Assignment 1

Student: Alexandra Paiu

**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

# 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.

# Functional Requirements

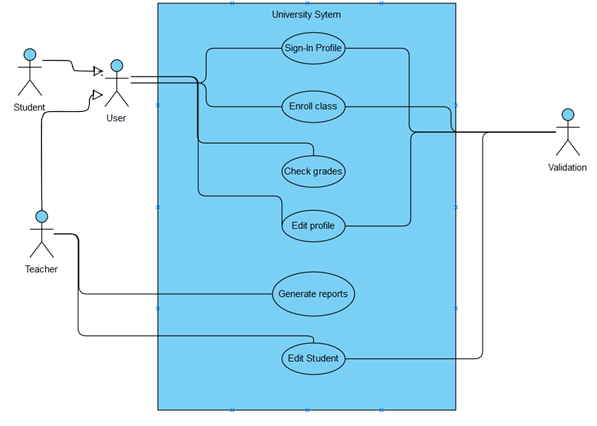
The regular user can perform the following operations: - Add/update/view client information (name, identity card number, personal numerical code, address, etc.). - Create/update/delete/view student profile (account information: identification number, group, enrolments, grades). - Process class enrolment (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.

# Non-functional Requirements

The data will be stored in a relational database, also it will give the user pop-ups about the security status (in case of invalid entries), it will be portable and runnable on almost any personal computers.

2. Use-Case Model



Use case: Student registration for classes

Level: User-goal level [1]

Primary actor: User

Main success scenario:

* User logs-in / signs-up with his correct data (mail and password);
* He checks his grades, courses;
* If the user sings with a teacher mail, he can generate reports;
* All data added or modified is check by the program, if it is correct, the new data is saved and displayed to the user;
* User closes the application.

Extensions:

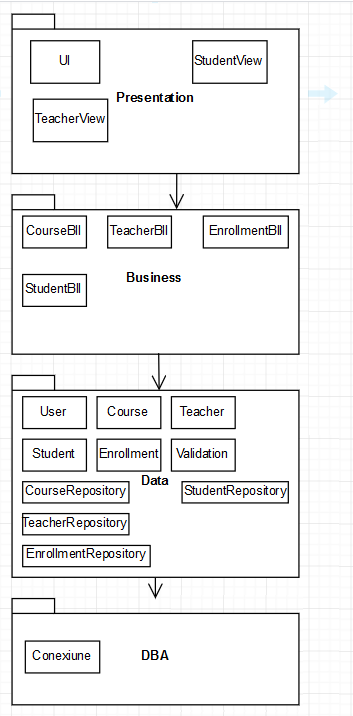
* A student does the log-in procedure, with his correct data, in this case se can enter the application, otherwise he will receive an invalid error message;
* He wants to enroll a course, search for it and with the right password enrolls, else he will be declined;
* He can see his grades or when the test will be up, if he tries to modify the grade, his gesture will have no effect;
* He logs out or closes the application.

3. System Architectural Design

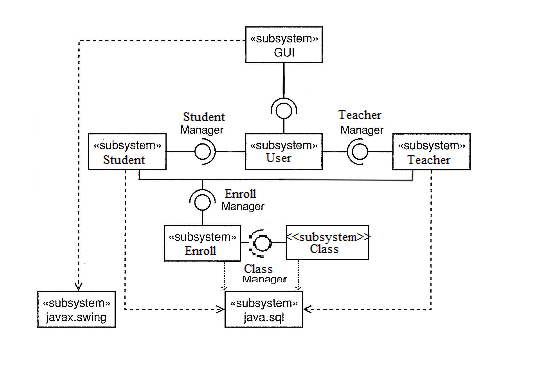
**3.1 Architectural Pattern Description**

Pattern used to implement the application:

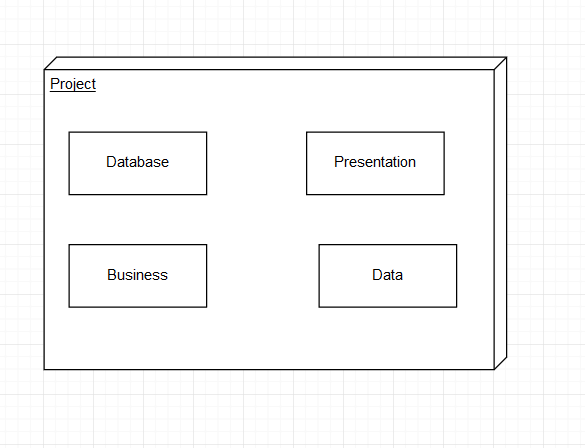
* Layer Pattern Architecture, it is used to separate the classes into 4 main categories, presentation, service, business and database access. In each of those categories will be stored classes related to it’s main purpose, for example, I will not store the updates and deletes for users profile in presentation layer, where will be stored the main implementation of how the application will look like. In the second layer, the service one I will try and put the users and their activities, follow by working with the data extracted from database access layer in the business layer and what it should do.

* 1. **Diagrams** 
     1. **Package Diagram**
     2. 

