Assignment 1

Analysis and Design Document

Student: Boros Hanniel
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 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 (students and teacher/administrator user) which have to provide a username and a password in order to use the application.

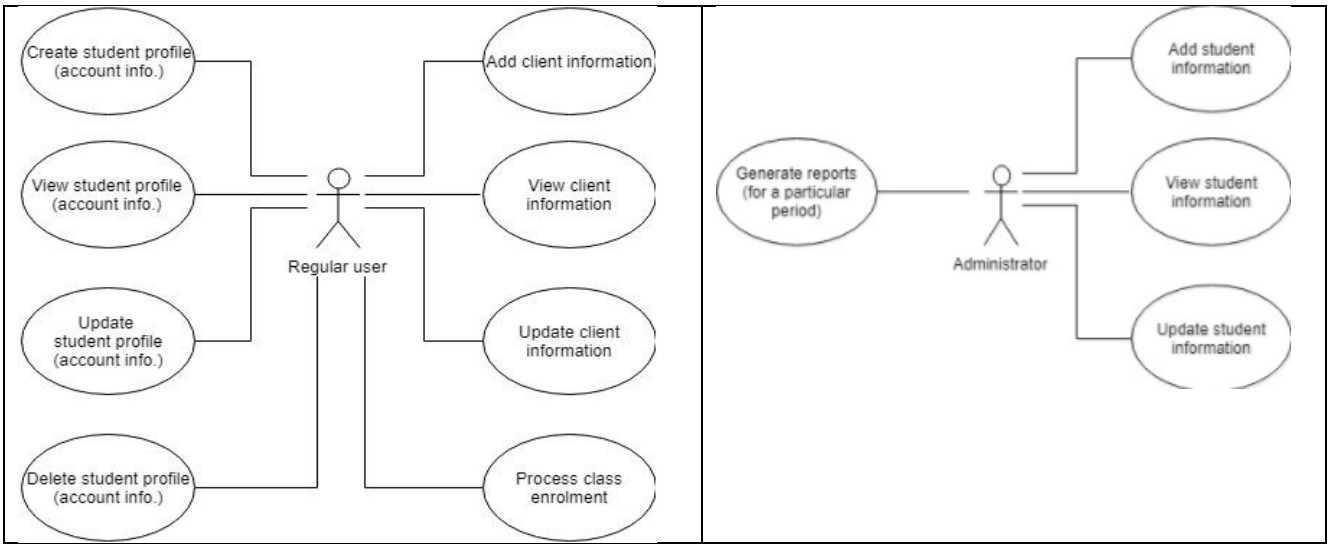
1.2 Functional Requirements

- Authentication
- Edit client information
- Edit profile
- Process class enrolment
- Generate reports

1.3 Non-functional Requirements

- Usability
- Performance (response time < 1s)
- Security
- Data integrity
- Documentation

2. Use-Case Model



One use case description
Use case: Update client information
Level: sub-function
Primary actor: student <a role name for the actor who initiates the use case>
Main success scenario: <the steps of the main success scenario from trigger to goal delivery>
Student selects Update Personal Information option from menu
Student updates the necessary information (name, identity card number, personal numerical code, address)

Student presses Update button, finishing update.
Student waits a short time while information is updated in the database.
Information is updated and student is notified.
Extensions:
Alternate scenario for failure:
if connection to database fails for some particular reasons, or an Exception is encountered, the information may not be updated.

3. System Architectural Design

3.1 Architectural Pattern Description

Layers Architectural Pattern
The architectural pattern used in this project is the Layers architecture, otherwise known as the n-tier architecture pattern. This architectural pattern helps us to structure applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction.

