Universitatea Dunărea de Jos

Facultatea de Automatică și Calculatoare

Proiect

Baze de Date II

Bogos Tudor

Grupa 22c32A

Profesor de curs: Conf. dr. ing. Tudorie Cornelia

Îndrumător de proiect: Conf. dr. ing. Munteanu Dan

Table of Contents

[Tema (Examen de admitere la facultate) 2](#_Toc188004381)

[Analiza problemei 2](#_Toc188004382)

[Interacțiunile dintre entități 4](#_Toc188004383)

[Specificarea cerințelor 5](#_Toc188004384)

[Proiectarea la nivel conceptual (Diagrama Entitate-Relatie) 6](#_Toc188004385)

[Proiectarea la nivel logic (Schema bazei de date relaționale) 7](#_Toc188004386)

[Proiectarea la nivel fizic (Tabele, câmpuri, tipul fiecărui câmp, dimensiune, restricții de integritate) 8](#_Toc188004387)

[Descrierea aplicației (manualul de utilizare) 12](#_Toc188004388)

[Bibliografie 15](#_Toc188004389)

## Tema (Examen de admitere la facultate)

Lucrarea de față își propune să prezinte o aplicație software care gestionează procesul de admitere la facultate. În principal, aplicația oferă două tipuri de roluri pentru utilizatori: student (candidați la admitere) și admin (administrator – de obicei comisia de admitere). Studenții pot verifica dacă au fost admiși sau respinși, iar administratorii pot realiza întreținerea bazei de date (inserare, actualizare, ștergere), pot realiza import și export de date, precum și statistici (grafice).

## Analiza problemei

În viața reală, procesul de admitere la facultate presupune interacțiunea mai multor entități și actori, fiecare cu roluri și scopuri specifice. Vom porni de la descrierea acestui proces înainte să introducem soluția informatică, pentru a surprinde cadrul conceptual.

Context general al examenului de admitere\textbf{Context general al examenului de admitere}Context general al examenului de admitere

În momentul organizării unui examen de admitere la o facultate, se pleacă de la următoarele realități:

1. Facultatea – este instituția care organizează concursul de admitere, pune la dispoziție un număr de locuri disponibile și stabilește condițiile de promovare și criteriile de selecție.
   * Atribute relevante în realitate: denumire, adresa, număr de locuri.
   * Acțiuni tipice:
     + primește înscrieri de la candidați;
     + organizează examenul sau ia în considerare anumite criterii de selecție (media bacalaureat, note la anumite discipline, etc.);
     + afișează rezultatele de admitere sub formă de liste cu admiși și respinși;
     + poate colabora cu alte entități, cum ar fi Ministerul Educației, pentru validarea numărului de locuri sau a metodologiei de concurs.
2. Studentul (Candidatul) – este persoana care își depune dosarul pentru a participa la examenul de admitere, susține eventualele probe necesare și ulterior așteaptă rezultatele.
   * Atribute relevante în realitate: nume, prenume, CNP (cod numeric personal unic), nota obținută la examen (sau mediile calculabile), opțiunea aleasă (facultatea pentru care candidează).
   * Acțiuni tipice:
     + susține examenul de admitere (dacă există probe scrise/orale);
     + așteaptă rezultatele;
     + primește decizia de admitere sau respingere;
     + în cazul admiterii, poate confirma locul.
3. Admin (Comisia de admitere sau personalul administrativ al facultății) – este cel care gestionează înscrierile, introduce informații despre candidați, gestionează listele de candidați admiși/respinși, realizează importul sau exportul de date (spre exemplu, import din fișiere CSV cu lista de candidați înscriși).
   * Atribute relevante în realitate: username, parolă (pentru acces la date confidențiale).
   * Acțiuni tipice:
     + verifică, introduce și validează datele despre studenți;
     + actualizează situația candidaților (e.g., cine a promovat examenul, cine a fost respins, dacă mai există contestații etc.);
     + exportă datele pentru a fi analizate în Excel sau publicate sub formă de PDF;
     + se asigură că datele sunt corecte și actualizate permanent;
     + importă datele din fișiere CSV, ceea ce asigură un mod rapid de încărcare a informațiilor de la un volum mare de candidați.
4. Procesul de admitere – se bazează pe criterii clare de selecție. În cazul de față, am presupus că se face un clasament în funcție de notă, iar un student este declarat „admis” dacă se află în primele X poziții (unde X este numărul de locuri pentru respectiva facultate).
   * Atribute relevante: notele candidaților, ordonarea descrescătoare a acestora, numărul total de locuri.
   * Acțiuni tipice:
     + sortare candidați în funcție de notă;
     + stabilirea pragului de admitere în funcție de capacitatea facultății;
     + afișarea listei de candidați admiși vs. respinși.

