandru', '1234', '123456777', 'Denis', 'Alexandru', 'Carbunesti, AndreiD 69', '0746965440', 'alexandrudenisi@yahoo.com', 'RO01BCR12347', 1236, 'student'),
('vasilepop', '1234', '123456456', 'Vasile', 'Pop', 'Cluj, Gradinarilor 72', '0746967428', 'vasilepop@yahoo.com', 'RO01BCR12885', 9871, 'profesor'),
('robertvarga', '1234', '123452289', 'Robert', 'Varga', 'Cluj-Napoca, Eminescu 12', '0745555428', 'robertvarga@yahoo.com', 'RO01BCR18895', 9872, 'profesor');
```

- Modificarea datelor din tabele

```
UPDATE `mydb`.`assign_profesori_cursuri` SET `pondere_curs` = '0.20',
| `pondere_laborator` = '0.40' WHERE (`id_asignare_prof_curs` = '2');
```

SQL Programatic

- Proceduri
 - Înregistrarea unui utilizator

```
DELIMITER $$
USE `mydb`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `RegisterUser`(IN username varchar(45), IN password varchar(45), IN CNP varchar(45),
IN nume varchar(45), IN prenume varchar(45), IN adresa varchar(45),
IN nr_telefon varchar(45), IN email varchar(45),
IN IBAN varchar(45), IN nr_contract varchar(45), IN role varchar(45))
begin
insert into users(username, password, CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_contract, role)
| values (username, password, CNP, nume, prenume, adresa, nr_telefon, email, IBAN, nr_contract, role);
end$$
```

DELIMITER ;

- Obținerea datelor unui utilizator

```
DELIMITER $$
USE `mydb`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetUserInfo`(IN username VARCHAR(45), OUT CNP varchar(45), OUT nume varchar(45),
OUT prenume varchar(45), OUT adresa varchar(45), out nr_telefon varchar(45),
OUT email varchar(45), OUT IBAN varchar(45), OUT nr_contract varchar(45))
begin
select u.CNP, u.nume, u.prenume, u.adresa, u.nr_telefon, u.email, u.IBAN, u.nr_contract
| into CNP,nume, prenume, adresa, nr_telefon, email, IBAN, nr_contract from users u where u.username = username;
end$$
```

DELIMITER ;



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Repartizarea unui student la profesorul cu cei mai putini studenti atribuiti

```
DELIMITER $$
USE `mydb`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `InsertStudentLaProfesorMin`(IN username_student varchar(45), IN nume_curs_ales varchar(45))
begin
    select tbl.valu into @id_prof_min_ales from (select apc.id_asignare_prof_curs as valu, count(asccl.id_asignare_student_prof_curs) cnt
    from asign_profesori_cursuri apc left join asign_studenti_cursuri asccl on apc.id_asignare_prof_curs = asccl.id_asignare_prof_curs
    where apc.nume_curs = nume_curs_ales group by apc.id_asignare_prof_curs) tbl order by tbl.cnt asc limit 1;

    insert into asign_studenti_cursuri(username, id_asignare_prof_curs) values (username_student, @id_prof_min_ales);
end$$

DELIMITER ;
```

- Verificarea existentei unei activitati intr-un interval de timp

```
DELIMITER $$
USE `mydb`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `checkIfActivityInInterval`(IN username varchar(45), IN startTime TIME, IN endTime TIME,
IN zi varchar(10), OUT isActivityInInterval int)
begin
    if exists
        (select activ.id_activ from activitate activ, asign_profesori_cursuri apc, asign_studenti_cursuri asccl, participanti part
        where asccl.username = username and asccl.id_asignare_prof_curs = apc.id_asignare_prof_curs
        and apc.id_asignare_prof_curs = activ.id_asignare_prof_curs and part.id_activ = activ.id_activ
        and part.username = username and ((ADDTIME(activ.ora, CONCAT(activ.durata, ":00:00")) > startTime
        and ADDTIME(activ.ora, CONCAT(activ.durata, ":00:00")) < endTime)
        or (activ.ora > startTime and activ.ora < endTime) or (activ.ora <= startTime
        and ADDTIME(activ.ora, CONCAT(activ.durata, ":00:00")) >= endTime )) and activ.zi = zi )
    then
        set isActivityInInterval = 1;
    else
        set isActivityInInterval = 0;
    end if;
end$$

