University Management System

Analysis and Design Document

Student:Ciontu Mihail

**Group:30233**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

*Aplicatia propusa este bazata pe creearea unui mediu de management al elevilor apartinand departamentului de Computer Science in cadrul TUCN.*

*Aplicatia dispune de doua tipuri de utilizatori: student si profesor.Studentul isi creeaza profilul cu diferite date,se inscrie la diferite examene,isi vede diferite note iar profesorul poate face modificari pe datele elevilor,cel mai relevant fiind datele care fac referire la evaluarea acestora.*

# Functional Requirements

*In meniul de login,utiliziatorilor le este cerut sa se autentifice,astfel acestia sunt impartiti in doua categorii:student si profesori.Fiecare pot efectua operatii diferite in cadrul programului.*

*Studentul trebuie sa poata efectua urmatoarele:*

*-Add/update/view pe informatiile personale (CNP,adresa,nume prenume)*

*-Create/update/delete/view pe informatiile de student(ID nr,grupa,cursuri si note)*

*-Posibilitatea de inrolare la diferite curusuri si examene.*

*Profesorul are urmatoarele optiuni:*

*-CRUD pe informatiile studentilor(Create/Read/Update/Delete).*

*-Inscrierea unor rapoarte pentru anumite perioade incluzand activitatile studentului.*

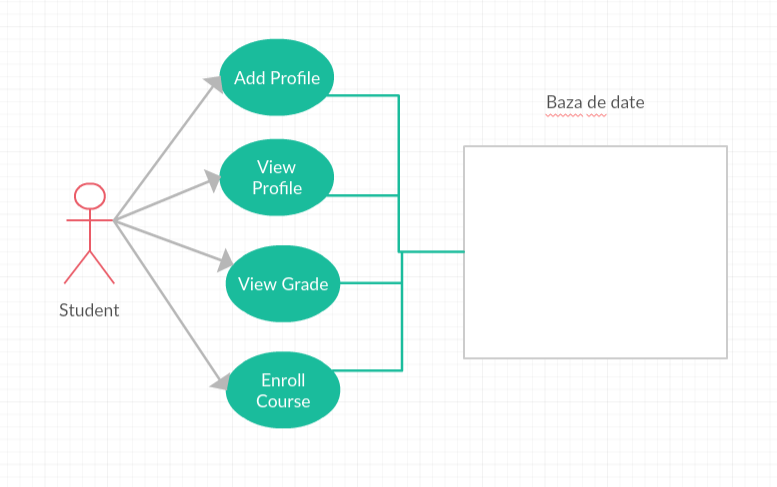
# Non-functional Requirements

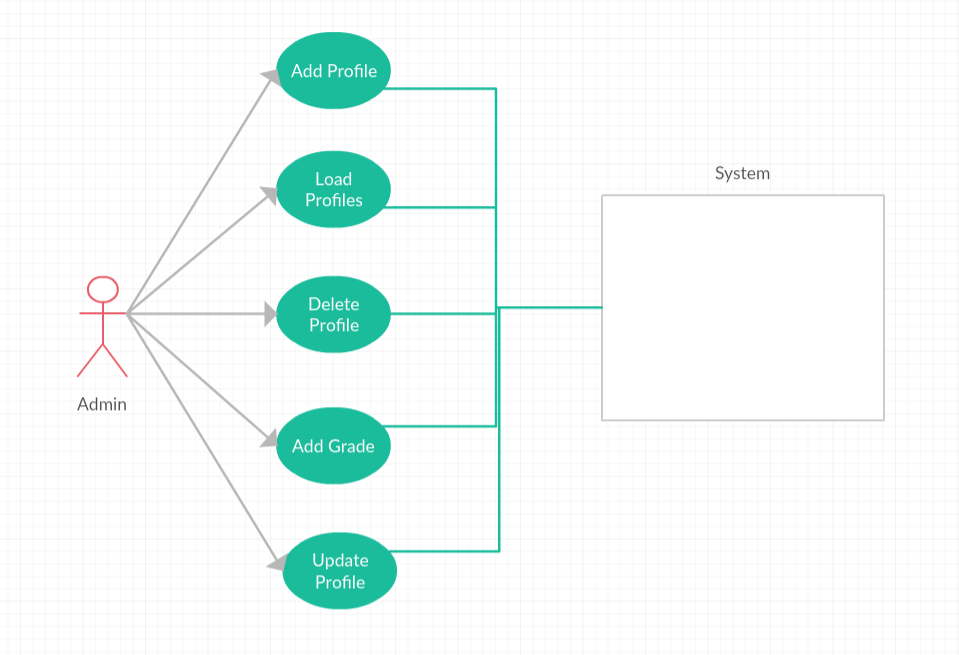
*Maintainability-In baza de date vor fii tinute datele aplicatiei,acestea pot fi atacate de virusi si sistemul poate crapa.Este necesar sa ca baza de date sa aiba backup-uri.*