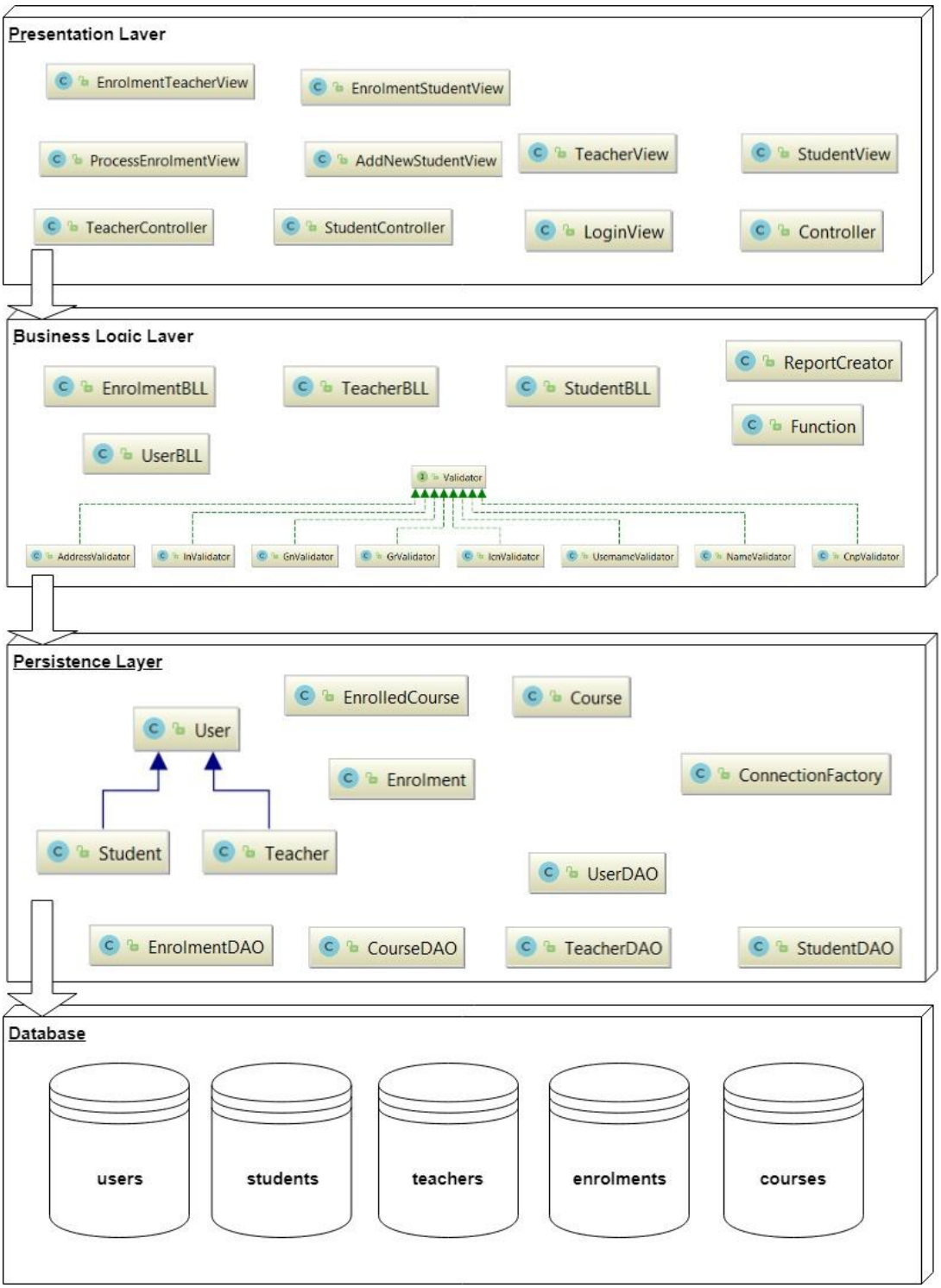
Presentation layer
This layer contains the user oriented functionality responsible for managing user interaction with the system, and generally consists of components that provide a common bridge into the core business logic encapsulated in the business layer.

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.

