ProTeach
Architecture Notebook

1. Purpose

This document describes the decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

2. Architectural goals and constraints

2.1. Functional requirements

Most important use-cases to be implemented in the first release.

2.1.1. Assign teacher to course

This functionality enables administration personnel to modify the association between teachers and courses. This includes altering the list of courses, if necessary, by adding new one.

2.1.2. Course enrolment request

This functionality enable student make request to enroll him/her self in the course in order to attend and perform tests. This include browse for available courses and see their basic informations.

2.1.3. Manage test

This functionality enables teacher to create and evaluate a test and enable students to perform a test. This include adding a new test with related questions for a certain course and store test's result for each student in the database.

2.2. Non-functional requirements

2.2.1 System security – limited access (priority: critical, category: security)

Users can access only the system functions they've been given clearance for (for example, the student can't have access to the module for teacher or administrator)

2.2.2 Data availability (priority: high, category: availability)

All services can be made available after the system crash within 1 day. All data will be backed up daily.

2.2.3 Persistency of data (priority: critical, category: usability)

The system has mechanisms for storing data permanently.

2.2.4 Input data validation (priority: high, category: security - usability)

Ensure that user input is correct, and useful.

2.2.5 Usability (priority: medium, category: usability)

The software should be easy to operate and requires a low level of support for the system users.

3. Decisions and justifications

Goal	How achieved (Tactics)
System security – limited access	Tactic: System use authentication and authorization mechanisms. Justification: System need to authenticate users in order to provide him the correct functionalities, and because it manages user's personal informations.
Data availability	Tactic: System backup data stored in DB . Justification: System must provide data availability if errors on DB occurs.
Persistency of data	Tactic: System use MySQL Server as relational DB. Justification: System need to store data permanently in rayalable way.
Input data validation	Tactic: Data is validated both on client and server side. Justification: Validation on client side allow better user experience. Validation on server allow correct control of input data.
System availability	Tactic: System checks if user web browser is compliant with the requirements needed to properly use the system. Justification: If user's web browser is not compatible an error message will be shown and the user will not be able to use the system, to avoid not correct behaviours.

4. Architectural Mechanisms

Mechanism to enable described tactics.

Authentication

The user have to login with the username and password.

Authorization

The User access the functionalities of the system will be based in their roles and permissions.

System backup

The data in the database are backed-up periodically by a script.

Persistency

MySQL Server is used as DBMS to store data.

Input data validation

Use JavaBeans Validation that is supported by constraints in the form of annotations placed on a field, method, or class to provide user input validation.

