The management of students in the CS Department

Analysis and Design Document

Student: Pop Raluca

**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 4

3. System Architectural Design 4

3.1 Architectural Pattern Description 4

3.2 Diagrams 5-6

4. UML Sequence Diagrams 7

5. Class Design 7

5.1 Design Patterns Description 7-8

5.2UML Class Diagram 8

6. Data Model 8-9

7. System Testing 9

8. Bibliography 9

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 a password in order to use the application.

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, enrolments, grades).

- Process class enrolment (enroll, exams, grades).

**Requirements**

- Create the analysis and design document

- Implement and test the application.

# Functional Requirements

To login on the application, every user has a username and a password to provide its identity. Depending on the user who is logged in, it will appear a different interface which has the possibility to do operations on it. The admin/ teacher user can update a students grade or delete him and the student can see his grades or change different personal information.

The requirements for the student:

- Create/update/delete/view on own profile (account information: identification number, group, enrolments, grades).

- Process class enrolment (enroll, exams, grades).

The requirements for the admin/teacher:

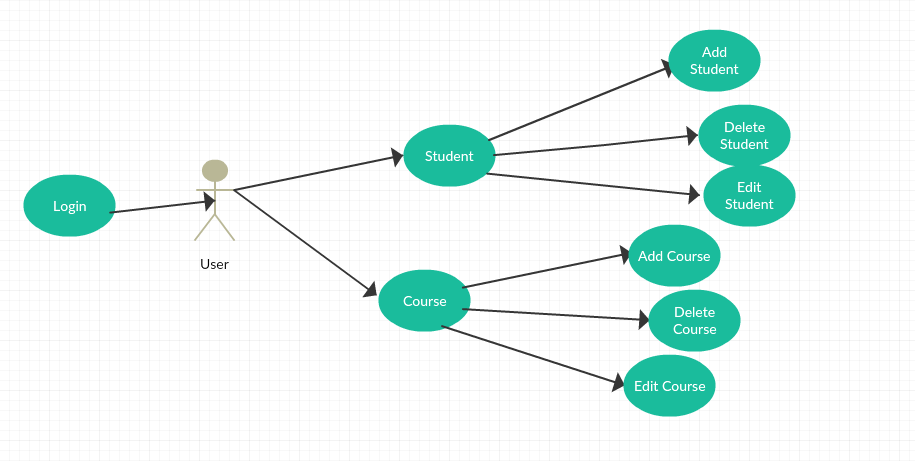
- CRUD on students information.

- Generate reports for a particular period containing the activities performed by a student

# Non-functional Requirements

This application will use PostgreSQL as a relational database, which should update when a change appears. Also, the passwords will be stored in the database.

2. Use-Case Model



Use-Case description format:

Use case: Login

Level: Summary level

Primary actor: User

Main success scenario: The actor add Student profile and Course profile

Extensions: scholarships, school situation of the last years

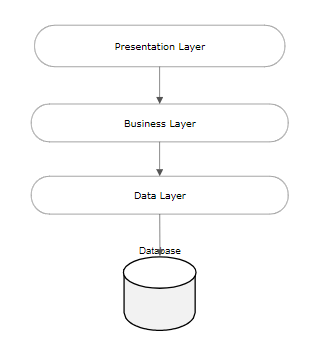
3. System Architectural Design

**3.1 Architectural Pattern Description**

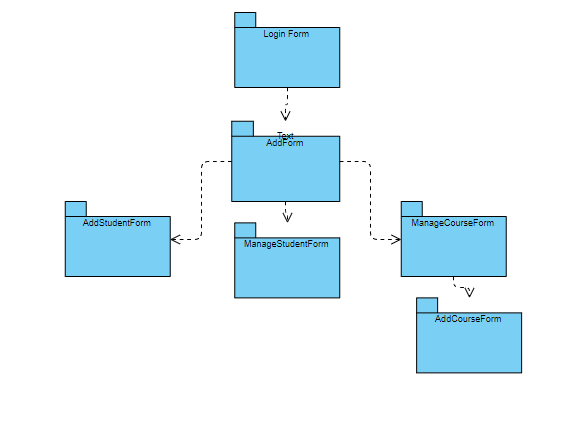
The following applications design is based on layered architecture pattern, otherwise known as the n-tier architecture pattern.  The layered architecture pattern closely matches the traditional IT communication and organizational structures found in most companies, making it a natural choice for most business application development efforts.

Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request.

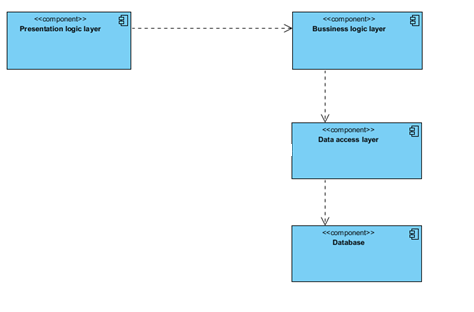
**3.2 Diagrams**



* **Presentation layer**. This layer contains the user oriented functionality responsible for managing user interaction with the system, and generally consists of components that provide a common bridge into the core business logic encapsulated in the business layer.
* **Business layer**. This layer implements the core functionality of the system, and encapsulates the relevant business logic. It generally consists of components, some of which may expose service interfaces that other callers can use**.**
* **Data layer**. This layer provides access to data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services. The data layer exposes generic interfaces that the components in the business layer can consume.

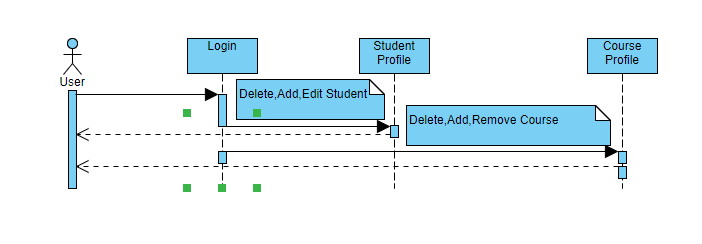
****

Package diagram



Component Diagram

4. UML Sequence Diagrams



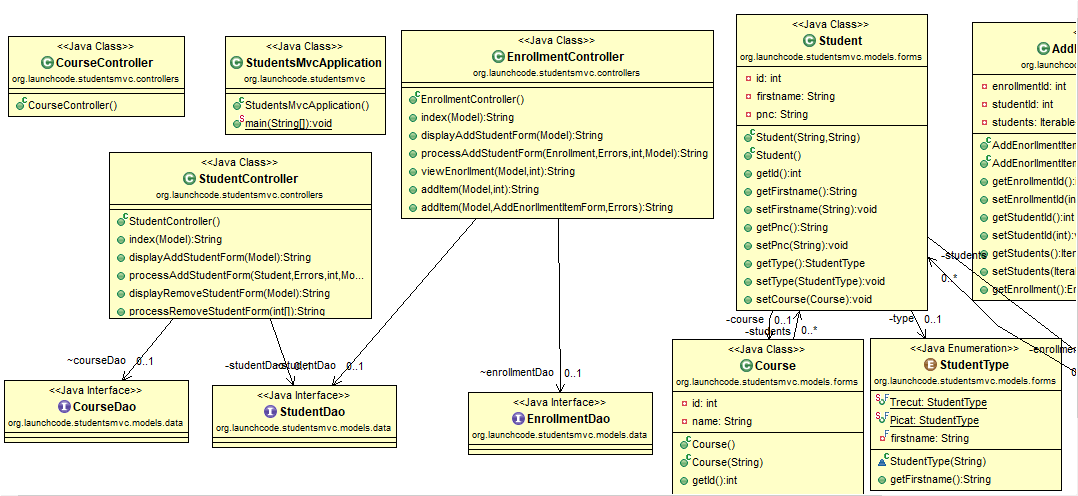
5. Class Design

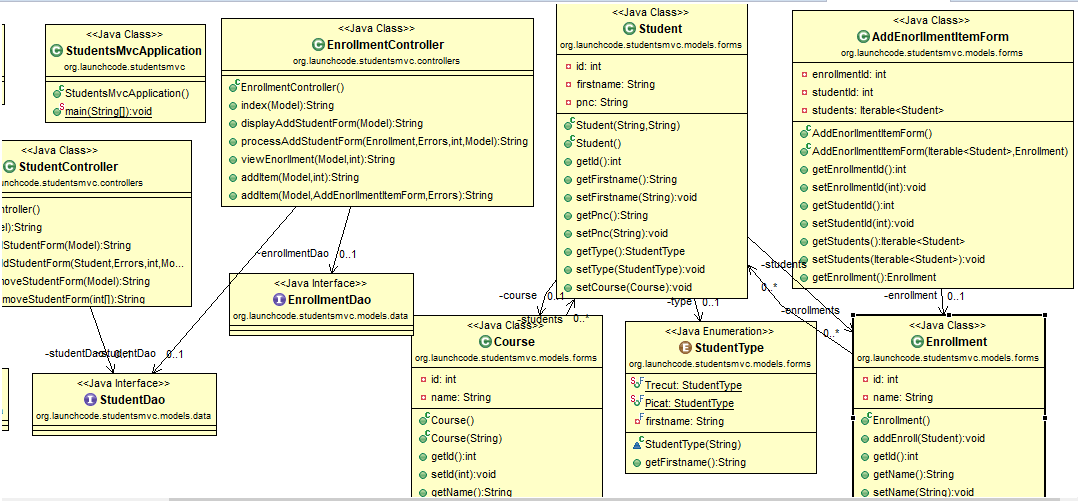