## Interacțiunile dintre entități

* Studentul interacționează cu Facultatea prin depunerea dosarului de admitere, susținerea examenelor și așteptarea rezultatului.
* Admin (comisia) mediază între Facultate și Student, ocupându-se de încărcarea datelor în aplicația centrală, de validări, importuri, exporturi, actualizări etc.
* Facultatea stabilește câte locuri oferă, iar admin introduce aceste date în sistem.
* Pe baza notei, se actualizează statusul fiecărui student (admis sau respins).
* Studenții pot, la rândul lor, consulta aplicația pentru a vedea la ce facultate s-au înscris, nota și dacă au fost admiși.

Astfel, din punct de vedere non-informatic, avem trei mari entități cu roluri clar definite: Student, Facultate și Admin, iar atributele lor principale se regăsesc în realitatea de zi cu zi. Aplicația va prelua și modela aceste entități pentru a putea realiza în mod automat gestionarea admiterii.

## Specificarea cerințelor

Aplicația propusă își propune următoarele funcționalități, organizate pe roluri:

1. Rolul Student:
   * Să se poată autentifica în aplicație pe baza unor date (nume, prenume, CNP).
   * Să poată verifica dacă a fost admis la facultate sau nu, pe baza statusului curent din baza de date.
   * Să vadă date de informare: ID-ul său de student, la ce facultate a aplicat și care este statusul (admis/respins).
2. Rolul Admin (administrator, comisia de admitere):
   * Să se poată autentifica pe baza unui username și a unei parole.
   * Să poată gestiona întreaga bază de date:
     + Inserare de studenți noi (introducerea datelor unui candidat);
     + Modificarea datelor unui student deja existent (în cazul în care se constată erori sau se schimbă informații);
     + Ștergerea unui student (ex. retragerea dosarului);
     + Importul de date dintr-un fișier CSV (pentru a adăuga un volum mare de candidați din surse externe);
     + Exportul datelor în format CSV și PDF (pentru analiză ori pentru publicare).
   * Să poată lansa o funcție de actualizare a statusului (admis/respins) pentru toți studenții în funcție de notele lor și de capacitatea (numărul de locuri) la fiecare facultate.
   * Să poată genera grafice sau diagrame care să arate situația admiterii (ex.: un grafic tip „felii de plăcintă” – pie chart, cu numărul de locuri ocupate vs. locurile totale).

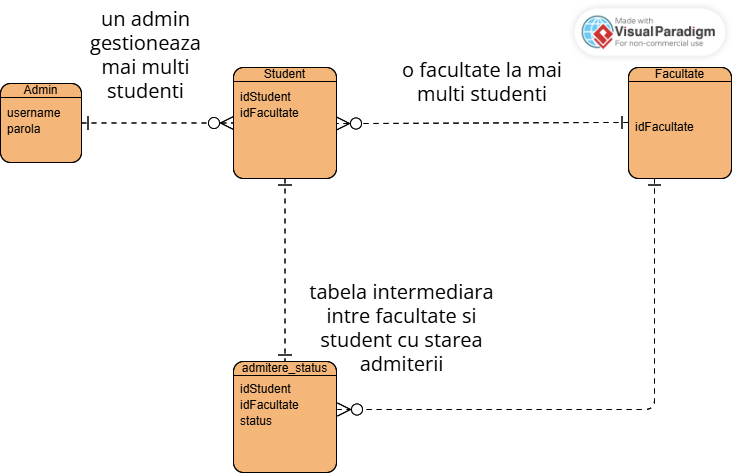
Prin urmare, ce am făcut în aplicație:

* Am implementat aceste două roluri (student și admin).
* Studentul se loghează, își introduce datele personale (nume, prenume, CNP), iar aplicația îl recunoaște și îi afișează statusul și facultatea pentru care a optat.
* Admin-ul poate face tot ceea ce ține de gestionarea bazei de date, inclusiv importul și exportul de fișiere.

## Proiectarea la nivel conceptual (Diagrama Entitate-Relatie)

În diagrama E-R (Entity-Relationship) avem următoarele entități și relații principale:

* Student – conține ID-ul studentului, Nume, Prenume, CNP, Nota, Opțiunea (facultate).
* Facultate – conține ID-ul facultății, Numele facultății, Adresa, numărul de locuri disponibile.
* Admitere\_Status – entitate (sau relație) ce marchează faptul că un anumit student are un anumit status (admis/respins) la o anumită facultate. Aceasta relaționează Student și Facultate.
* Admin – entitate care conține username, parolă, ultimul login (pentru monitorizarea accesului).