System availability

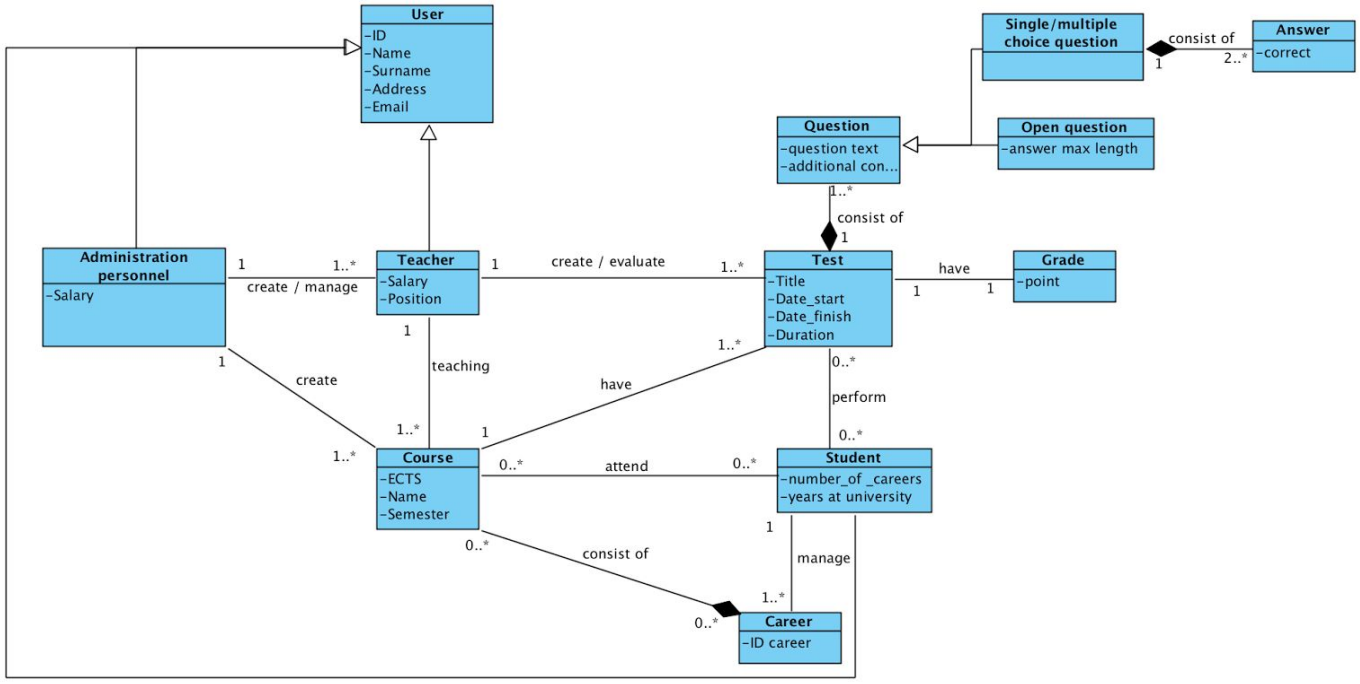
System must be available from the most greater number of browser in order to enable the largest number of users to use it.

5. Key abstractions

List and briefly describe the key abstractions of the system.

User	Represent the subject of the system, they interacting with the system. They can be created, modified, deleted.
Administrator , Teacher, Student	Represent the different roles of the user.
Course	Represent the relation with student-teacher-exam, can be created, modified, deleted.
Exam	Represent the set of the question, it can be create, modified, deleted, assigned to the course, solved
Question	Represent the a basic element of the exam, It can

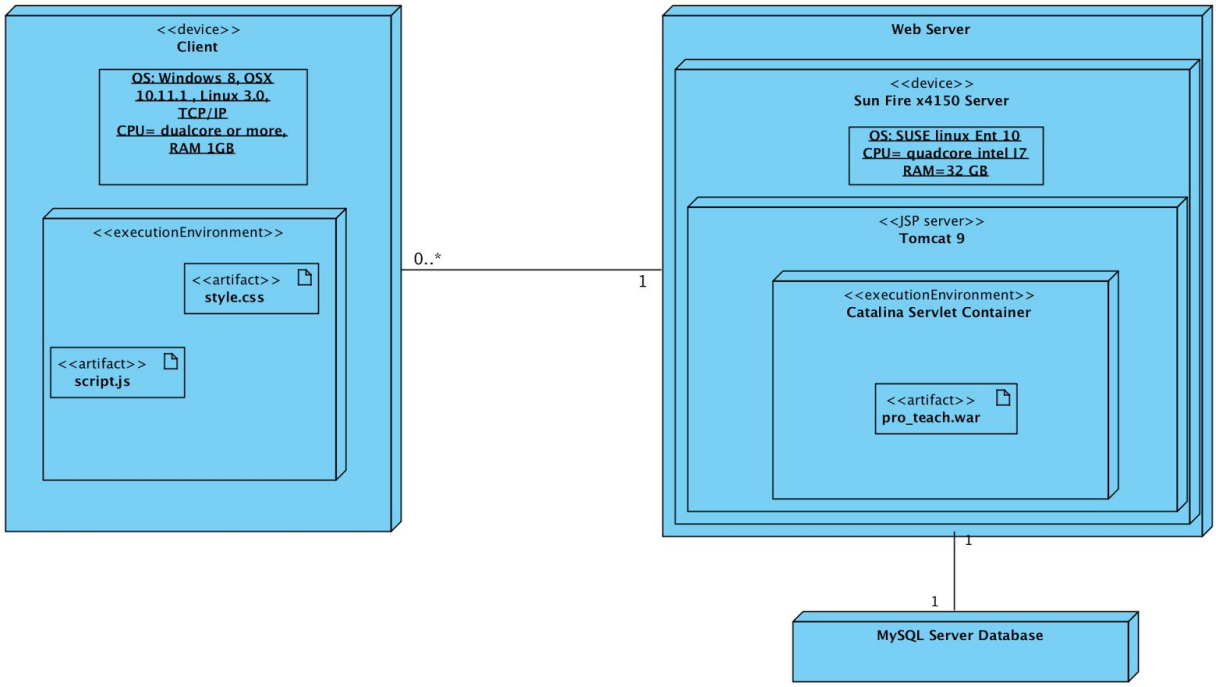
	be created, modified and deleted
Report	Represent the result of test or tests (statistics) . It can be generated and saved.