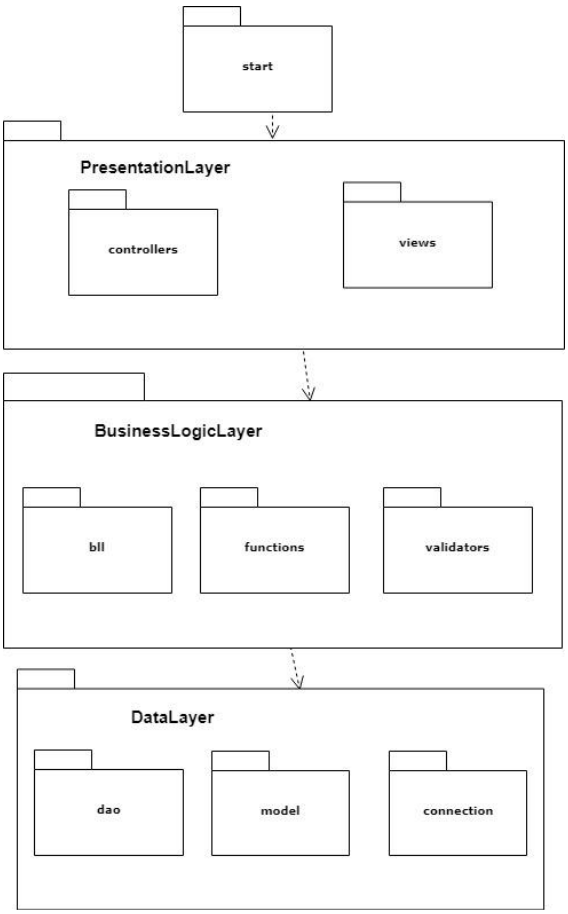
Data layer
This layer provides access to data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services. The data layer exposes generic interfaces that the components in the business layer can consume.

3.2 Diagrams

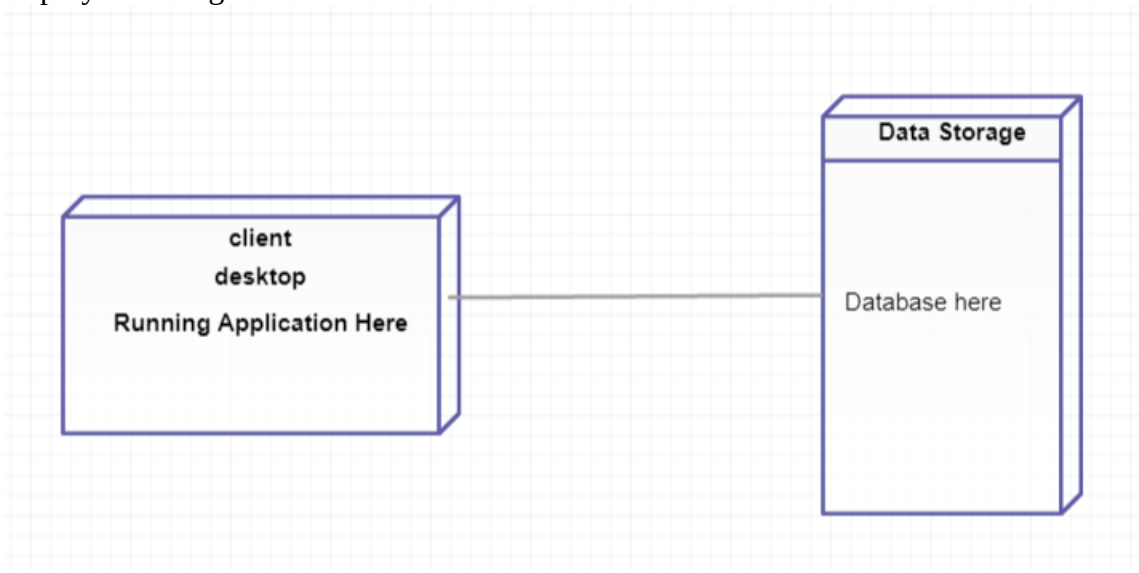
Layered Architecture Pattern of the solution



