Management of students
Analysis and Design Document

Student:Dobner Imola
Group:30233

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Proiectarea și implementarea unei aplicații Java pentru administrarea stundeților unei universități.
Aplicația are două tipuri de utilizator: student și administrator/profesor, aceștia fiind nevoiți să furnizeze un nume de utilizator și o parolă pentru a putea fii autentificați și a utiliza aplicația.
Utilizatorul de tip student poate executa următoarele oprații:

- a adăuga/schimba/vizualiza informații personale(nume, CNP, serie buletin, adresa șamd.)
- a crea/schimba/șterge/vizualiza profilul studențesc (număr matricol, grupă, cursuri, note)
- a procesa cursurile, examenele, notele.

Utilizatorul de tip administartor poate executa următoarele oprații:

- CRUD pe datele studenților
- A generea rapoarte/statistici legate de activitate studenților

1.2 Functional Requirements

Cerințele funcționale ale acestui proiect presupun urmatoarele operații: adăugarea, schimbarea si vizualizarea informațiilor despre studenți(date personale și date legate de student în facultate), crearea, ștergerea, modificare și adăugarea conturilor studenților, precum și generarea de statistici. Aceste date sunt funcționale datorită textfield-urilor din interfata grafica.

1.3 Non-functional Requirements

Cerințele non-funcționale ale proiectului sunt validarea datelor de autentificare înainte ca acestea să fie transmise către baza de date și salvate acolo(mai este important și să fie distincte diferitele accese la date de diferiți utilizatori), precum și ca interfața utilizator să fie intuitivă, ușoară de utilizat, cât mai practică.

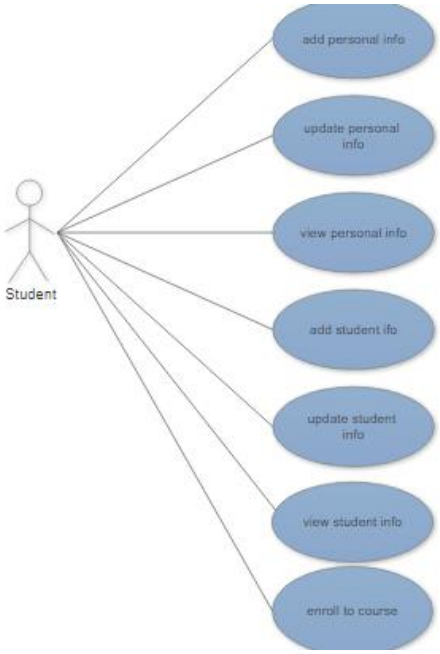
2. Use-Case Model

Utilizatorul de tip student poate executa următoarele oprații:

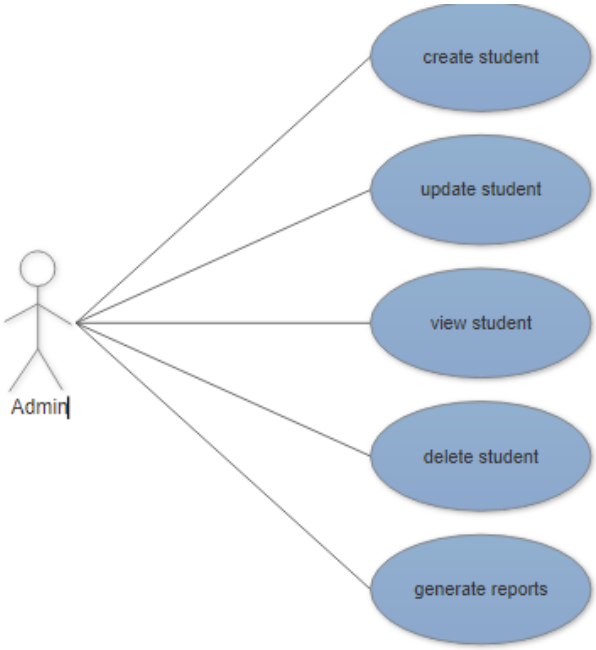
- a adăuga/schimba/vizualiza informații personale(nume, CNP, serie buletin, adresa șamd.)
- a crea/schimba/șterge/vizualiza profilul studențesc (număr matricol, grupă, cursuri, note)
- a procesa cursurile, examenele, notele.

Utilizatorul de tip administartor poate executa următoarele oprații:

- CRUD pe datele studenților
- a generea rapoarte/statistici legate de activitate studenților



Use case: Student oprations
Level: user-goal level
Primary actor: Student
Main success scenario: Se autentifică studentul, din prima fereastră își alege dacă vrea să acceseze date presonale, date legate de profilul de student sau legate de facultate și cursuri. Din următoarea fereastră își allege operația pe care vrea să o săvârșească: adăugare, modificare sau vizualizare, mai apoi aplică schimbările efectute.
Extensions: Studentul nu are cont sau nu se autentifică corect.