**5.1 Design Patterns Description**

Builderis a creational design pattern that lets you construct complex objects step by step. The pattern allows you to produce different types and representations of an object using the same construction code

Factory Method is a creational design pattern that provides an interface for creating objects in a super-class, but allows sub-classes to alter the type of objects that will be created. The Factory Method pattern suggests that you replace direct object construction calls (using the new operator) with calls to a special factory method

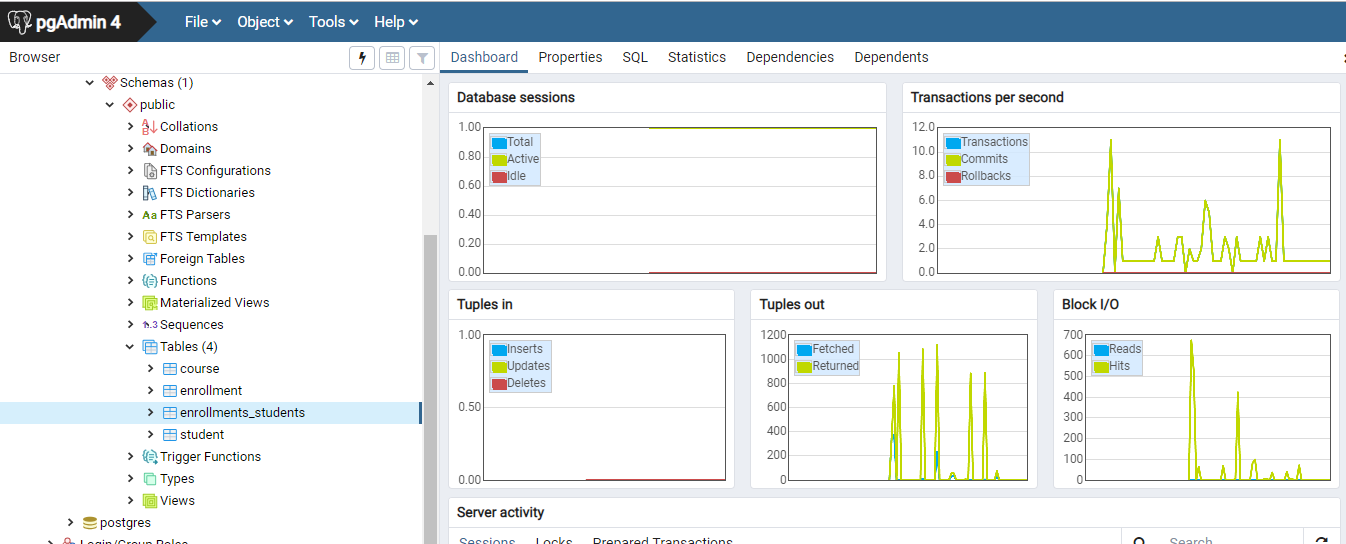
**5.2 UML Class Diagram**





6. Data Model

The data model contains four tables: Student, Course, Enrollment.



8. Bibliography

<https://spring.io/projects/spring-boot>

<https://www.tutorialspoint.com/spring_boot/index.htm>

<https://www.youtube.com/watch?v=EeAMVMt2vMU>

<https://www.youtube.com/watch?v=inhuiXTYUpI>

<https://www.youtube.com/watch?v=fuQJ5oh08_0>

<https://online.visual-paradigm.com/tutorials/use-case-diagram-tutorial/>

<https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

<https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff650706(v=pandp.10)>

<https://en.wikipedia.org/wiki/Software_design_pattern>