ASSIGNMENT A2 – Student Management System

Analysis and Design Document

Student: Debre Lóránd - Sándor

**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 4

4. UML Sequence Diagrams 6

5. Class Design 7

6. Data Model 8

7. System Testing 8

8. Bibliography 8

1. Requirements Analysis

# Assignment Specification

Design and implement a Java application for the management of students in the CS Department at TUCN. The application should have two types of users (student and teacher/administrator user) which have to provide a username and password in order to use the application.

# Functional Requirements

The regular user can perform the following operations:

* Add/update/view client information (name, identity, card number, personal numerical code, address, etc.).
* Create/update/delete/view student profile (account information: identification number, group, enrollments, grades).
* Process class enrolment (enroll, exams, grades).

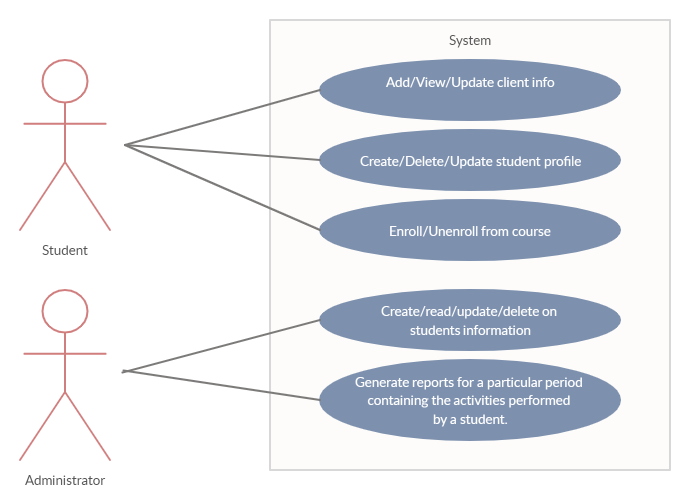
The administrator user can perform the following operations:

* CRUD on student information.
* Generate reports for a particular period containing the activities performed by a student.

# Non-functional Requirements

* **Securitate:** aplicatia trebuie sa faca diferenta intre student si administrator folosind tehnica de protectie
* **Utilizabilitate:** aplicatia trebuie sa fie usor de folosit pentru toate lumea
* **Extensibilitate:** folosind layerul Architectural Pattern trebuie sa fie usor de extins
* **SOLID:** aplicatia trebuie sa foloseasca principiile SOLID

2. Use-Case Model



**Fig. 1**: Use-Case model

Aplicatia are urmatoarele actori: **Student** (are urmatoarele posibilitati: *Add/View/Update client info; Create/Delte/Update student profile; Enroll/Unenroll from course*)si **Teacher** (are urmatoarele posibilitati: *Crud on students info; Generate reports for a particular period of contatining the activities performed by student*).

Exemplu:

**Use case:** add (create) client info

**Level:** user goal level

**Primary actor:** student

**Main success scenario:** Trebuie sa fie introdus un username si o parola pentru contul nou, si toate informatiile necesare (CNP, Adresa, etc)

**Exenstions:** Username-ul respectiv este ocupat si/sau parola respectiva este prea usora

3. System Architectural Design

**3.1 Architectural Pattern Description**

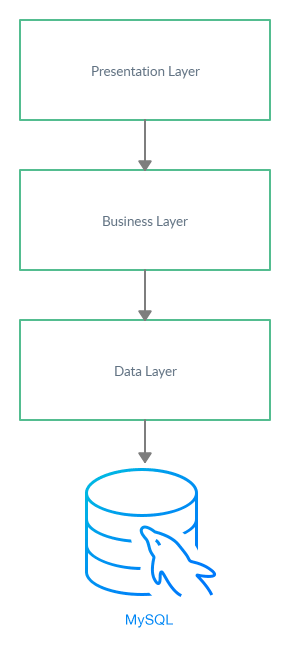
In acest proiect folosesc pe **Layered Application Pattern:**

