Students Management

Analysis and Design Document

Student: Costea Ana-Maria

**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

The application which I am about to implement is an application designed for the management of students from department of Computer Science at Technical University of Cluj-Napoca. This application must have two types of users, one for student user and the other one for teacher / admin user. The user must enroll himself with an username and a unique password in order to use the application.

# Functional Requirements

The regular user, the student, has access to the following operations :

* Add, update, view all users information ( name, number of identification card, address, etc)
* Create, update, delete, view student profile ( account information : number of identification, group, enrolments, grades)
* Enrolment to classes ( enrolment, exams, grades)

The admin user has access to the following operations :

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

As functional requirements I must implement the described operations, using Spring Boot, in order to create a web application, having dependency management software, working with Hibernate and with a new database, PostgreSQL.

# Non-functional Requirements

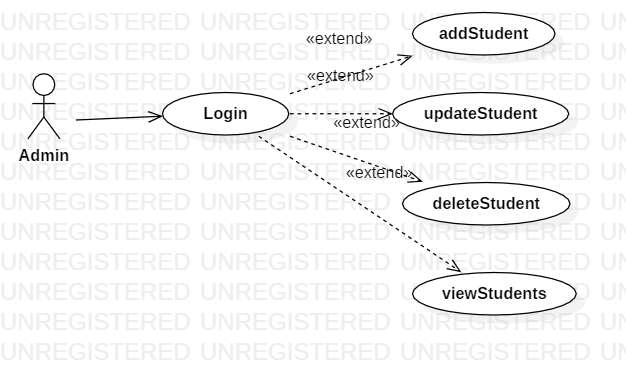
As non-functional requirements of the application which I am about to implement, I can mention the program’s performance ( for example, response time, utility), availability, security, data integrity. All data which I am about to work with will be introduced by using a database. The difference between functional requirements and the non-functional ones is that the first category describes what the system is supposed to do, while the second category describes how the system works.

2. Use-Case Model

Use Case for Student



Use Case for Admin



Use-Case description :

* **Use case** : selecting a desired operation by the user
* **Level** : use-goal level
* **Primary actor** : student or teacher / admin user
* **Main success scenario** : The user enrolls, choose the type of user, introduces the username and the password of account, then he select the operation he wants. The system analyses the requirement of the user and displays the output.
* **Extensions**  The user choose the type of user and then he introduces data with which he connects to the application. A success scenario could be : the user introduces correctly data, and so he has access to the operations which are designed for his type of user. A fail scenario could be : the user doesn’t introduce data correctly and so he doesn’t have access to the operations designed for him, an error message being displayed ( for example, “Username or password not valid ! Please enroll again!”).

3. System Architectural Design

**3.1 Architectural Pattern Description**

Implementing the application, I am going to use the following design pattern :

* **Layered pattern**

I choose to use this design pattern to structure the programs which can be composed into groups of subtasks, each of them being at a particular abstraction level. Each layer provides services to the next bigger layer. Most common used layers are :

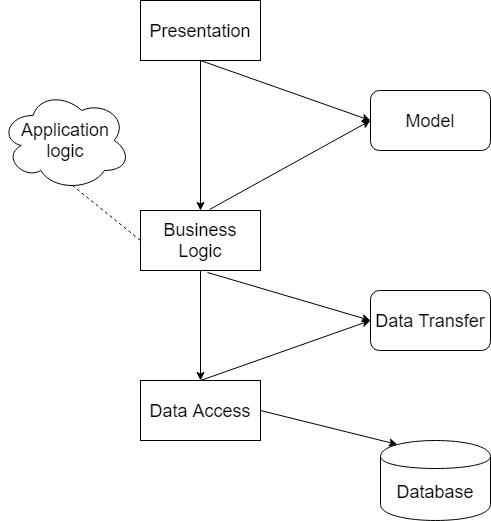
- Presentation layer ( UI layer)