*Safety-Datele personale fiecarui student vor fi private,nici un alt utilizator normal nu va putea accesa acestea..*

*Interfata aplicatiei-aceasta va avea un login cu user si parola personala si unica.*

2. Use-Case Model





3. System Architectural Design

**3.1 Architectural Pattern Description**

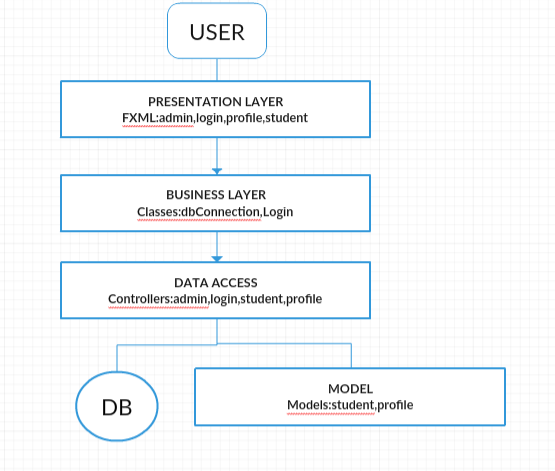
*Layer Pattern:*

*Programele vor fi structurate in grupuri,fiecare din aceste grupuri fiind la un nivel particular de abstractie.Layeru de pe nivelul mai inalt furnizeaza servicii urmatorului layer.Cele mai commune layer sunt: Presentation layer(UI),Application Layer,Business logic layer si data acces layer.*

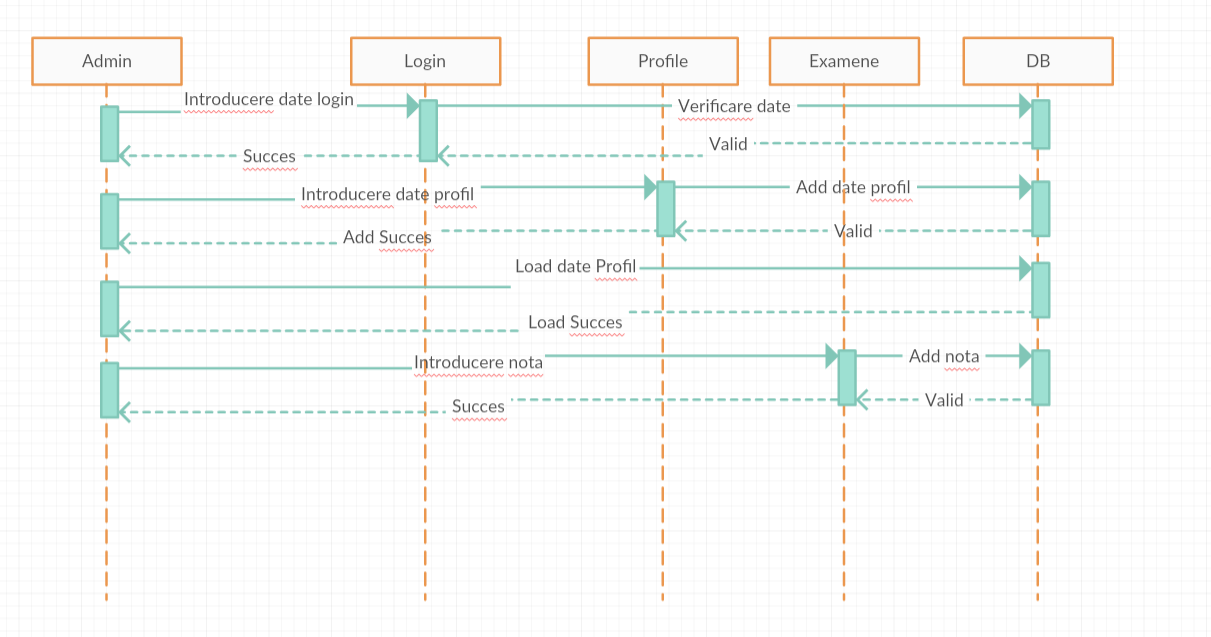
*MVC Pattern:*

*Cu ajutorul unui MVC putem separa sarcinile aplicatiei,astfel cu un niste Controllere putem modifica ,adauga si sterge date din baza de date.In modele tinem datele care vor fi accesate de controlere iar partea de view este interfata aplicatiei.*

