Assignment 2

Student: Andreea Sabina Lazaroiu

**Group: 30432**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 4

4. UML Sequence Diagrams 5

5. Class Design 6

6. Data Model 7

7. Bibliography 7

1. Requirements Analysis

# Assignment Specification

Design and implement a Java application for the management of students in the CS Department at TUCN.

# Functional Requirements

The application should have two types of users (student and teacher/administrator user) .

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).

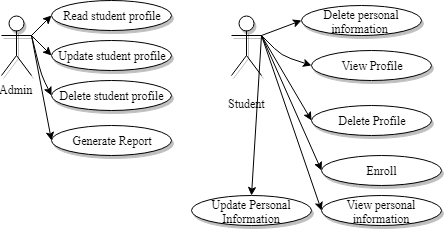
- Process class enrolment (enroll, exams, grades).

The administrator user can perform the following operations:

- CRUD on students information.

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

2. Use-Case Model



Use case: Update Personal Information

Level: User-goal level

Primary actor: Student

Main success scenario:

1. The student chooses to update his personal information.
2. The student modifies the current information.
3. The updated information is saved.

Extensions: The user selects to cancel the operation.

Use case: Generate report

Level: User-goal level

Primary actor: Teacher

Precondition: The teacher has to select a student.

Main success scenario:

1. The teacher selects a student.
2. The teacher generates a report based on the student’s account information.

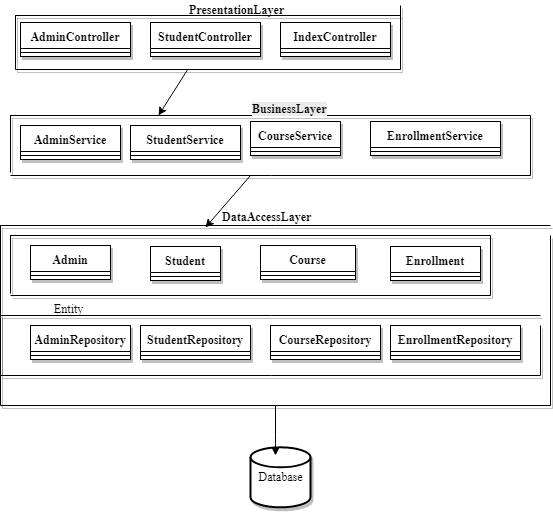
3. System Architectural Design

* 1. **Architectural Pattern Description**

This application will be implemented based on two architectural patterns: Layered Architecture Pattern and Model View Controller Pattern.

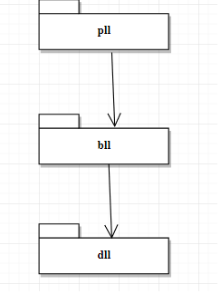
The Layered Architecture Pattern logically separates the components on layers that communicate with each other.

The idea in the MVC Architecture Pattern is to separate the user interface (the Presentation in the previous example) into a View (creates the display, calling the Model as necessary to get information), and Controller (responds to user requests, interacting with both the View and Controller as necessary).

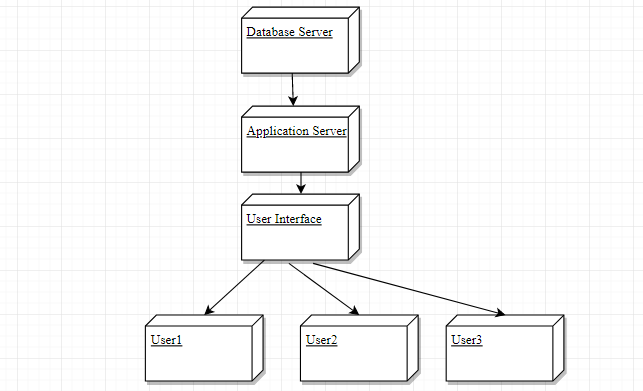
**

* 1. **Diagrams**

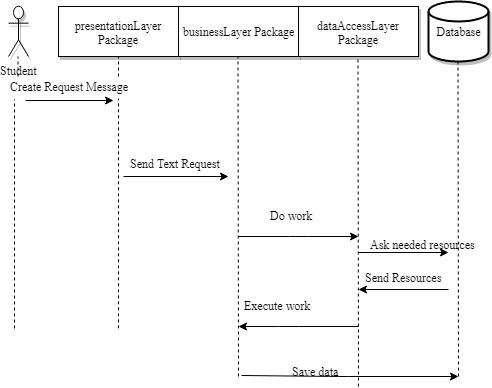
Package Diagram:



Deployment Diagram:



4. UML Sequence Diagrams



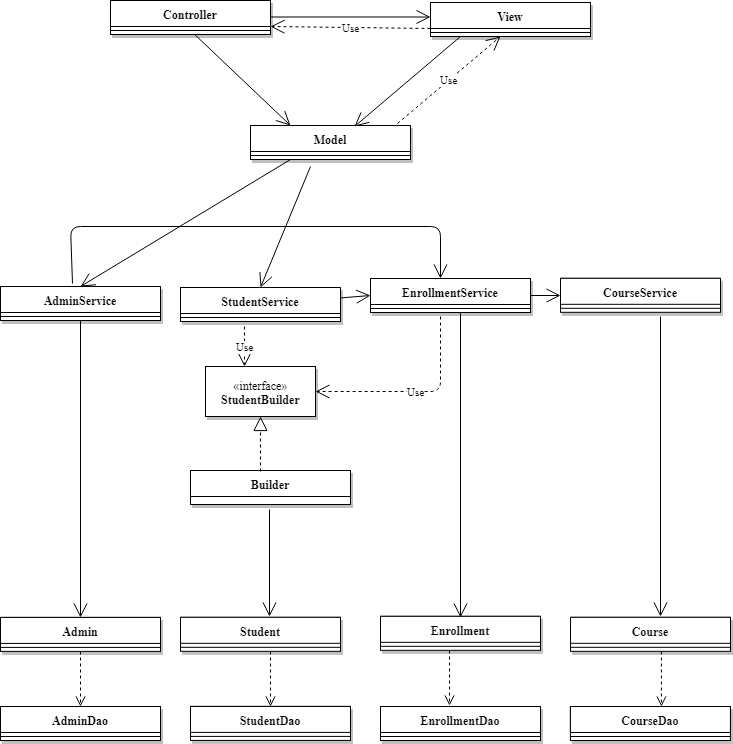
5. Class Design

**5.1 Design Patterns Description**

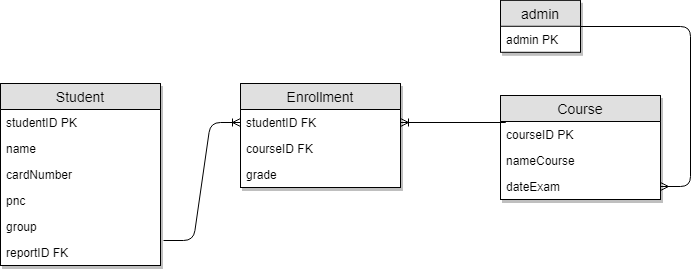
Builder Pattern: Separate the construction of a complex object from its representation so that the same construction process can create different representations.

This Pattern is used explicitly for the Student class because this class has a significant number of instance variables.

**5.2 UML Class Diagram**

**

6. Data Model

**

7. Bibliography

<https://sourcemaking.com/design_patterns/builder>

<http://www.leepoint.net/GUI/structure/40mvc.html>