Package diagram

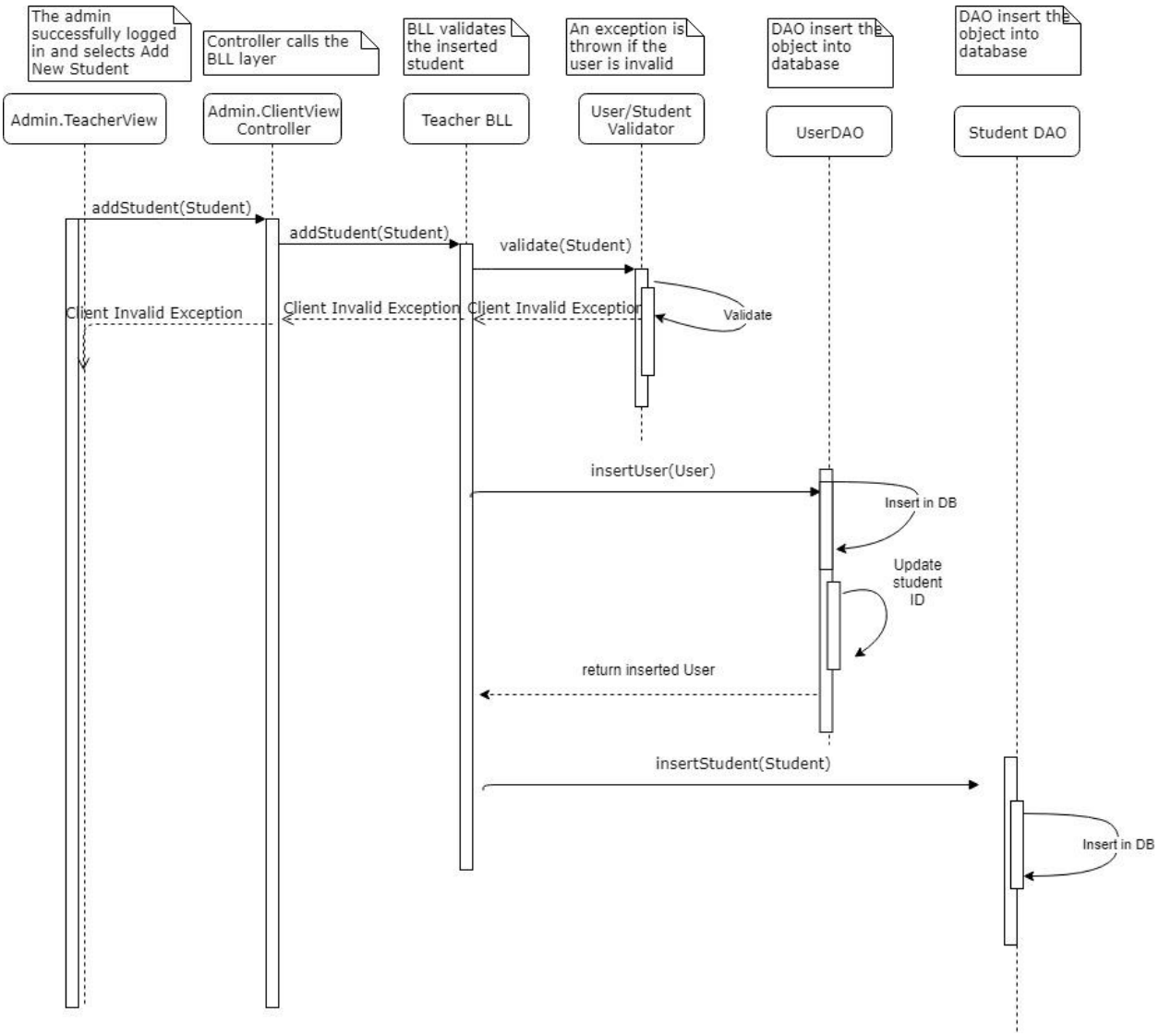


Deployment diagrams



4. UML Sequence Diagrams

Sequence diagram for inserting a new student.