Use case: Administrator oprations
Level: user-goal level
Primary actor: Administrator
Main success scenario: Se autentifică administratorul, din prima fereastră își alege dacă vrea să acceseze date legate de studenți sau vrea să creeze anumite statistici. Din următoarea fereastră își allege operația pe care vrea să o săvârșească: adăugare, modificare, ștergere sau vizualizare, mai apoi aplică schimbările efectute.
Extensions: Administartorul nu se autentifică cu datele necesare corect.

3. System Architectural Design

3.1 Architectural Pattern Description

Componentele din modelul de arhitectură Layer sunt organizate în straturi orizontale, fiecare strat realizând un anumit rol în cadrul aplicației.
Una dintre caracteristicile puternice ale modelului arhitectural stratificat este separarea preocupărilor între componente. Componentele dintr-un anumit strat se ocupă numai de logica care se referă la acel strat.
În pachetul “User Interface” am folosit șablonul arhitectural Model View Controller.

3.2 Diagrams

Diagrama de pachete:

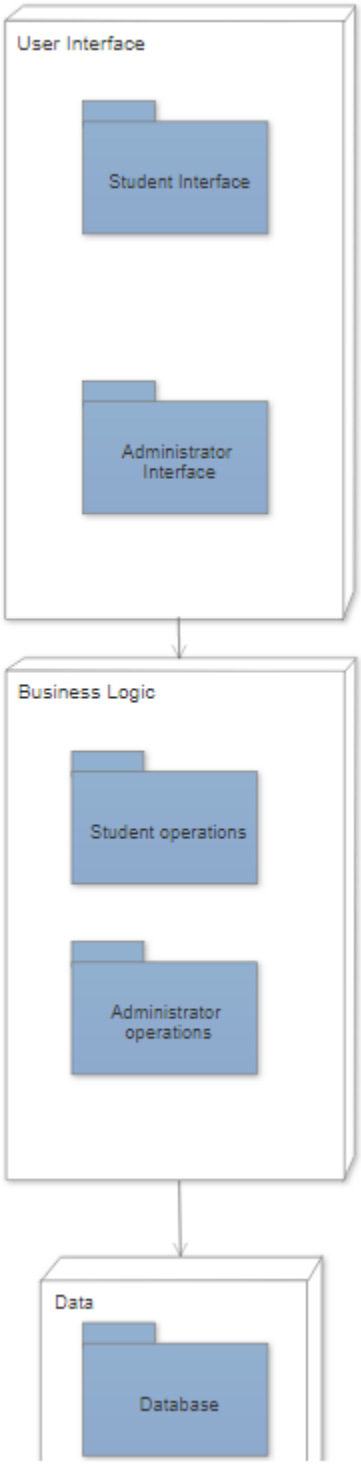
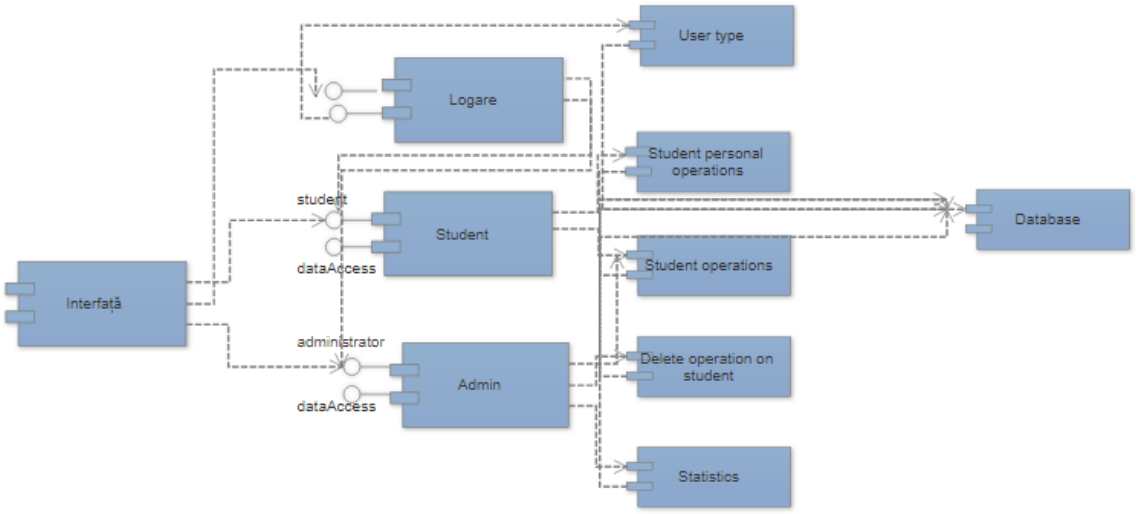
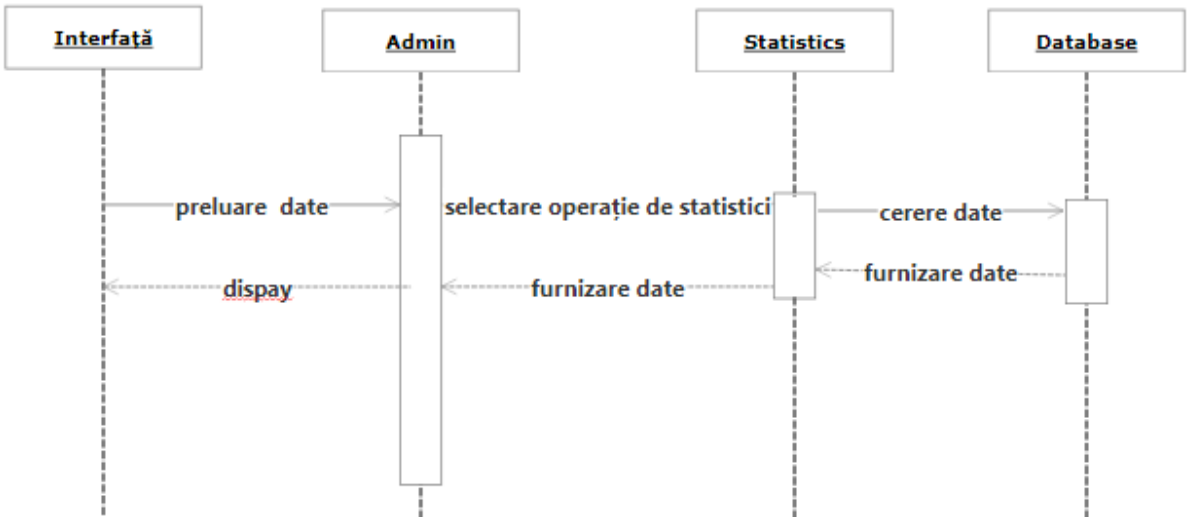