The administrator can create course and teacher, he/she can join teacher and course. The teacher can create and evaluate the test. The course can be attend only from student. Each student can manage his/her careers. Each Test has a set of questions, they can be single/multiple or open question. A course have one or more test.

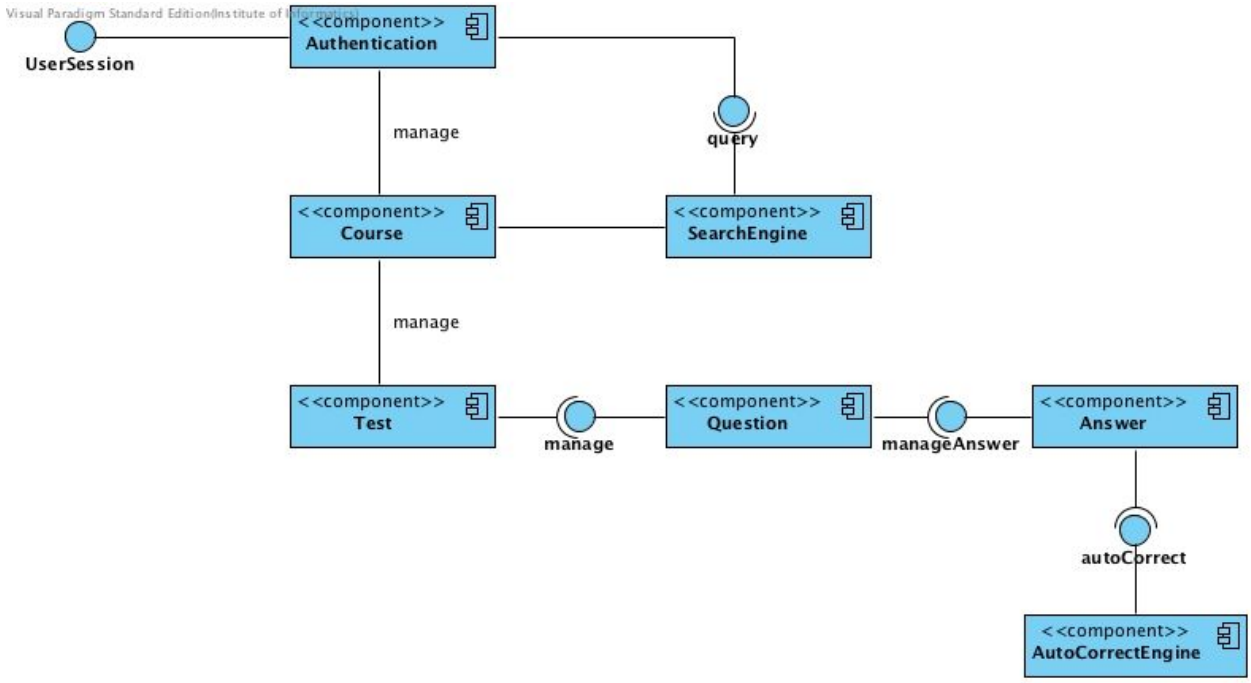
6. Architectural views

6.1. Deployment viewpoint



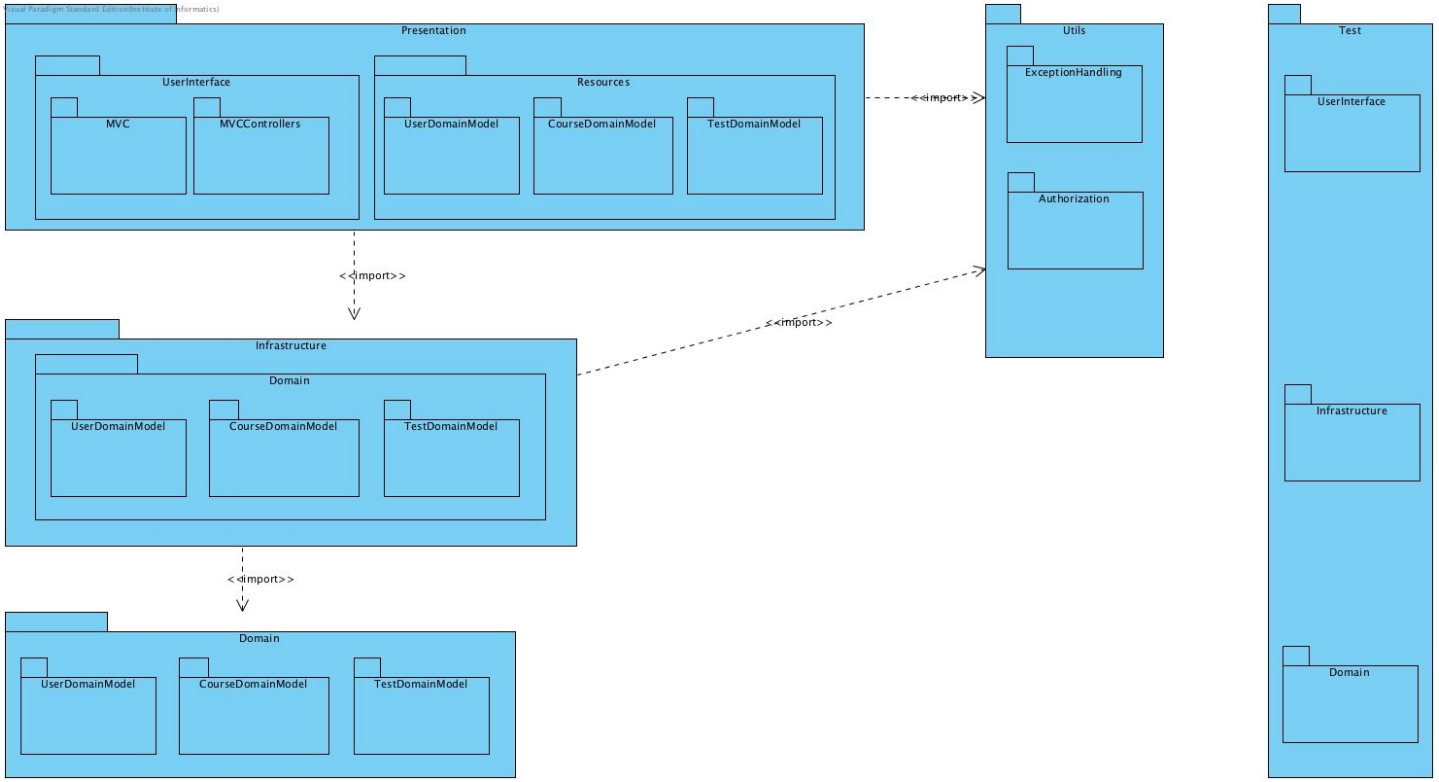
Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. We can see the minimum hardware component of the client and the artifact of the client. More client can connect to one Web Server with installed a linux OS with Tomcat 9. Tomcat has a servlet that manage the request of the client and it can do the query to the DB.

6.2. Functional viewpoint



Component diagram above presents the main components that will the system contain. To start user must pass *Authetntacion* to manage any work in the system. User can manage courses and search for courses make a query to *SearchEngine* using defined interface. Component *Course* is connected to Test component. To manage *Question* certain rules must be followed using interface. Same goes for the *Answer* component. *AutoCorrectEngine* component will enable automatic check if provided answer is correct.

