CoursesManagementApp

Sprint Report

MYY803
Athanasopoulos Alexandros - 4020
Sirpas Vasileios - 4174
Boulotis Panagiotis - 4271

VERSIONS HISTORY

Date	Version	Description	Authors
18/5/2022	1.0	Implemented the US, there is still room for more user-friendly interface	Athanasopoulos Alexandros Sirpas Vasileios Boulotis Panagiotis

1 Introduction

1.1 Purpose

This is an application designed for university instructors, so they can manage their courses information, add, delete, and edit the students enrolled in them, as well as configure their grades. Using the grade information, an overall statistics page can be displayed for each course.

1.2 Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies the Sprint's backlog. Section 3 specifies the main design concepts for this release of the project. Section 4 describes the design of the project using UML diagrams.

2 Scrum team and Sprint Backlog

2.1 Scrum team

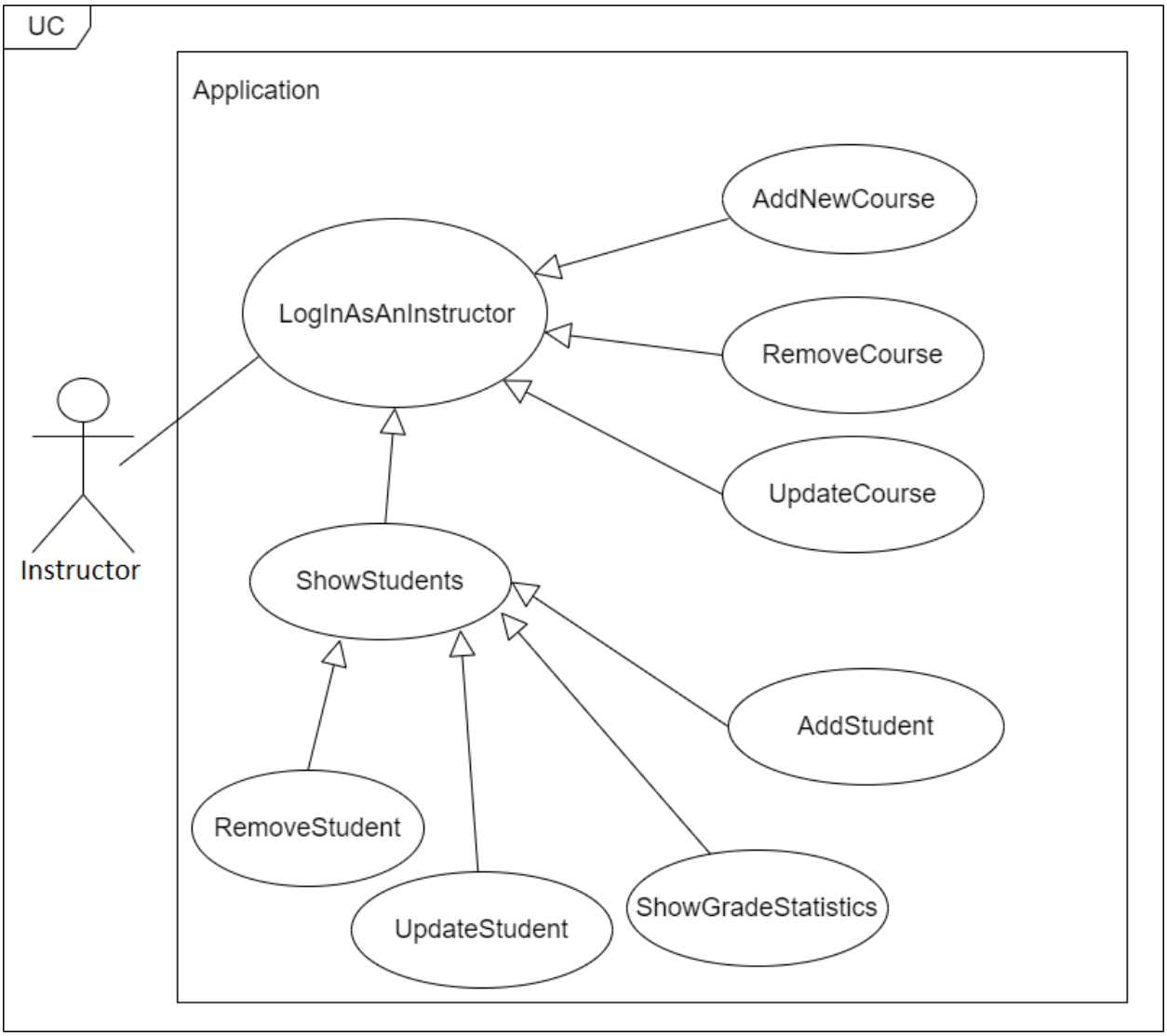
Product Owner	University of Ioannina
Scrum Master	Athanasopoulos Alexandros
Development Team	Athanasopoulos Alexandros Sirpas Vasileios Boulotis Panagiotis