Se poate observa o relație 1-to-M între Facultate și Studenți (un student are o singură opțiune principală, dar la nivel logic se pot gestiona și mai multe opțiuni), iar statusul concret (admis/respins) este stocat în entitatea intermediară.

## Proiectarea la nivel logic (Schema bazei de date relaționale)

Fiecare entitate devine un tabel în baza de date:

* Student – (idStudent, Nume, Prenume, CNP, Nota, idFacultateOptiune, Optiune)
* Facultate – (idFacultate, Nume\_Facultate, Adresa, numar\_locuri)
* admitere\_status – (idStudent, idFacultate, status)
* admin – (username, password, last\_login)

Cheile primare, cheile străine, tipurile de date, precum și relațiile dintre tabele sunt clar indicate în schema logică.

A screenshot of a computer screen

Description automatically generated

## Proiectarea la nivel fizic (Tabele, câmpuri, tipul fiecărui câmp, dimensiune, restricții de integritate)

La nivel fizic, fiecare tabel primește detalii mai precise despre tipurile de date din SGBD (e.g., MySQL):

Student(idStudent INT AUTO\_INCREMENT, Nume VARCHAR(50), Prenume VARCHAR(50), CNP CHAR(13), Nota FLOAT(3,2), ...)

Facultate(idFacultate INT AUTO\_INCREMENT, Nume\_Facultate VARCHAR(50), Adresa VARCHAR(100), numar\_locuri INT, ...)

admitere\_status(idStudent INT, idFacultate INT, status VARCHAR(20), ...)

admin(username VARCHAR(30), password VARCHAR(30), last\_login DATE, ...)  
  
Scripturile de creeare a tabelelor sunt următoarele:  
  
CREATE TABLE `admin` (

`idAdmin` int NOT NULL AUTO\_INCREMENT,

`username` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`last\_login` datetime DEFAULT NULL,

PRIMARY KEY (`idAdmin`)

)  
  
CREATE TABLE `facultate` (

`idFacultate` int NOT NULL AUTO\_INCREMENT,

`Nume\_Facultate` varchar(45) DEFAULT NULL,

`Adresa` varchar(45) DEFAULT NULL,

`numar\_locuri` int DEFAULT NULL,

PRIMARY KEY (`idFacultate`)

)

CREATE TABLE `student` (

`idStudent` int NOT NULL AUTO\_INCREMENT,

`Nume` varchar(45) DEFAULT NULL,

`Prenume` varchar(45) DEFAULT NULL,

`CNP` varchar(45) DEFAULT NULL,

`nota` decimal(4,2) DEFAULT NULL,

`idFacultateOptiune` int DEFAULT NULL,

`Optiune` varchar(100) DEFAULT NULL,

PRIMARY KEY (`idStudent`),

KEY `idFacultate` (`idFacultateOptiune`),

CONSTRAINT `student\_ibfk\_1` FOREIGN KEY (`idFacultateOptiune`) REFERENCES `facultate` (`idFacultate`) ON DELETE CASCADE ON UPDATE CASCADE

)   
  
CREATE TABLE `admitere\_status` (

`idStudent` int NOT NULL,

`idFacultate` int NOT NULL,

`status` varchar(10) DEFAULT NULL,

PRIMARY KEY (`idStudent`,`idFacultate`),

KEY `idFacultate` (`idFacultate`),

CONSTRAINT `admitere\_status\_ibfk\_1` FOREIGN KEY (`idStudent`) REFERENCES `student` (`idStudent`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `admitere\_status\_ibfk\_2` FOREIGN KEY (`idFacultate`) REFERENCES `facultate` (`idFacultate`) ON DELETE CASCADE ON UPDATE CASCADE

)   
  
  
  
  
  
  
  
  
Implementare: limbaje, tehnologii, librării

Pentru implementarea proiectului, am utilizat:

1. **Limbajul Java** – partea de backend (logica aplicației), cu biblioteci standard pentru conectarea la baza de date:
   * java.sql.\* – pentru gestionarea conexiunii cu baza de date (JDBC).
   * javax.swing.\* – pentru interfața grafică minimală (Form-uri, Panel-uri).
2. **SGBD MySQL** – pentru stocarea datelor. S-a folosit driver-ul mysql-connector-j.
3. **Biblioteca iText** – pentru exportul în format PDF.
4. **Alte biblioteci**:
   * java.io.\* – pentru operații de import/export CSV;
   * java.util.\*, java.awt.\* – la partea de colecții și interfață grafică.

**Scopul utilizării** este crearea unei aplicații desktop cu două panouri principale: unul de autentificare și, în funcție de rol, afișarea panourilor de Admin ori Student.