6.3. Development viewpoint



For development viewpoint Package Diagram is shown above. There are three layers in package structure: Presentation, Infrastructure and Domain. In Presentation package *UserInterface* package will contain everything related to provide UI and Model–view–controller (MVC) software architectural pattern will be used. *Domain* package will contain instances that need to be persisted in a database like User, Course and Test. Package *Infrastructure* will provide everything need to manipulate with data defined in *Domain* package. *Utils* package provided method need in multiple layers like Exception handling. *Test* package will contain all test data that will used for testing the system.

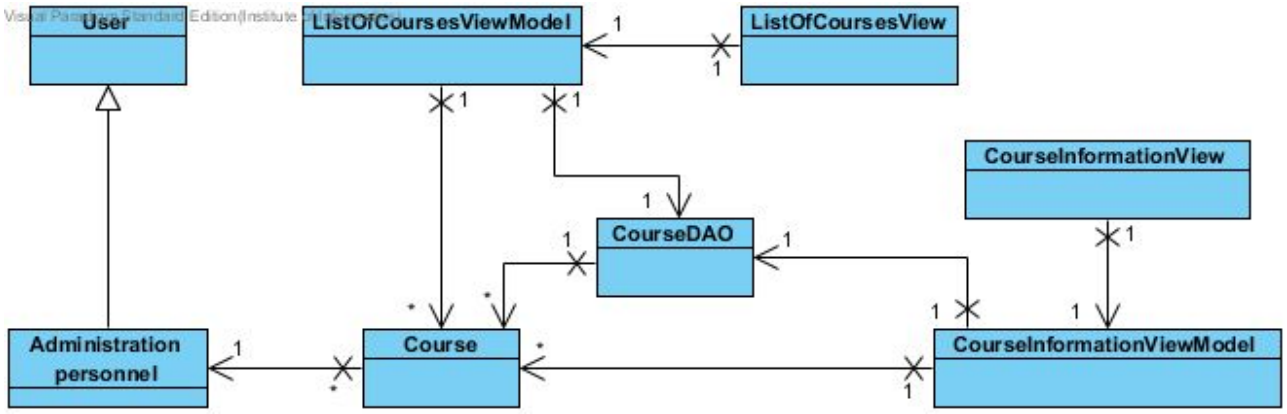
7. Use-case realizations (for selected use-cases)