Diagrama de componente:



4. UML Sequence Diagrams

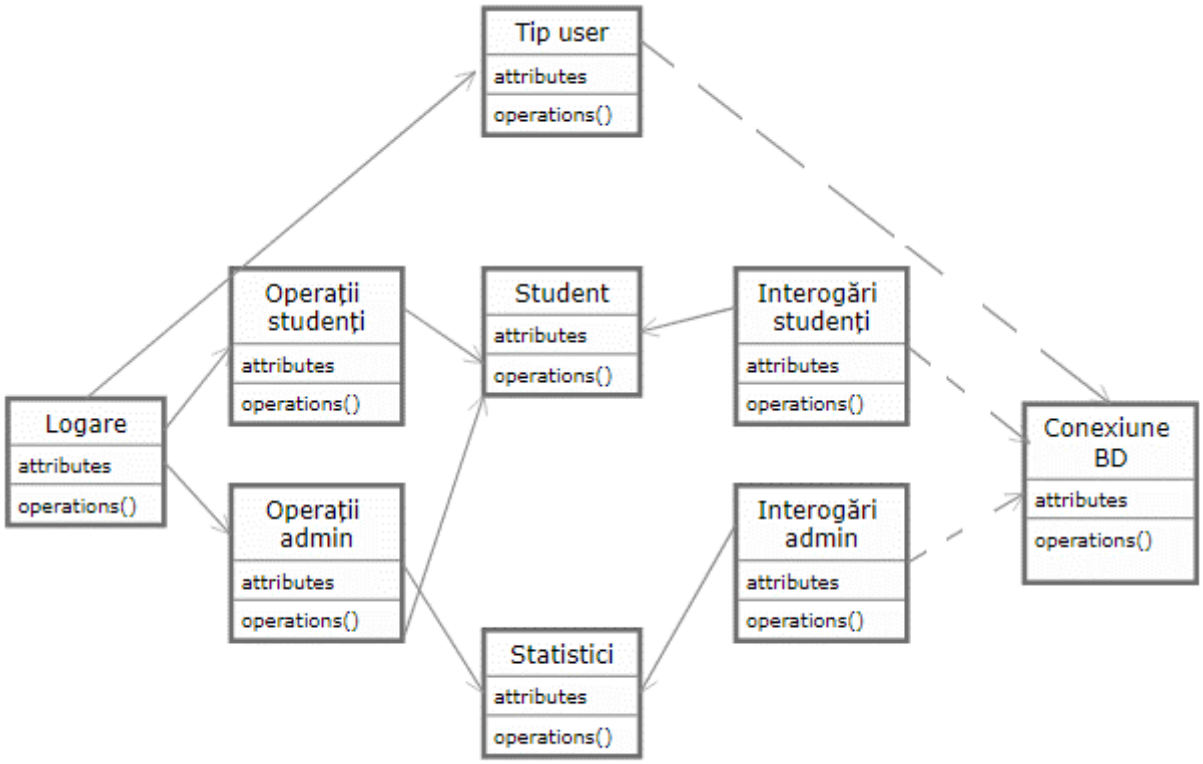


5. Class Design

5.1 Design Patterns Description

Layers: Componentele sunt organizate în straturi orizontale, fiecare strat realizând un anumit rol în cadrul aplicației.
MVC: Scopul lor este de a ajuta la structurarea codului și de a separa preocupările unei aplicații în trei părți.

5.2 UML Class Diagram



6. Data Model

[Present the data models used in the system's implementation.]

7. System Testing

[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]

8. Bibliography