5. Class Design

5.1 Design Patterns Description

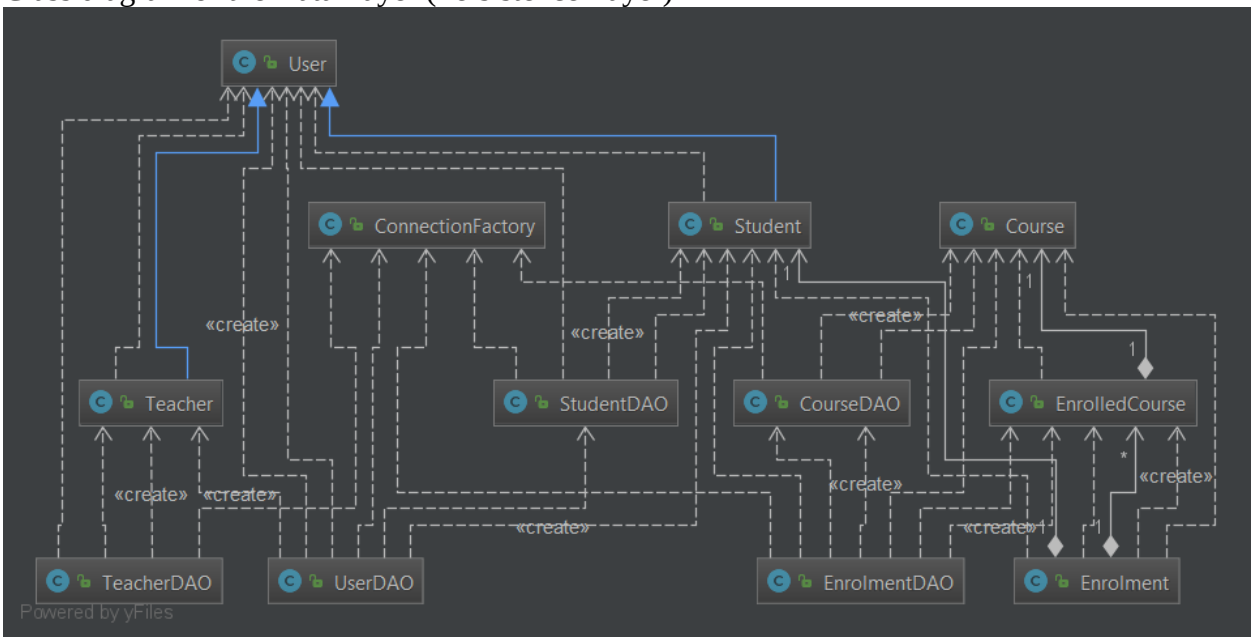
Singleton Creational Design Pattern
This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.
This pattern was used for the design of the Connection Factory class which is responsible for connection to the database.

Data Mapper Design Pattern
A Data Mapper is a Data Access Layer that performs bidirectional transfer of data between a persistent data store (often a relational database) and an in-memory data representation (the domain layer). The goal of the pattern is to keep the in-memory representation and the persistent data store independent of each other and the data mapper itself. The layer is composed of one or more mappers (or Data Access Objects), performing the data transfer.
This pattern was used for the design of the Data Access Objects.