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**11/4/22**	**15/4/22**	**1**	**US1**

2	**30/4/22**	**12/5/22**	**3**	**US2-10**
3	**13/5/22**	**16/5/22**	**1**	**US11&12**

3 Use Cases



3.1 LogInAsAnInstructor

Use case ID	UC1
Actors	Instructor
Preconditions	The instructor must be registered to the system, so that they have an username and a password.
Main flow of events	1. The use case starts when the instructor inputs their credentials (username and password). 2. The instructor presses the “Log In” button
Alternative flow	If the username or/and password are not valid, the system displays an error page.
Post conditions	The instructor has successfully logged in to the system and a page with all the courses is displayed.

3.2 AddNewCourse

Use case ID	UC2
Actors	Instructor
Preconditions	The instructor must be logged in and the course list must be displayed.
Main flow of events	1. The use case starts when the instructor presses the “Add Course” button. 2. The system displays a new page where the instructor can enter the course’s information. 3. The instructor inputs the relative information in each field provided. 4. The instructor presses the “Submit” button. 5. The system adds the new course to the course list.
Alternative flow	If the instructor presses the “Back to Course List” link, they return to the course link page, without adding a new course.
Post conditions	A new course is now added, and it is displayed on the course list page.

3.3 RemoveCourse

Use case ID	UC3
Actors	Instructor

Preconditions	The instructor must be logged in and the course list must be displayed.
Main flow of events	1. The use case starts when the instructor presses the “Delete” button. 2. The system displays a warning message. 3. The instructor presses the “OK” button. 4. The system deletes the course selected from the course list.
Alternative flow	If the instructor presses the “Cancel” button, when the warning is displayed, they return to the course link page, without deleting the selected course.
Post conditions	The selected course is now deleted from the course list.

3.4 UpdateCourse

Use case ID	UC4
Actors	Instructor
Preconditions	The instructor must be logged in and the course list must be displayed.
Main flow of events	1. The use case starts when the instructor presses the “Update” button. 2. The system displays a new page where the instructor can edit the current course’s information. 3. The instructor inputs the relative information in each field provided. 4. The instructor presses the “Submit” button. 5. The system updates the course on the course list.
Alternative flow	If the instructor presses the “Back to Course List” link, they return to the course link page, without updating the course.
Post conditions	The course is now updated.

3.5 ShowStudents

Use case ID	UC5
Actors	Instructor
Preconditions	The instructor must be logged in and the course list must be displayed.
Main flow of events	1. The use case starts when the instructor presses the “Show Students” button.

	2. The system displays a new page with the students list for the selected course.
Alternative flow	-
Post conditions	-

3.6 AddStudent

Use case ID	UC6
Actors	Instructor
Preconditions	The instructor must be logged in and the student list must be displayed.
Main flow of events	1. The use case starts when the instructor presses the “Add Student” button. 2. The system displays a new page where the instructor can enter the student’s information. 3. The instructor inputs the relative information in each field provided. 4. The instructor presses the “Submit” button. 5. The system adds the new student to the student list.
Alternative flow	If the instructor presses the “Back to Student List” link, they return to the student link page, without adding a new student.
Post conditions	A new student is now added, and they are displayed on the student list page.