DELIMITER ;
```

- Obținerea activitatilor la care poate participa un student

```
DELIMITER $$
USE `mydb`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `showActStudPosibile`(in username varchar(45))
begin
    select distinct activitate.id_activ from activitate, asign_studenti_cursuri, participanti
    where asign_studenti_cursuri.username=username and
    asign_studenti_cursuri.id_asignare_prof_curs = activitate.id_asignare_prof_curs
    and activitate.locuri_max > activitate.nr_part_curent
    and not exists (select id_part from participanti p where p.id_activ = activitate.id_activ and p.username = username);
end$$

DELIMITER ;
```



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Triggere
 - Incrementarea numărului participanților la o activitate

```
DELIMITER $$
USE `mydb`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `mydb`.`incrementNumberOfParticipantiActivitate`
AFTER INSERT ON `mydb`.`participanti`
FOR EACH ROW
begin
    update activitate set nr_part_curent = nr_part_curent + 1 where activitate.id_activ = new.id_activ;
end$$
```

- Calcularea notei finale pe baza notelor obtinute si ponderilor stabilite

```
delimiter //
Create Trigger nota_finala before update on asign_studenti_cursuri for each row
SET
    new.nota_finala = ( SELECT pondere_curs
                        from asign_profesori_cursuri where id_asignare_prof_curs = new.id_asignare_prof_curs ) * new.nota_curs
    + ( SELECT pondere_seminar
        from asign_profesori_cursuri where id_asignare_prof_curs = new.id_asignare_prof_curs ) * new.nota_seminar
    + ( SELECT pondere_laborator
        from asign_profesori_cursuri where id_asignare_prof_curs = new.id_asignare_prof_curs ) * new.nota_laborator;
//
DELIMITER ;
```

- Event

```
SET GLOBAL event_scheduler="ON";
DELIMITER ..
CREATE EVENT stergere_activitati_expire_event
ON SCHEDULE EVERY 1 MINUTE STARTS NOW()
ON COMPLETION PRESERVE ENABLE
DO
BEGIN
    UPDATE activitati_grupa_studiu SET data_expirare=timediff(data_expirare, '00:01:00') where data_expirare > TIME('00:00:00');

    delete ags from activitati_grupa_studiu ags join activitate activ on activ.id_activ_grupa = ags.id_activ_grupa where
    ags.data_expirare <= '00:00:00' and activ.nr_part_curent < ags.participanti_min;

END ..
delimiter ;
```



Detalii de implementare Java

Interacțiunea dintre utilizator și baza de date se realizează cu ajutorul interfeței grafice, creată special pentru a satisface nevoile fiecărui utilizator. Ideile centrale în jurul cărora s-a dezvoltat interfața au fost separarea utilizatorilor în roluri și respectarea arhitecturii MVC. Fiecărui obiect de tip **User** îi corespunde un parametru **role**, care oferă utilizatorului acces la **view**-ul corespunzător prin intermediul unui **controller**.

```
public class User {  
    protected String username, password, CNP, nume, prenume,  
        adresa, nr_telefon, email, IBAN, nr_contract, role;  
    public User(String username, String password){  
        this.username = username;  
        this.password = password;  
    }  
    public User(String username, String password, String role){  
        this.username = username;  
        this.password = password;  
        this.role = role;  
    }  
}
```

```
public void switchToUserScreen(String username, String password, String role){  
    loginController.setFrameVisible(false);  
    switch(role){  
        case "student":  
            bdconn.setNewCurrentUser(username, password, role);  
            studentViewController.setCurrentUser(bdconn.getUserInformation());  
            studentViewController.resetView();  
            break;  
  
        case "profesor":  
            bdconn.setNewCurrentUser(username, password, role);  
            profesorViewController.setCurrentUser(bdconn.getUserInformation());  
            profesorViewController.resetView();  
            break;  
  
        case "admin":  
            bdconn.setNewCurrentUser(username, password, role);  
            adminViewController.resetView();  
            adminViewController.fillCursesList();  
            adminViewController.fillGrupetList();  
            break;  
  
        case "superAdmin":  
            bdconn.setNewCurrentUser(username, password, role);  
            superAdminController.resetView();  
            superAdminController.fillCursesList();  
            superAdminController.fillGrupetList();  
            break;  
    }  
}
```




UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Panoul principal al userului de tip student:

```
public StudentHomePanel(ActionListener logoutButtonListener, ActionListener vizDatePersListener,
    ActionListener inscriereCursuriListener, ActionListener inscriereActivitatiListener,
    ActionListener VizualizareNoteListener, ActionListener VizActivitatiListener,
    ActionListener vizGrupaStudiuListener) {
    initComponents();
    buttonLogOut.addActionListener(logoutButtonListener);
    buttonVizDatePers.addActionListener(vizDatePersListener);
    buttonInscriereCurs.addActionListener(inscriereCursuriListener);
    buttonInscriereActivitati.addActionListener(inscriereActivitatiListener);
    buttonVizNote.addActionListener(VizualizareNoteListener);
    buttonVizActivitati.addActionListener(VizActivitatiListener);
    buttonVizGrupStudiu.addActionListener(vizGrupaStudiuListener);
}
```

Fragmente de cod din interiorul controller-ului studentului:

```
public void changePanel(String stringPanel){
    cardLayout.show(userFrame.mainPanel, stringPanel);
    userFrame.pack();

    switch(stringPanel){
        case "studenthomepanel":
            userFrame.setSize(studentHomePanel.getPreferredSize());
            break;
        case "vizdatestudentpanel":
            userFrame.setSize(vizDateStudentPanel.getPreferredSize());
            break;
        case "inscrierecurspanel":
            userFrame.setSize(inscriereCursPanel.getPreferredSize());
            break;
        case "inscriereactivitatipanel":
            userFrame.setSize(inscriereActivitatePanel.getPreferredSize());
            break;
        case "viznotestudentpanel":
            userFrame.setSize(vizNoteStudentPanel.getPreferredSize());
            break;
        case "vizactivitatistudent":
            userFrame.setSize(vizActivitatiStudent.getPreferredSize());
            break;
        case "grupstudiupanel":
            userFrame.setSize(grupStudiuPanel.getPreferredSize());
            break;
        case "selectareCursGrupaPanel":
            userFrame.setSize(selectareCursGrupaPanel.getPreferredSize());
            break;
        case "vizualizareGrupaPanel":
            userFrame.setSize(vizualizareGrupaPanel.getPreferredSize());
            break;
    }
}
```



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

```
class VizualizareGrupaListener implements ActionListener{
    public void actionPerformed(ActionEvent e){
        int index = grupStudiuPanel.listaGrupe.getSelectedIndex();
        int idGr = grupStudiuPanel.grupe.get(index);
        initializeParticipantiGrupa(idGr);
        currentUser = bdconn.getUserInformation();
        if(bdconn.VerificareInscriereStudentGrupa(currentUser.username, idGr)){
            vizualizareGrupaPanel.buttonProgramareActivitate.setEnabled(true);
            vizualizareGrupaPanel.inviteButton.setEnabled(true);
            vizualizareGrupaPanel.buttonSendMSG.setEnabled(true);
            initializeGrupaChat(idGr);
        }
        else{
            vizualizareGrupaPanel.buttonProgramareActivitate.setEnabled(false);
            vizualizareGrupaPanel.inviteButton.setEnabled(false);
            vizualizareGrupaPanel.buttonSendMSG.setEnabled(false);
            initializeGrupaChat(-1);
        }

        // set-up sugestii participanti
        initializeSugestiiGrupa(idGr);

        changePanel("vizualizareGrupaPanel");
    }
}
```

În urma folosirii elementelor interfeței (butoane, liste, tabele, panouri), pentru ca operația cerută de către utilizator să poată fi realizată, este nevoie de a interacționa cu baza de date prin intermediul interogărilor sau modificărilor de tip ALTER, INSERT, DELETE. Aceste operații sunt posibile cu ajutorul unor metode definite în clasa **BDConnection**.

Clasa **BDConnection** este responsabilă de a stabili conexiunea dintre baza de date și aplicația Java prin intermediul **JDBC**-ului.