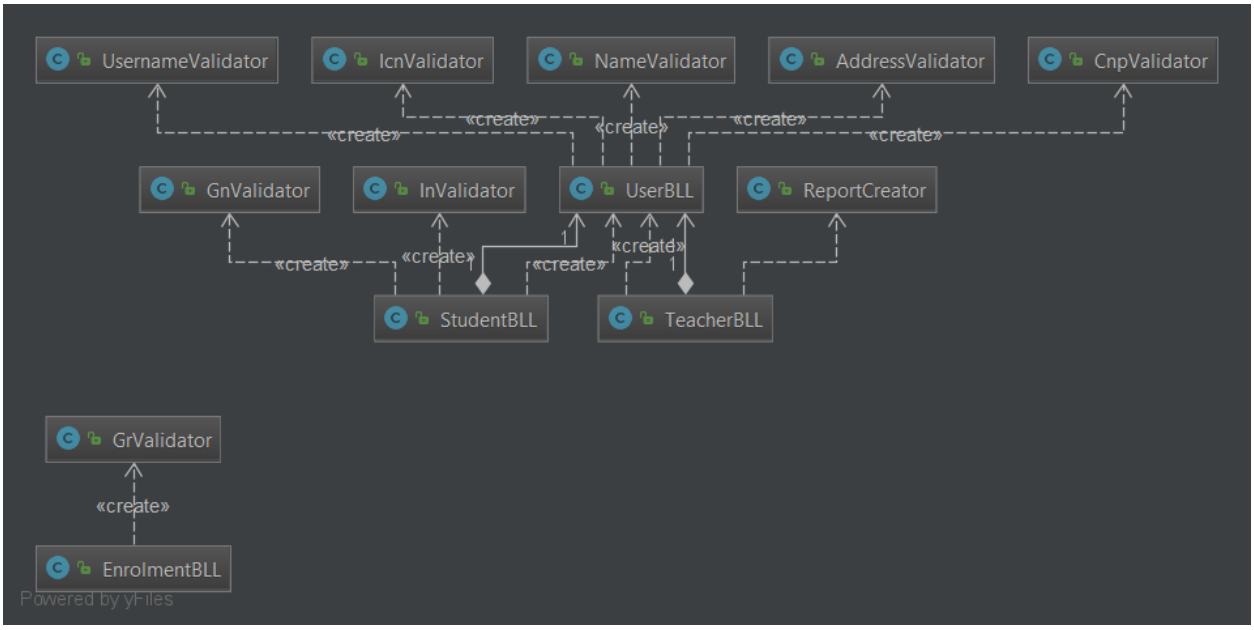
5.2 UML Class Diagram

Sinces the UML Diagram for the whole system would be of great proportions, I decided to present the class diagrams of the modules (layers), to have a better and more specific understanding.

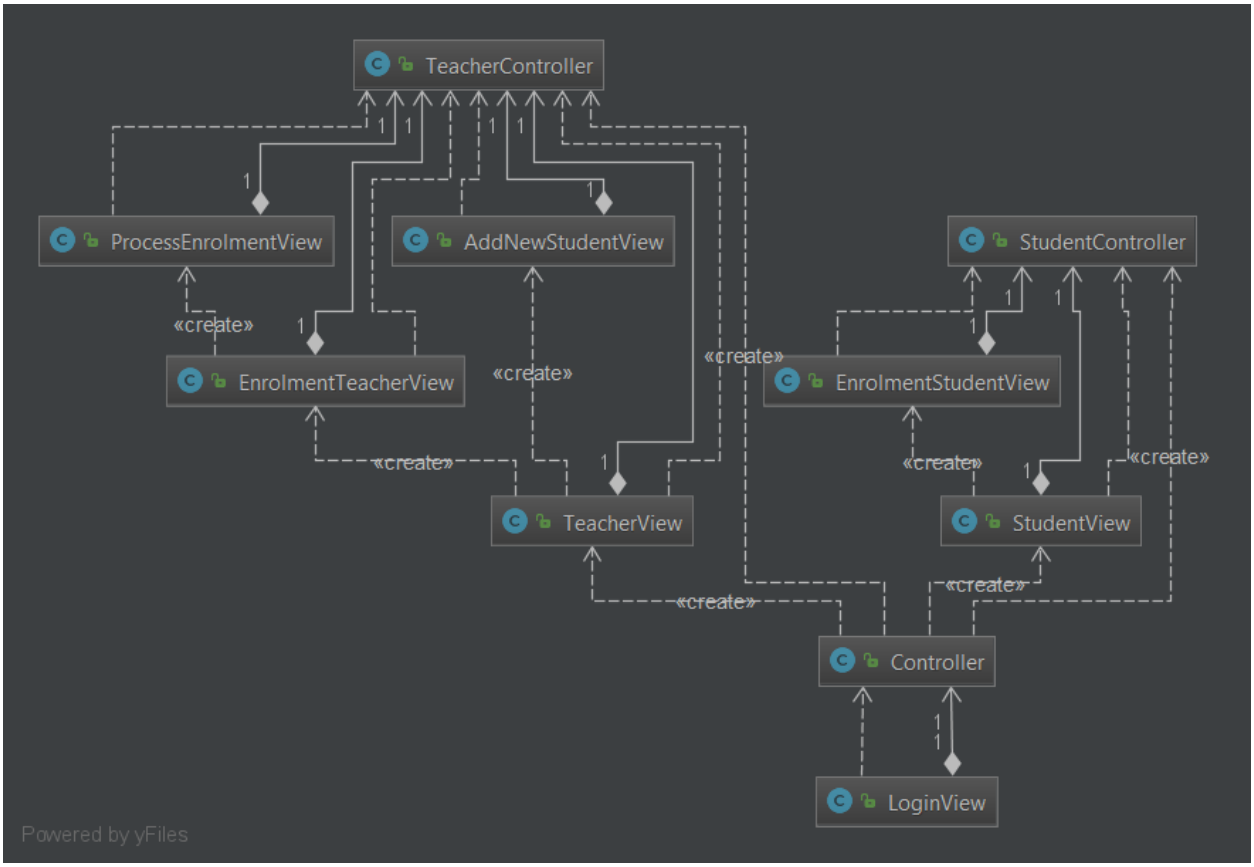
Class diagram of the Data Layer (Persistence Layer)