**3.2 Diagrams**

4. UML Sequence Diagrams



5. Class Design

**5.1 Design Patterns Description**

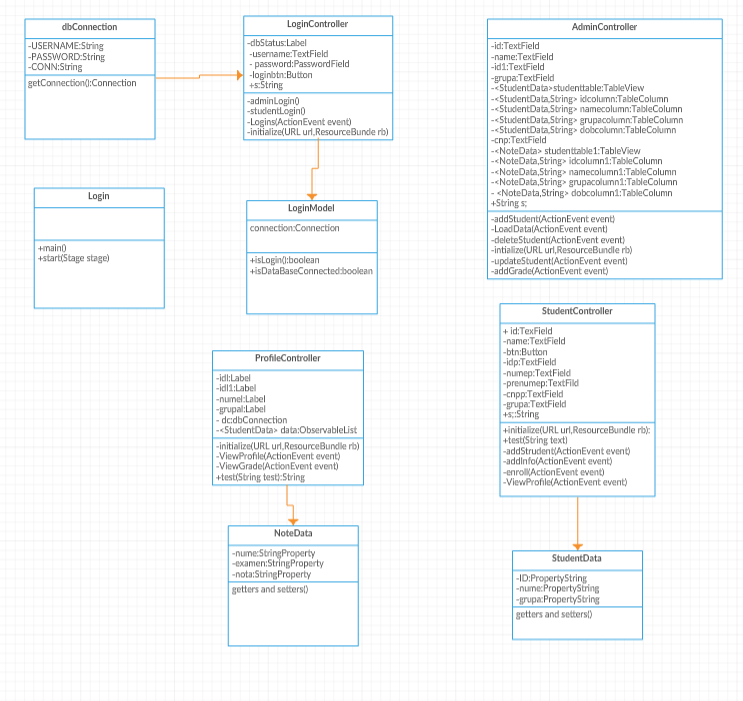
*Observer:vom implementa functii care vor face unele date dependente de altele,astfel cand vom modifica ceva,spre exemplu cursurile la care va participa un student,aceste date vor produce modificari in diferite locuri.*

*Builder:acest pattern ne va ajuta sa construim unele relatii de agregare sau compositie intre clase.*

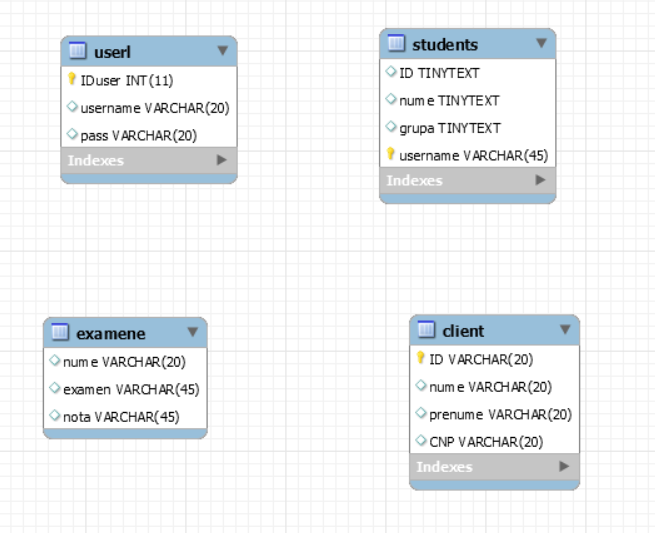
*Adapter:va fi folosit in unele locuri cand vrem sa afisam informatii din baza de date*

*Singleton: o sa fie folosit in comunicarea cu baza de date*

**5.2 UML Class Diagram**

**

6. Data Model



7. Bibliography

<https://medium.com/educative/the-7-most-important-software-design-patterns-d60e546afb0e>

<https://www.scribd.com/doc/14649618/SRS-of-University-Management-System-by-Balwinder-Singh-Vehgal>

<https://www.academia.edu/9385119/UML_diagram_for_University_Information_System> --- uml

<https://web.csulb.edu/~mopkins/cecs493/sdsfinaledit-140325142607-phpapp01.pdf> --architect

https://pdfs.semanticscholar.org/53ae/ebe9c51aed7384b4f06e6359141958dd4ae2.pdf