```
public BDConnection(){
    try{
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb?allowPublicKeyRetrieval=
    )
    catch(SQLException ex){
        ex.printStackTrace();
    }
}
```




UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Exemple de metode din cadrul clasei `BDConnection` (metode apelate din cadrul tuturor controllerelor):

```
String isUserRegistered(String username, String password){
    try{
        Statement stmt = conn.createStatement();
        String command = "select role from users where username =\"" + username
            + "\" and password=\"" + password + "\"";

        ResultSet rset = stmt.executeQuery(command);
        if(!rset.next()){
            System.out.println("Logarea nu a avut succes");
            return null;
        }
        else{
            System.out.println("Userul este inregistrat");
            return rset.getString("role");
        }
    }
    catch(SQLException ex){
        ex.printStackTrace();
    }
    return null;
}
```

```
Integer getIDAssignProfCurs(String numecurs){
    Integer queryData = new Integer(0);
    try{
        Statement stmt = conn.createStatement();
        String command = "select id_asignare_prof_curs from asign_profesori_cursuri where username =
            + currentUser.username + \"\" and nume_curs = \"\" + numecurs + \"\"";

        ResultSet rset = stmt.executeQuery(command);
        //System.out.println(rset.getInt("id_asignare_prof_curs"));
        rset.next();
        System.out.println(rset.getInt("id_asignare_prof_curs"));
        queryData=rset.getInt("id_asignare_prof_curs");
        System.out.println(queryData);
        return queryData;
    }
    catch(SQLException ex){
        ex.printStackTrace();
        return null;
    }
}
```



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

```
public void deleteInviteProfesorGrupa(String username, int idActiv){
    try{
        String command = "delete from invitatii_activitate_prof where id_activ = "
            + idActiv+ " and username_destinator = \"" + username + "\";";
        PreparedStatement stmt = conn.prepareStatement(command);
        stmt.execute();

    }
    catch(SQLException ex){
        ex.printStackTrace();
    }
}

public void InsertGrupaStudiu(String curs){
    try{
        String command = "insert into grupa_studiu(nume_curs) values(?);";
        PreparedStatement stmt = conn.prepareStatement(command);
        stmt.setString(1, curs);
        stmt.execute();
    }
    catch(SQLException ex){
        ex.printStackTrace();
    }
}

public void InsertActivitateProf(JTextField[] fields, String tipActivitate, int idAPC){
    try{
        String command = "insert into activitate(start_date, final_date, zi, ora, durata,"
            + " tip_activitate, locuri_max, id_asignare_prof_curs, nr_part_curent) values(\""
            + fields[0].getText() + "\", \"" + fields[1].getText() + "\", \""
            + fields[2].getText() + "\", \"" + fields[3].getText() + "\", "
            + Integer.parseInt(fields[4].getText()) + ", \"" + tipActivitate + "\", "
            + Integer.parseInt(fields[5].getText()) + ", " + idAPC + ", 0);";

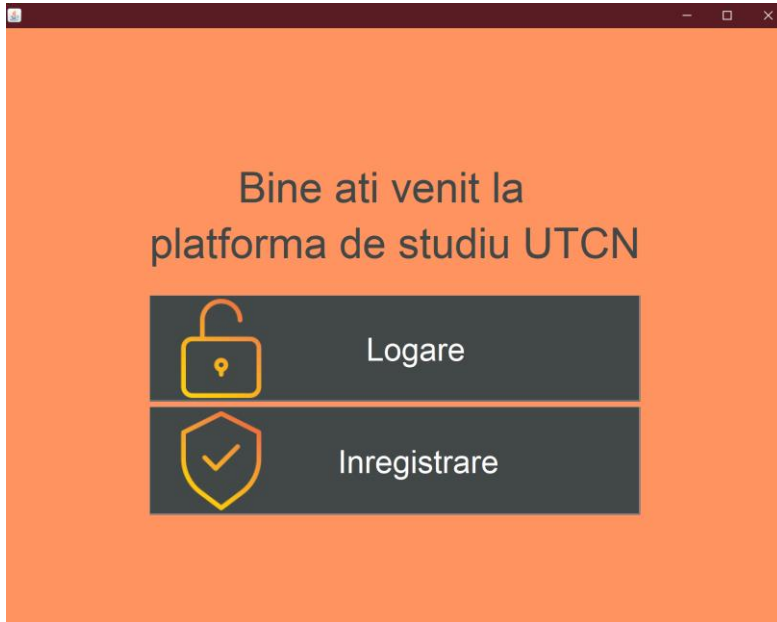
        Statement stmt = conn.createStatement();
        stmt.execute(command);