Class diagram of the Business Logic Layer.

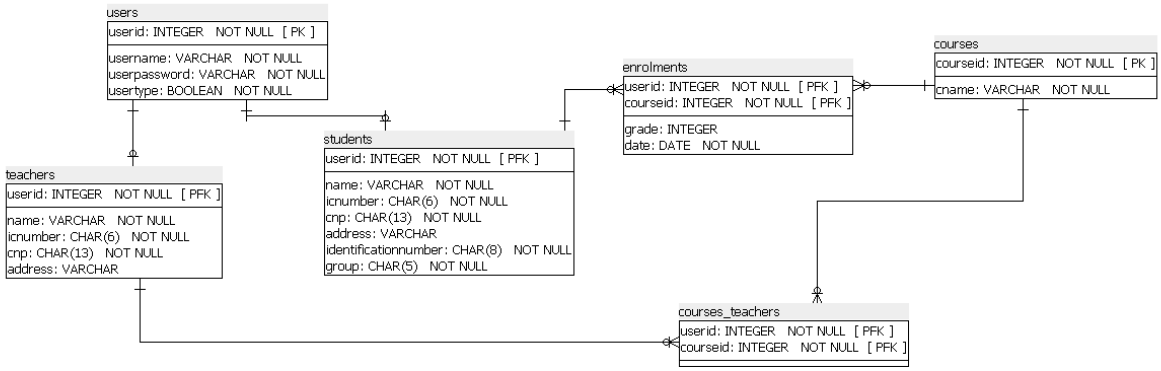


Class diagram of the Presentation Layer



6. Data Model IR diagram

Below I present the Data IR diagram. This was used to represent and model the data of the application. There are six tables: four of them for direct data storage (client and other information) and the other two for relationships (the model a one-to-many ⇔ many-to-one relationship).



7. System Testing

I will use unit testing in order to test my application.
I tried to make an implementation which will be favorable and sustain testing.
Mocking is primarily used in unit testing. An object under test may have dependencies on other (complex) objects. To isolate the behavior of the object one wants to test, it replaces the other objects by mocks that simulate the behavior of the real objects. This is useful if the real objects are impractical to incorporate into the unit test.
In short, mocking is creating objects that simulate the behavior of real objects.
Example of mocking test if a student was correctly inserted in the database.

```
public static void testStudentInsert(){
    Student student = new Student();
    student.setUsername("test");
    student.setPassword("123");
    student.setName("Test students");
    student.setUsertype(1);
    student.setGroup("30432");
    student.setIdentificationnumber("12345678");
    student.setAddress("beius");
    student.setCnp("1234567891111");
    student.setIcnumber("123456");

    StudentBLL test = new StudentBLL();
    test.insertStudent(student);
    try{
        Student verifystudent = test.getSpecificStudent(student);
        assert verifystudent.getUserid()==student.getUserid();
    }catch (IllegalArgumentException e){
        System.out.println(e.getMessage());
    }
```

8. Bibliography

Here are mentioned some resources which were helpful for writing this documentation and as well gave me a better understanding of the concepts and guided me in the implementation process:

Information and knowledge:
Microsoft Application Architecture Guide, 2nd Edition
Software Architecture Patterns, by Mark Richards
https://www.tutorialspoint.com
https://en.wikipedia.org

Tools helpful for the documentation:
https://www.draw.io/
https://www.smartdraw.com
https://creately.com