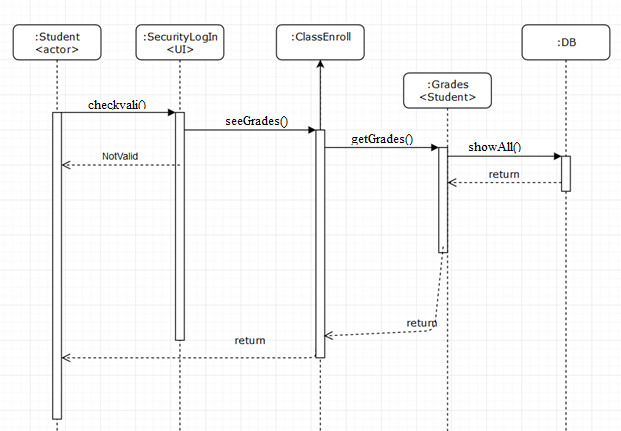
**Component Diagram**

****

* + 1. **Deployment Diagram**



4. UML Sequence Diagrams

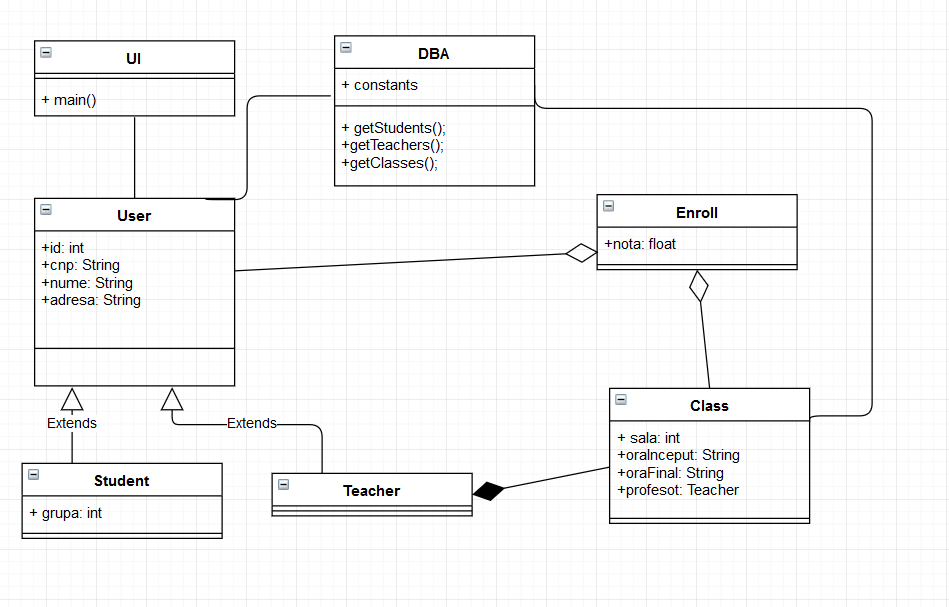


5. Class Design

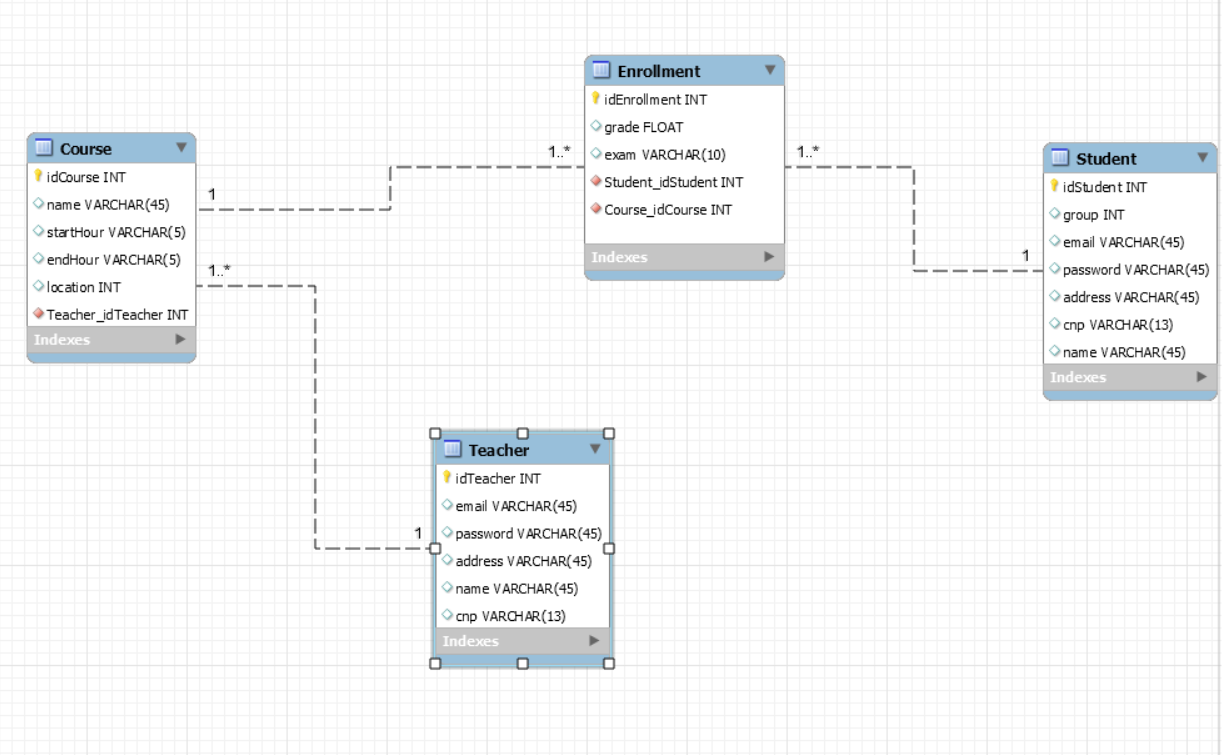
**5.1 Design Patterns Description**

Next week, during implementation I will decide.

**5.2 UML Class Diagram**



6. Data Model



7. System Testing

8. Bibliography

* [1] <https://pjhobday.wordpress.com/2010/05/28/setting-use-case-goal-levels>
* [1] <https://wiki.nci.nih.gov/display/seminfra/Use+Case+Leveling+Definitions>
* <https://is.muni.cz/el/1433/podzim2014/PB007/um/sem/en_files/12/pb007-12.pdf>
* <https://en.wikipedia.org/wiki/Non-functional_requirement>
* <http://www.cs.uregina.ca/Links/class-info/215/erd/>
* <https://www.callicoder.com/javafx-registration-form-gui-tutorial/>
* <https://o7planning.org/en/10857/javafx-scrollpane-tutorial>
* https://docs.oracle.com/javafx/2/ui\_controls/table-view.htm