    }
    catch(SQLException ex){
        ex.printStackTrace();
    }
}

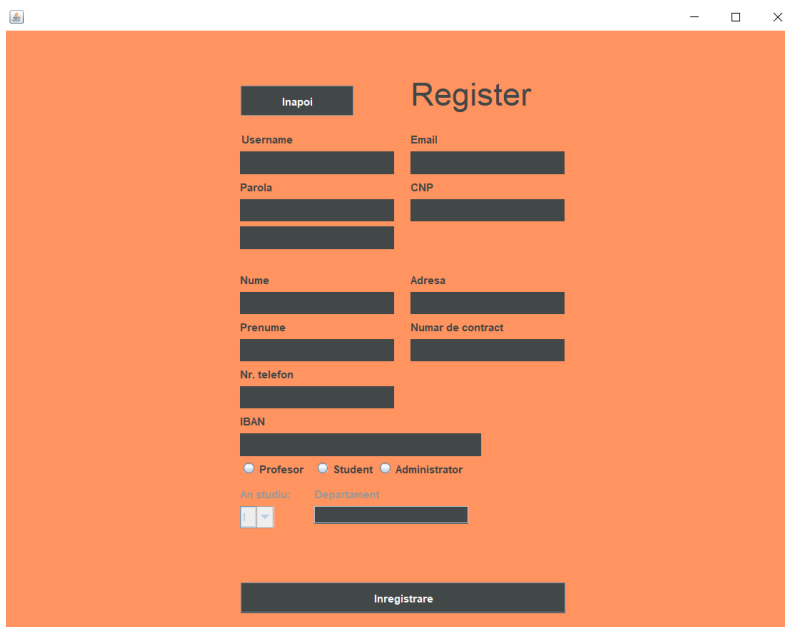
public void SendMSG(String username, String msg, int idGrupa){
    try{
        String command = "insert into grupa_chat(username, textmesaj, id_grupa) values(\""
            + username + "\", \"" + msg + "\", " + idGrupa + ");";
        PreparedStatement stmt = conn.prepareStatement(command);
        stmt.executeUpdate();
    }
    catch(SQLException ex){
        ex.printStackTrace();
    }
}
```

Utilizarea aplicației

➤ Pagina de întâmpinare



➤ Panoul de înregistrare





UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

➤ Panoul de autentificare

Inapoi

Login

Username:

Password:

Login

➤ Interfața studentului

Vizualizare date personale

Vizualizare note

Vizualizare activitati

Grup de studiu

Inscriere activitati

Inscriere cursuri

Log out



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Elemente ale interfeței studentului
 - Panoul grupei de studiu

Grupele mele

Grupa 13 - CAN
Grupa 14 - Engleza

Vizualizare grupa

Inscriere grupa

Creare grupa

Invitatii grupe

Accepta

Inapoi

- Panoul pentru vizualizarea notelor

Disciplina	Nota curs	Nota seminar	Nota laborator	Nota finala
CAN	5.0	8.5	10.0	6
Engleza	0.0	0.0	0.0	0

Intoarcere



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Panoul pentru înscrierea la cursuri

Cursuri la care esti inscris:

CAN
Engleza

Descriere:

Circuite analogice si numerice

Profesor:

Peculea Adrian

Paraseste curs

Cursuri la care te poti inscrie:

MSI
PC
PLA
PSN
SDA

Inscrie-te

Back

- Panoul pentru vizualizarea activităților

Activitatile in ziua curenta

Disciplina	Tip activitate	Ora	Durata
CAN	examen final	10:00:00	2

Toate Activitatile

Parasire Activitate

Disciplina	Tip activitate	Ora	Durata	Ziua
CAN	examen final	10:00:00	2	Joi
CAN	curs	18:00:00	2	Luni

Descarcare activitati din ziua curenta

Descarcare toate activitatile

Inapoi



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Panoul pentru înscrierea la activități

Curs	Tip Activitate	Ziua	Ora	Durata
CAN	examen final	Joi	10:00:00	2

Intoarcere Inscriere acti...

- Interfața profesorului

Vizualizare date personale Cursuri Catalog


Activitatile mele Programare activitati Vizualizare Invitatii

Log out



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Panoul pentru administrarea cursurilor



—

□

×

Cursurile mele

CAN

PLA

Pondere Curs

0.7

Pondere Laborator

0.0


Pondere Seminar

0.3

Inapoi

Salveaza modificari

- Panoul pentru vizualizarea activităților



—

□

×

Activitățile în ziua curentă

Disciplina	Tip activitate	Ora	Durata
CAN	examen final	10:00:00	2

Toate activitățile

Disciplina	Tip activitate	Ora	Durata	Ziua
CAN	examen final	10:00:00	2	Joi
CAN	curs	18:00:00	2	Luni

Descarcare activitati din ziua curenta

Descarcare toate activitățile

Salveaza modificarile

Inapoi



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Panoul pentru vizualizarea si administrarea catalogului

Alegeti un curs:

CAN

PLA

Studenti:

Nume	Prenume	Username	Nota Curs	Nota Seminar	Nota Laborator	Nota Finala