3.7 RemoveStudent

Use case ID	UC7
Actors	Instructor
Preconditions	The instructor must be logged in and the student list must be displayed.
Main flow of events	1. The use case starts when the instructor presses the “Delete” button. 2. The system displays a warning message. 3. The instructor presses the “OK” button. 4. The system deletes the student selected from the student list.

Alternative flow	If the instructor presses the “Cancel” button, they return to the student list page, without deleting the selected student.
Post conditions	The student is removed from the student list.

3.8 UpdateStudent

Use case ID	UC8
Actors	Instructor
Preconditions	The instructor must be logged in and the student list must be displayed.
Main flow of events	1. The use case starts when the instructor presses the “Update” button. 2. The system displays a new page where the instructor can edit the current student’s information. 3. The instructor inputs the relative information in each field provided. 4. The instructor presses the “Submit” button. 5. The system updates the student on the student list.
Alternative flow	If the instructor presses the “Back to Student List” link, they return to the student list page, without updating the student.
Post conditions	The student is now updated.

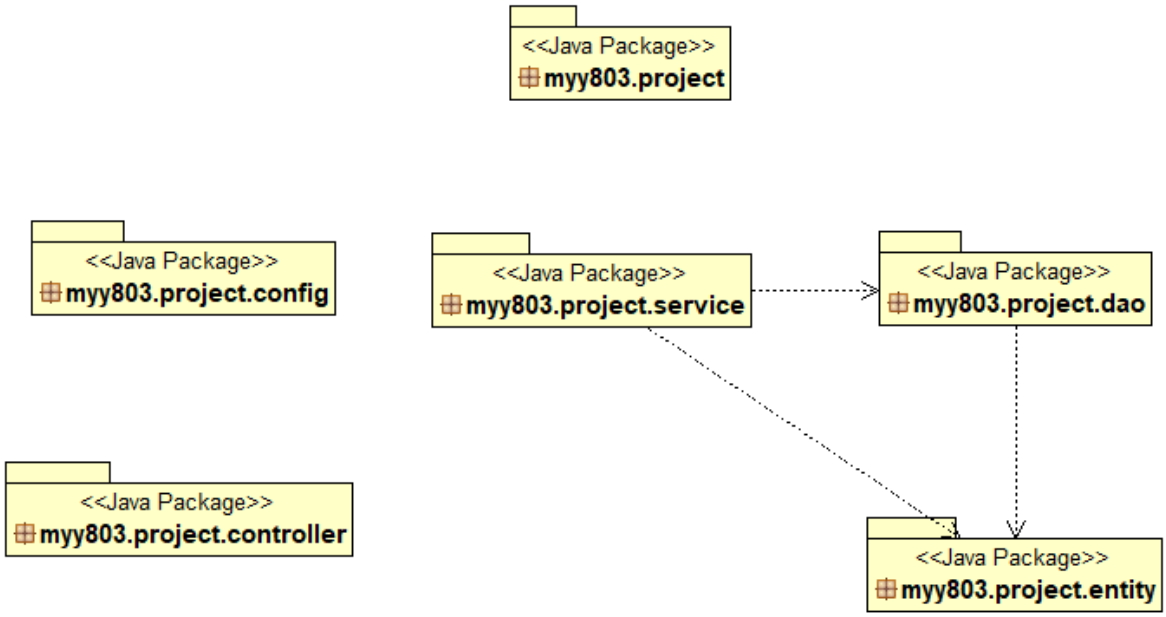
3.9 ShowGradeStatistics

Use case ID	UC9
Actors	Instructor
Preconditions	The instructor must be logged in and the student list must be displayed.
Main flow of events	1. The use case starts when the instructor presses the “View Statistics” button. 2. The system displays a new page with certain statistics.
Alternative flow	If the instructor presses the “Back to Student List” link, they return to the student list page.
Post conditions	-