**Modul de utilizare** al bibliotecilor:

* DatabaseManager (clasa Java) se ocupă de conexiune (open/close) și de rularea interogărilor (SELECT, INSERT, UPDATE, DELETE).
* AuthenticationPanel se ocupă de logica de autentificare (student vs. admin).
* DataBase\_UI\_Admin și DataBase\_UI\_Student reprezintă interfețele principale pentru fiecare rol.

**Validarea datelor de intrare** este exemplificată de funcțiile:

* validCNP(String cnp) – verifică dacă string-ul are 13 caractere numerice.
* validID() – parsează textul dintr-un JTextField pentru a obține un int. Aruncă excepție dacă nu e valid.
* validNota() – parsează nota, o transformă în float și verifică dacă se află în intervalul 1–10.
* validOptiune() – verifică dacă facultatea introdusă chiar există în tabela Facultate.

**Tratarea excepțiilor**:

* Toate cazurile în care inputul utilizatorului nu respectă condițiile (ex. ID invalid, CNP nevalid, date lipsă) sunt interceptate prin excepții personalizate. Acestea sunt afișate pe ecran sub formă de mesaje de eroare (folosind, de exemplu, JOptionPane.showMessageDialog).
* Dacă apar erori de conectare la baza de date, se aruncă excepții care sunt capturate și afișate tot prin mesaje de tip „Eroare”.
* Astfel, utilizatorul/administratorul primește mereu feedback clar cu privire la problemele apărute.

**Alternative**:

* Am fi putut folosi un framework Java web (ex. Spring Boot + Thymeleaf) pentru a face o aplicație web.
* Pentru partea de front-end, s-ar fi putut folosi Angular/React, iar pentru back-end, un REST API.
* Pentru generarea de PDF, se puteau utiliza și alte librării (Apache PDFBox, JasperReports etc.).

## Descrierea aplicației (manualul de utilizare)

Loc de inserare capturi de ecran conform cerințelor.\textit{Loc de inserare capturi de ecran conform cerințelor.}Loc de inserare capturi de ecran conform cerințelor.

Manualul de utilizare se împarte pe **două roluri**:

1. **Rolul Admin**
   * **Autentificare**: introduceți username și password în panoul dedicat.
   * **Inserare student**: completați câmpurile (fără ID) și apăsați butonul „Insert”. Datele sunt inserate în tabelul Student.
   * **Update student**: completați ID-ul și câmpurile pe care doriți să le modificați, apoi apăsați „Update”.
   * **Delete student**: introduceți ID-ul studentului și apăsați „Delete”. Se va șterge acel student.
   * **Filter/Update tabela Studenți**: dacă doriți să filtrați, completați câmpurile dorite și apăsați „Filter/Update tabela Studenți”; dacă vreți să vedeți toți studenții, lăsați câmpurile goale și apăsați butonul.
   * **Import CSV**: se deschide un file chooser, se selectează fișierul CSV ce conține datele (Nume, Prenume, CNP, Nota, Opțiune), după care datele vor fi încărcate în baza de date.
   * **Export to Excel**: pentru a exporta lista de studenți sub formă de fișier CSV.
   * **Export to PDF**: exportă statusul admiterii (tabela admitere\_status) în format PDF.
   * **Generare Graph**: la click, se solicită numele facultății, se contorizează câți studenți au fost admiși vs. numărul de locuri, și se afișează un grafic tip „felii de plăcintă” (pie chart).
2. **Rolul Student**
   * **Autentificare**: introduceți Nume, Prenume, CNP în panoul dedicat.
   * **Afișare status**: după autentificare reușită, vi se afișează ID-ul și, într-un tabel, puteți vedea la ce facultate apare studentul și ce status are (admis/respins).
   * **Refacere status**: dacă se face un update de la Admin (ex. a schimbat ceva la note), studentul poate revedea rezultatul final prin simpla reautentificare (pentru actualizarea tabelei).

După fiecare operație (inserare, update, delete), aplicația actualizează statusul automat (admis/respins) prin rularea funcției updateAdmitereStatus() care compară note și locurile disponibile la facultăți.  
  
Panel autentificare:  
  
A screenshot of a computer login screen

Description automatically generated

Panel Admin:

A screenshot of a computer

Description automatically generated

Sectiune introducere date:

A screenshot of a computer

Description automatically generated

Sectiune operatii + tabel Student:  
  
A screenshot of a computer

Description automatically generated

Sectiune tabel facultate + table admitere\_status cu operatiunile de export csv (compatibil si Excel) si PDF. De asemenea e optiune si de graphic pentru o anumita facultate:

A screenshot of a computer

Description automatically generated

Bibliografie

https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html  
https://docs.oracle.com/javase/8/docs/api///?javax/swing/package-summary.html

https://www.geeksforgeeks.org/introduction-to-java-swing/

https://www.w3schools.com/java/

https://chatgpt.com