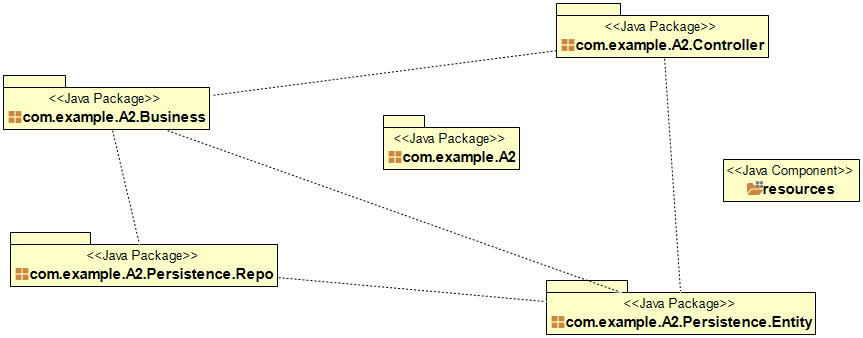
- Application layer ( service layer)

- Business logic layer ( domain layer)

- Data access layer ( persistence layer)

**3.2 Diagrams**

****



4. UML Sequence Diagrams

I will design the sequence diagram for the scenario of adding a new student account by a student user.



5. Class Design

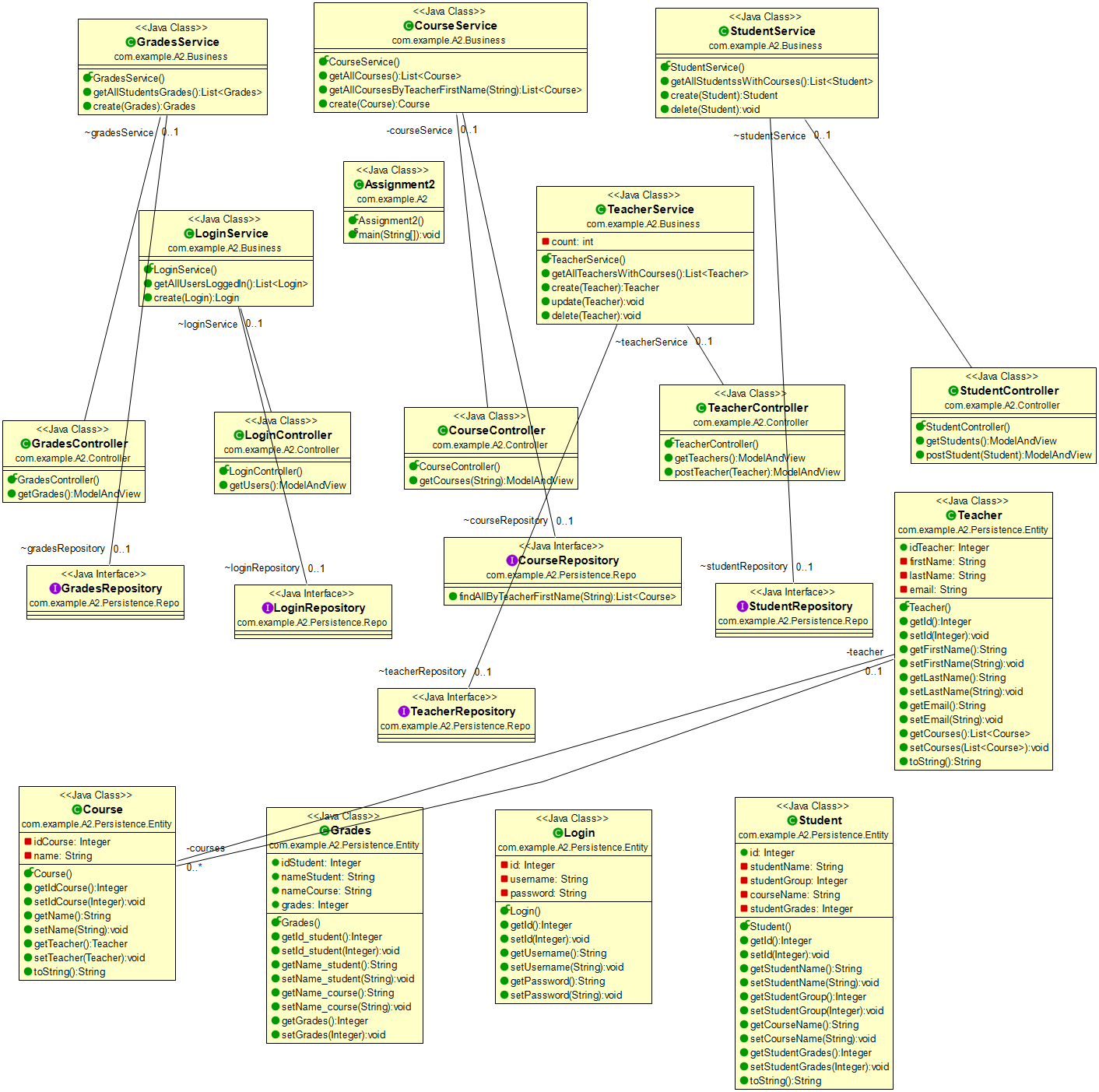
**­**

**5.1 Design Patterns Description**

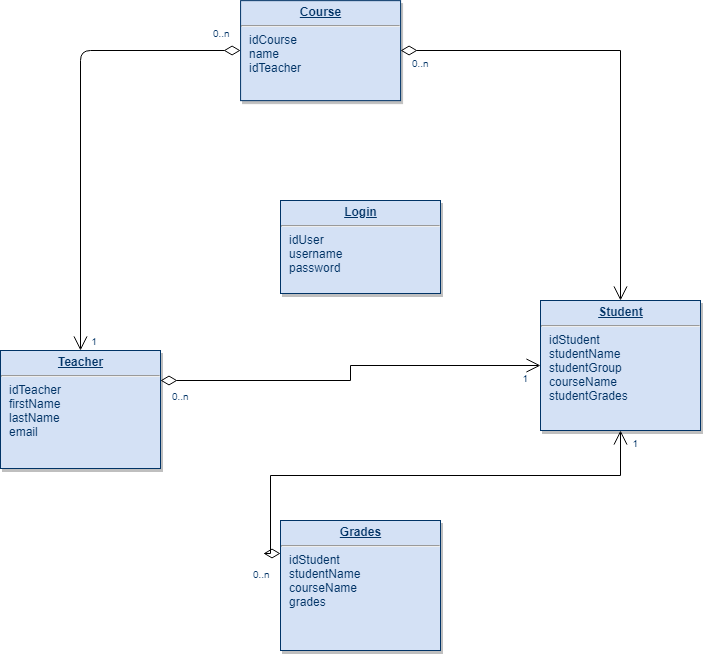
For this application, I used the following design patterns :

* **MVC** ( Model-View-Controller), for connecting the different parts of this application into model, view and controller. The model represents the Java objects, the view represents the way in which data can be viewed and the controller controls the data flow and modifies the view when parts of the model change.

**5.2 UML Class Diagram**



6. Data Model



7. System Testing

For this assignment, I made a Maven project, which includes classes for tests. For this part, I took tests for different cases of use, such as creating a new student, adding a new course for a teacher, updating some information for a student.

8. Bibliography

[1] <https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>

[2] <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>

[3] <https://www.journaldev.com/1827/java-design-patterns-example-tutorial>

[4] <https://www.baeldung.com/the-persistence-layer-with-spring-and-jpa>

[5] <http://www.springboottutorial.com/hibernate-jpa-tutorial-with-spring-boot-starter-jpa>

[6] <https://dzone.com/articles/spring-boot-and-postgresql>