Tamas	Dari	daritamas	5.0	0.0	10.0	4

Descarcare Catalog

Salveaza Modificarile

Inapoi

- Panoul pentru programarea activităților

Cursurile mele

CAN

PLA

Programare Curs

Programare Laborator

Programare Seminar

Programare Examen

Data incepere

YYYY-MM-DD

Data finalizare

YYYY-MM-DD

Ziua

Ora

HH:MM:SS

Durata

in ore

Locuri maxime

Programeaza activitatea

Data incepere

YYYY-MM-DD

Data finalizare

YYYY-MM-DD

Ziua

Ora

HH:MM:SS

Durata

in ore

Locuri maxime

Programeaza activitatea

Data incepere

YYYY-MM-DD

Data finalizare

YYYY-MM-DD

Ziua

Ora

HH:MM:SS

Durata

in ore

Locuri maxime

Programeaza activitatea

Colocvlu

Examen final

Data

YYYY-MM-DD

Ziua

Ora

HH:MM:SS

Durata

in ore

Locuri maxime

Programeaza activitatea

Pentru a programa o activitate, nu uitati sa selectati cursul din lista afisata in stanga!

Intoarcere



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Panoul pentru vizualizarea invitațiilor

A screenshot of a web application window with a dark red header and an orange background. The window contains a table with the following data:

Activitate	Data	Ora	Durata
Consultatie	2022-01-22	18:00:00	2

Below the table is a large empty rectangular area. At the bottom of the window are two buttons: "Intoarcere" (left) and "Accepta" (right).

- Interfața administratorului


A screenshot of the "Interfața Super-Administratorului" (Super-Administrator Interface). The interface is divided into several sections:

- Log Out**: A button to log out.
- Selectează un utilizator după nume**: A search bar with "Nume" and "Prenume" tabs.
- Selectează un utilizator din listă**: A dropdown menu with "Student" selected, showing a list of students including Ada Vadean, Ana Mudura, Andrei Mota, Robert Branzoi, Tudor Buz, Cristian Bales, Daria Franciolu, Dori Tamas, Alexandru Denis, Florin Itana, Lorena Maties, Daniel Lungu, Petra Linsaru, Rares Motupan, Robert Bob, and Grumazescu Silviu.
- Date personale**: Fields for Username (branzorobert), Password (1234), Name (Branzoi), Prenume (Robert), Contract (154), CNP (123454545), Adresa (Sighet, str. Postasului nr. 10), Nr. Telefon (0740574836), Email (branzorobert@yahoo.com), and IBAN (RO81140534054663). Buttons for "Salveaza modificarile" and "Sterge utilizatorul" are present.
- Selectează un curs**: A list of courses: CAN, Engleza, MSI, PC, PLA, PSN, SDA.
- Profesorii la cursul selectat**: A list of professors: Peculea Adrian, Buzura Sorin.
- Studenti**: A list of students: Tamas Dori.
- Grupele de studiu**: A list of study groups: Grupa 13, Grupa 14.
- Asigneaza un profesor**: Radio buttons for Curs, Seminar, and Laborator.
- Asigneaza un student**: A section for assigning students.
- Adauga un nou curs**: Fields for Name Curs, Nr. maxim de studenti, and buttons for Adauga curs, Sterge cursul, Asigneaza profesor, Sterge profesorul din curs, Asigneaza student, and Sterge studentul din curs.



UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

➤ Panoul de date personale

— □ ×

Date personale

Nume	Peculea
Prenume	Adrian
Username	adrianpeculea