4 Design

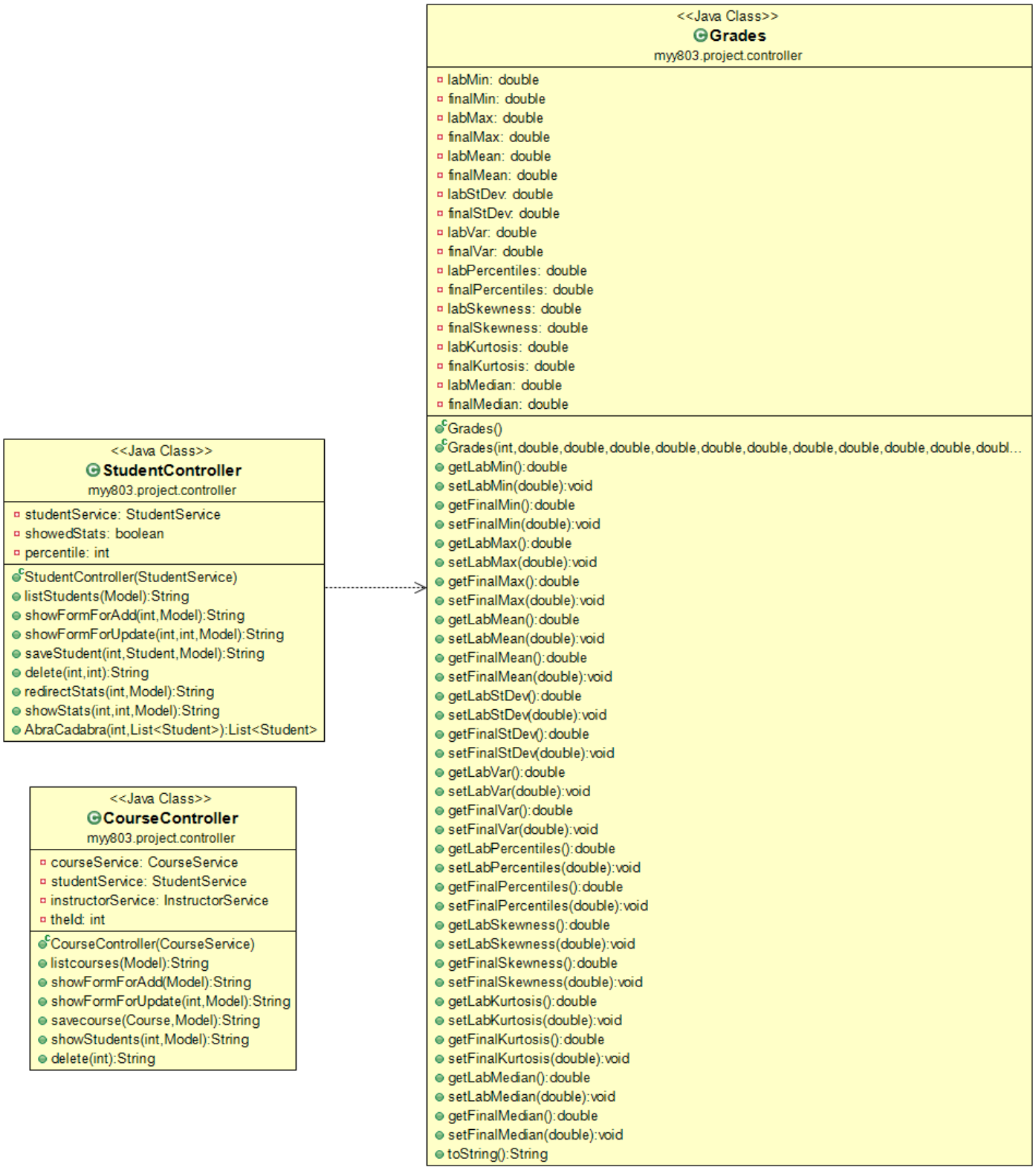
4.1 Architecture

Package diagram:

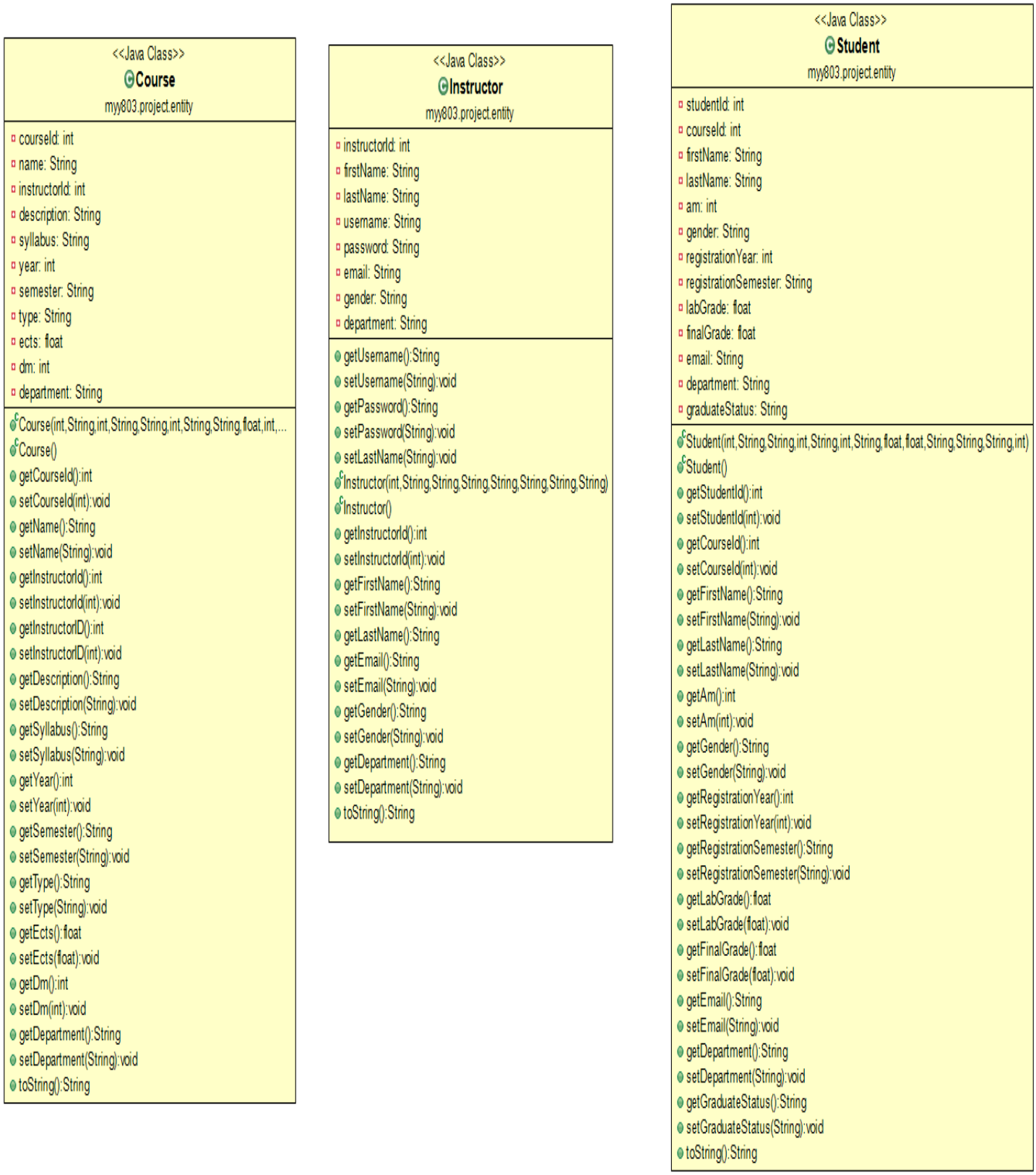


4.2 Design

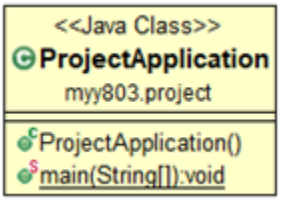
Package controller:



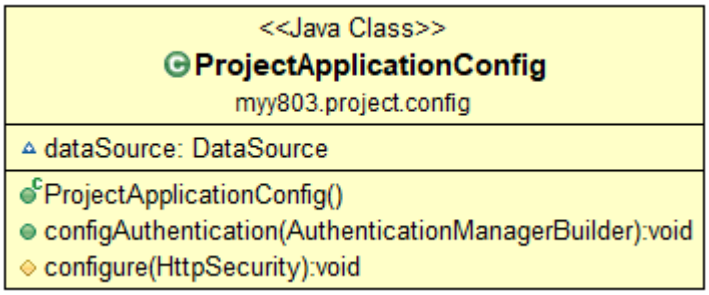
Package entity:



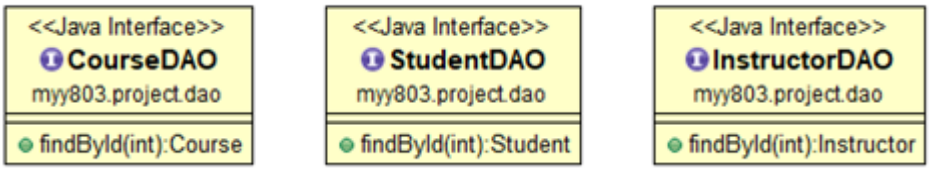
Package project:



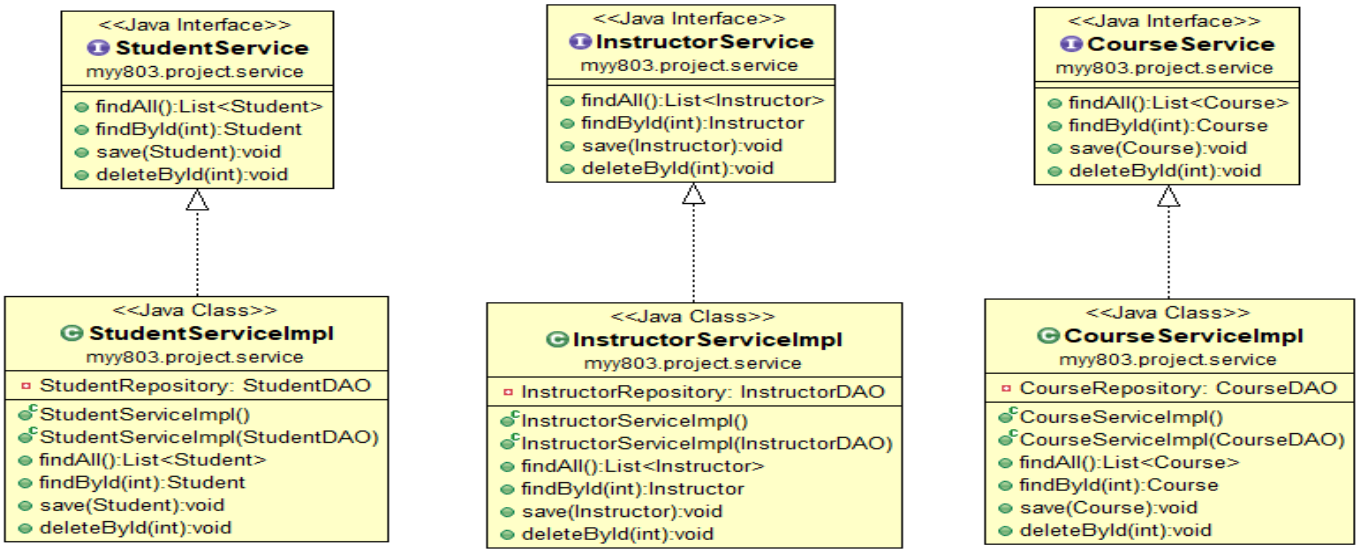
Package config:

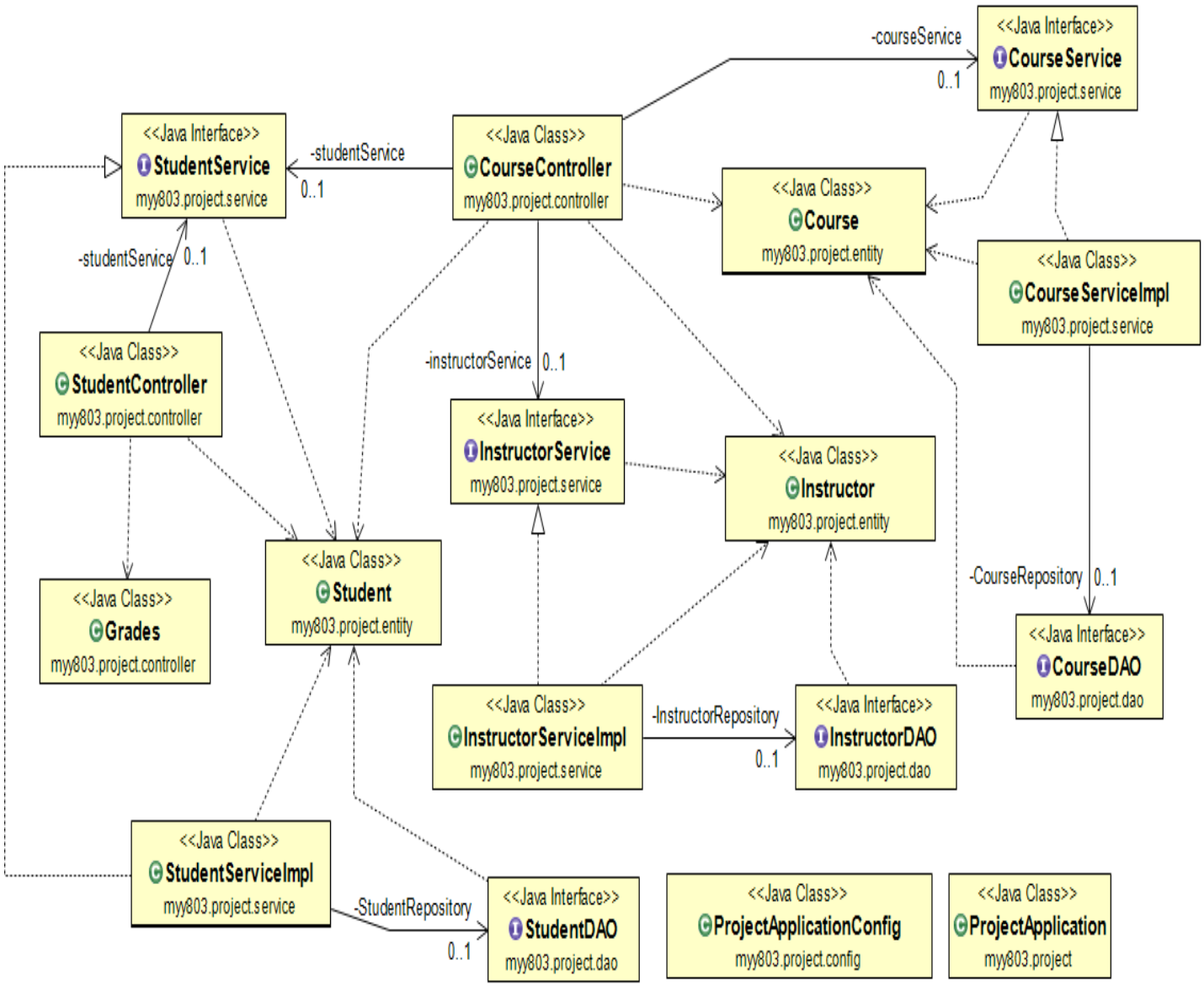


Package dao:



Package service:





Class Name: ProjectApplication	
Responsibilities: ▪ Executes the project	**Collaborations:** ▪ -

Class Name: ProjectApplicationConfig	
Responsibilities: ▪ Configures the login page and connects the database to the project	**Collaborations:** ▪ -

Class Name: Student	
Responsibilities: ▪ Constructs information for student entity	**Collaborations:** ▪ -

Class Name: Course	
Responsibilities: ▪ Constructs information for course entity	**Collaborations:** ▪ -

Class Name: Instructor	
Responsibilities: ▪ Constructs information for user entity	**Collaborations:** ▪ -

Class Name: Grades	
Responsibilities: ▪ Constructs information for grade statistics	**Collaborations:** ▪ -

Class Name: StudentDAO	
Responsibilities: ▪ Connects the StudentServiceImpl with the Student entity	**Collaborations:** ▪ Student

Class Name: CourseDAO	
Responsibilities: ▪ Connects the CourseServiceImpl with the Course entity	**Collaborations:** ▪ Course

Class Name: InstructorDAO	
Responsibilities: ▪ Connects the InstructorServiceImpl with the user entity	**Collaborations:** ▪ Instructor

Class Name: StudentService	
Responsibilities: ▪ Contains the methods of StudentServiceImpl	**Collaborations:** ▪ Student

Class Name: CourseService	
Responsibilities: ▪ Contains the methods of CourseServiceImpl	**Collaborations:** ▪ Course

Class Name: InstructorService	
Responsibilities: ▪ Contains the methods of InstructorServiceImpl	**Collaborations:** ▪ Instructor

Class Name: StudentServiceImpl	
Responsibilities: ▪ Connects the student repository with the database ▪ Finds all students from the repository ▪ Finds a student from the repository based on their id ▪ Saves a student to the repository ▪ Deletes a student from the repository	**Collaborations:** ▪ StudentDAO ▪ StudentService ▪ StudentService ▪ StudentService ▪ StudentService

Class Name: CourseServiceImpl	
Responsibilities: ▪ Connects the course repository with the database ▪ Finds all courses from the repository ▪ Finds a course from the repository based on their id ▪ Saves a course to the repository ▪ Deletes a course from the repository	**Collaborations:** ▪ CourseDAO ▪ CourseService ▪ CourseService ▪ CourseService ▪ CourseService

Class Name: InstructorServiceImpl	
Responsibilities: ▪ Connects the user repository with the database ▪ Finds all users from the repository ▪ Finds a user from the repository based on their id ▪ Saves a user to the repository ▪ Deletes a user from the repository	**Collaborations:** ▪ InstructorDAO ▪ InstructorService ▪ InstructorService ▪ InstructorService ▪ InstructorService

Class Name: StudentController	
Responsibilities: ▪ Adds/Updates a student via the html page ▪ Adds/Updates a student via the html page ▪ Finds students for specific course ▪ Displays the form page for adding a student ▪ Displays the form page for updating a student ▪ Displays the list of students	**Collaborations:** ▪ Student & StudentService ▪ Student & StudentService ▪ Student & StudentService ▪ Student & StudentService ▪ Student & StudentService ▪ Student & StudentService

▪ Displays the grades stats ▪ Updates the stats page if a new value is given to percentiles	▪ Student & StudentService & Grades ▪ Student & StudentService & Grades

Class Name: CourseController	
Responsibilities:	**Collaborations:**
▪ Adds/Updates a course via the html page	▪ Course & Instructor
▪ Deletes a course via the html page	▪ Course
▪ Displays the form page for adding a student	▪ Course
▪ Displays the form page for adding a student	▪ Course & CourseService
▪ Displays all students of the specific course	▪ Student & StudentService
▪ Displays all courses of the specific instructor	▪ Instructor & InstructorService & CourseService