7.1 Assign teacher to course

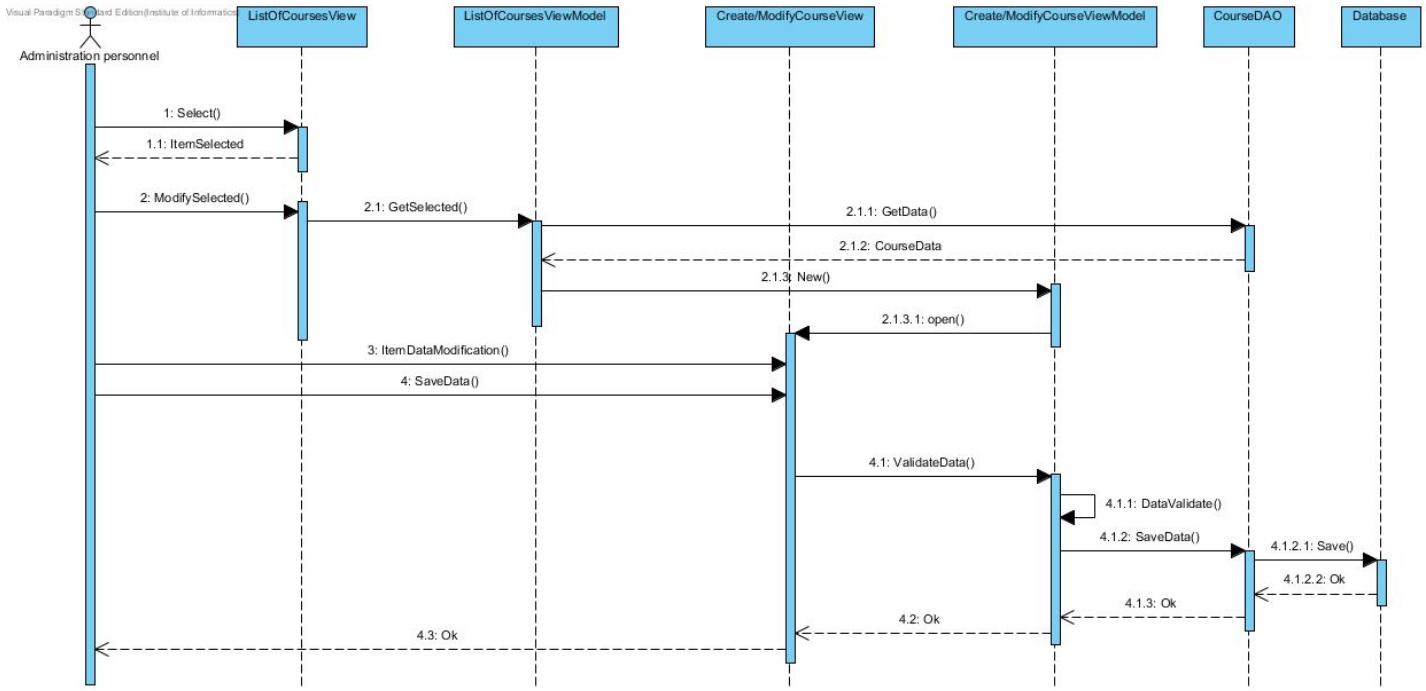
7.1.1 Class diagram

Here are shown the components involved in this use case, the user involved is *Administration personnel* that must interact with *Course*. The component responsible for communicating with BD for *Course* is *CourseDAO*. For the course are shown the view-model and view related to:

- Presentation of the course information;
- Listing the courses.