CNP	123459989
Adresa	Cluj-Napoca, Eminescu 99
Nr. Telefon	0745595428
Email	adrianpeculea1@yahoo.com
Nr. Contract	9111
Departament	matematica
Nr. ore minim	10
Nr. ore maxim	30

Intoarcere

Concluzii. Limitări si dezvoltări ulterioare

Proiectul prezentat oferă o soluție pentru digitalizarea și gestionarea eficientă a sistemului informațional al unei platforme de studiu al unei universități. Utilizatorii către care este destinată platforma sunt studenții, profesorii și administratorii.

Aplicația oferă diferite servicii în funcție de tipul utilizatorului autentificat, accesarea și corectitudinea acestor servicii fiind facilitată de interfața grafică configurată special în acest scop. Vizualizarea propriilor date este o funcționalitate comună pentru toate tipurile de utilizator, la fel ca vizualizarea programului de activități.

În timp ce conturile de student, profesor sau administrator sunt generate în urma unei înregistrări, rolul de super-administrator este setat prin popularea implicită a bazei de date. Există funcționalități particulare determinate de rolul ales la înregistrare (Ex: Un student nu poate modifica ponderile sau notele de la un curs, un profesor nu poate vizualiza chatul unei grupe de studiu). Super-administratorul are, în plus față de un administrator normal, dreptul de a vizualiza și manipula datele personale ale celorlalți administratori.

În cazul unor dezvoltări ulterioare, platforma poate beneficia de un design atractiv al interfeței grafice prin adăugarea unor pictograme sugestive sau a unei secțiuni de Ajutor. Platforma are deja caracteristica de a programa activități de tip examen sau colocviu, așa ca ar fi destul de intuitiv ca în viitor să se implementeze și capacitatea susținerii acestor examene prin intermediul platformei.

Pentru a i se asigura practicitatea, platforma ar putea beneficia în viitor și de o bază de date stocată în mediul CLOUD, lucru ce ar putea facilita accesarea platformei pe dispozitive care nu au bază de date creată în prealabil, deci implicit, de pe telefoane mobile.

Bibliografie

1. MVC Architecture in Java - <https://www.javatpoint.com/mvc-architecture-in-java>
2. Package javax.swing - <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>
3. Event Scheduler Overview - <https://dev.mysql.com/doc/refman/5.7/en/events-overview.html>
4. Apache POI - the Java API for Microsoft Documents - <https://poi.apache.org/>
5. JDBC Basics - <https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>
6. How To Use MySQL Triggers - <https://phoenixnap.com/kb/mysql-trigger>
7. MySQL ON DELETE CASCADE - <https://www.educba.com/mysql-on-delete-cascade/>
8. What is Normalization in SQL and what are its types?
<https://www.edureka.co/blog/normalization-in-sql/>
9. Class Collections - <https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html>
10. How to save edited JTable data to database? -
<https://stackoverflow.com/questions/18151123/how-to-save-edited-jtable-data-to-database>
11. Introducerea in Baze de Date, Cosmina Ivan -
https://ftp.utcluj.ro/pub/users/civan/IBD/1_CURS_Resurse/