Aplicatia va fi impartita in urmatoarele 3 componente:

* Presentation layer (sau UI layer)
* Business layer
* Data layer

**3.2 Diagrams**

**a) Diagrama conceptuala:**



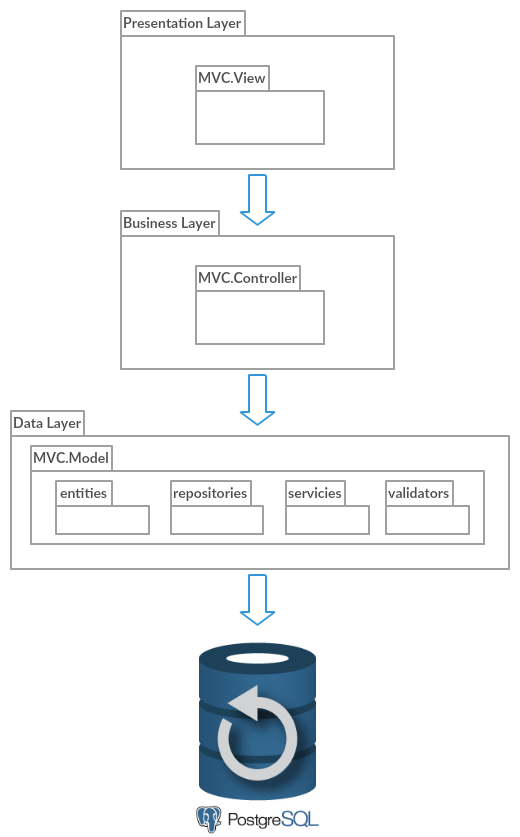
**Fig.2:** Diagrama conceptuala a sistemului

**Presentation layer:** contine o functionalitate catre utilizator cu sistemul

**Business layer:** implementeaza functionalitatea sistemului

**Data layer:** poate sa faca conexiune cu baza de data (MySQL) si returneze datele catre business layer, aici sunt stocate entitatiile ale sistemului

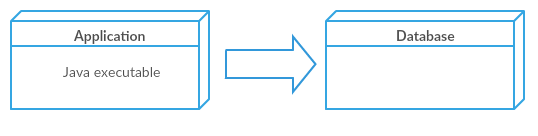
**b) Diagrama de pachet:**



**Fig.3:** Diagrama de pachet

Paternul Layered se vede si pe diagrama de pachet, deoarece toate separarile reiese fiindca sunt 3 pachete mari (Presentation – Business – Data).

**c) Diagrama de deployment:**

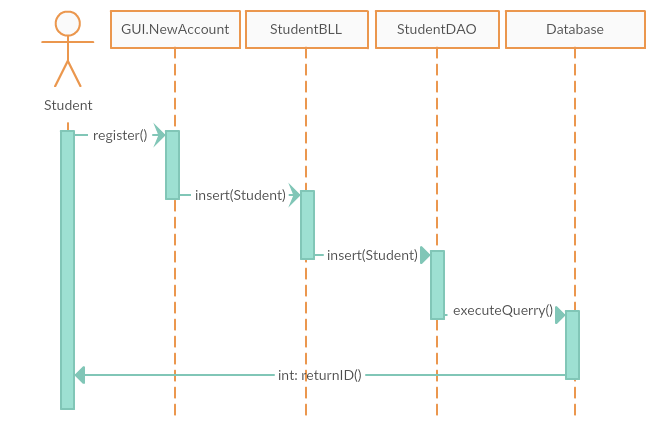


**Fig.4:** Diagrama de deployment

Din diagrama de deployment rezulta ca aplicatia noastra are doar 2 componente si o conexiune de la componenta **Application** catre **Database**. Aplicatia de Java executable este o aplicatie de Desktop si Database este o baza de date SQL.