7.1.2 Sequence diagram

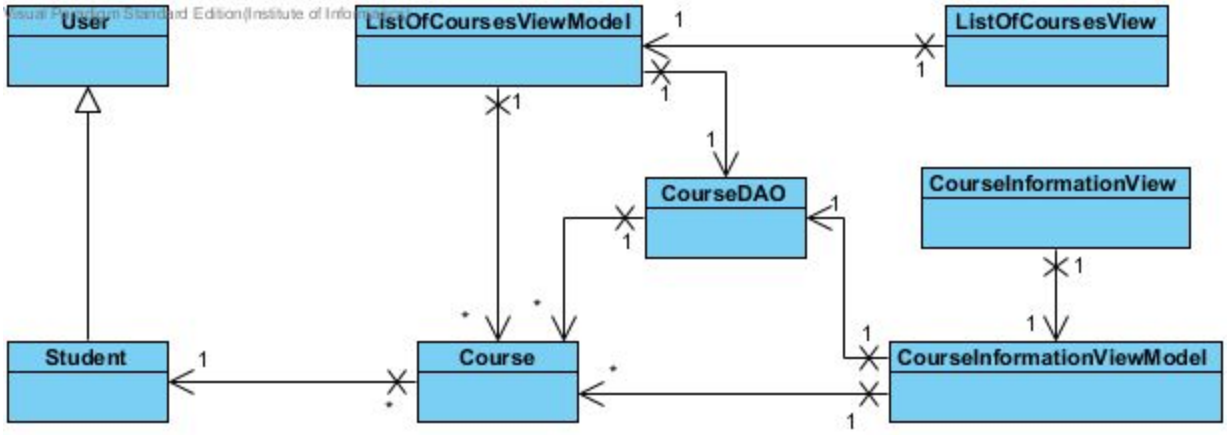


7.2 Course enrolment request

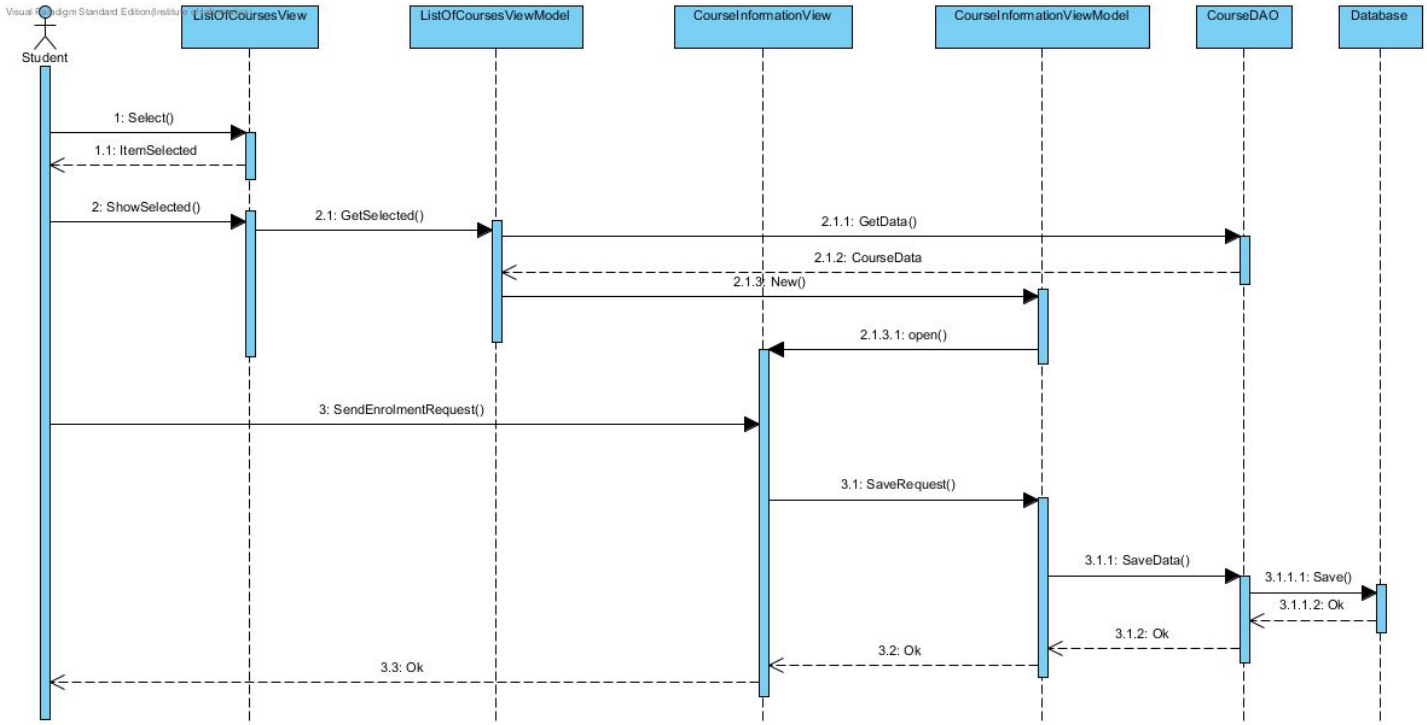
7.2.1 Class diagram

In the below schema we can see the components involved in this use case, here *Student* must interact with *Course*. The component responsible for communicating with BD for *Course* is *CourseDAO*. For the course are shown the view-model and view related to:

- Presentation of the course information;
- Listing the courses.