4. UML Sequence Diagrams

Daca un utilizator vreau sa faca un cont de Student nou, trebuie sa parcurge toate etapele urmoatoare:



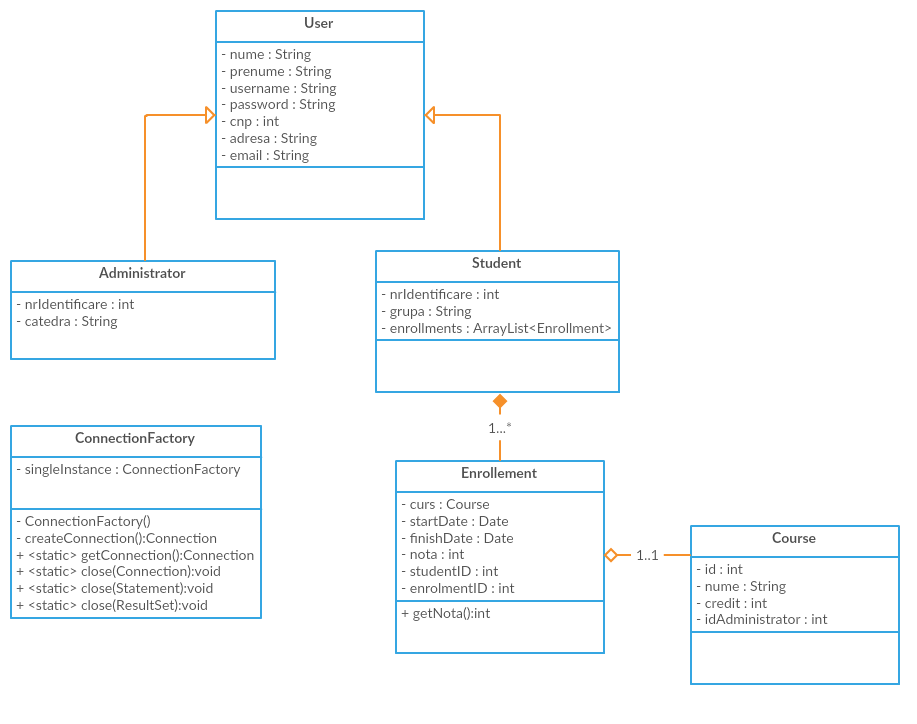
**Fig.5:** Diagrama de secventa UML pentru Create Student Account

5. Class Design

**5.1 Design Patterns Description**

Putem sa folosim paternul Factory si paternul Singleton, deoarece trebuie sa facem o conexiune cu baza de date MySQL folosind obiectul ConnectionFactory si conexiunea se realizeaza folosind aceste patternuri.

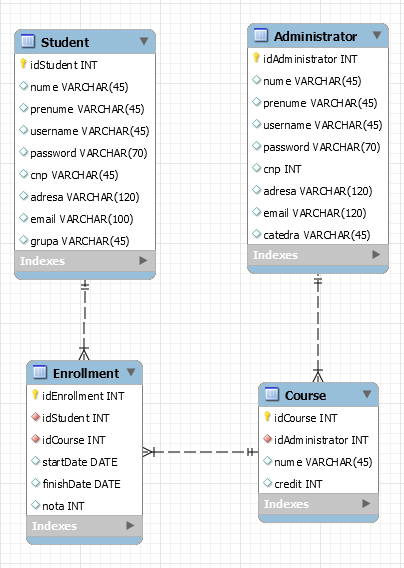
**5.2 UML Class Diagram**



**Fig.6:** Diagrama UML

6. Data Model

Datele sunt stocate intr-o baza de date de tip MySQL.



**Fig.7:** Diagrama baza de date PostgreeSQL

7. System Testing

Strategia de testare folosita:

* Unit testing: predefinim o bucata de cod pe care o executam folosind toolul JUnit si rezulta daca testare s-a finalizat cu succes sau cu fail

8. Bibliography

* <https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff650706(v=pandp.10)>
* <https://en.wikipedia.org/wiki/Command_pattern>
* <https://www.tutorialspoint.com/junit/junit_test_framework.htm>