7.2.2 Sequence diagram



7.3 Manage test

7.3.1 Class diagram

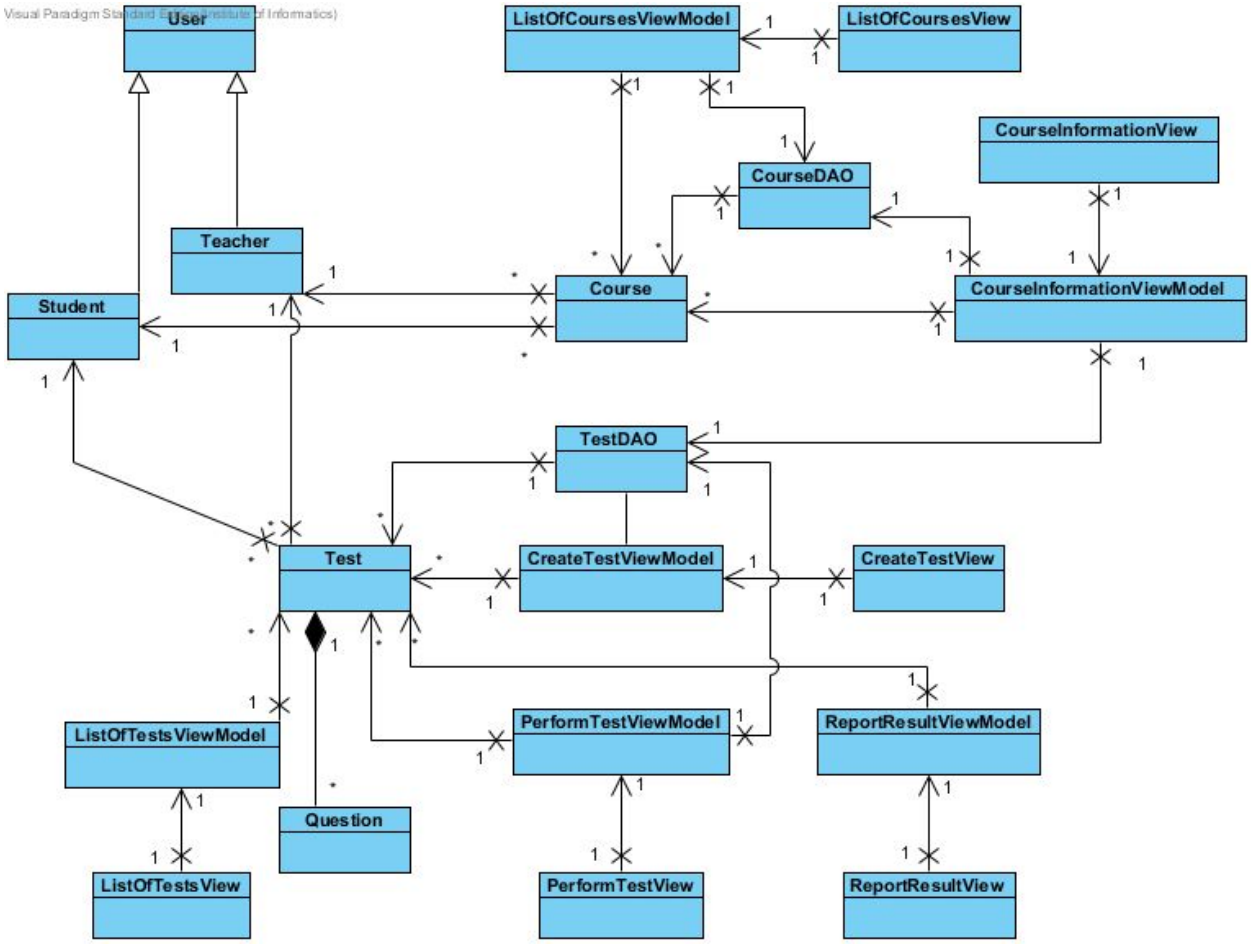
In the below schema we can see the components involved in this use case. We have two type of user, *Teacher* and *Student* that must interact with *Course* and *Test*. For the *Course* are shown the view-model and view related to:

- Presentation of the course information;
- Listing the courses.

For the *Test* are shown the view-model and view related to:

- Creation of a test, only *Teacher* will use this component;
- Perform of a test, only *Student* will use this component;
- Report result of a test;
- List the tests of one course.

In the schema are also shown the components responsible for communicating with BD for *Test* and *Course*, respectively *TestDAO* and *CourseDAO*.


User
ListOfCoursesViewModel
ListOfCoursesView
CourseInformationView
CourseDAO
Teacher
Student
Course
CourseInformationViewModel
TestDAO
Test
CreateTestViewModel
CreateTestView
ListOfTestsViewModel
PerformTestViewModel
ReportResultViewModel
Question
ListOfTestsView
PerformTestView
ReportResultView

7.3.2 Sequence diagram