CoursesManagementApp

Sprint Report

The Team

Καπενής Αλέξανδρος

VERSIONS HISTORY

Date	Version	Description	Author
5/18/2022	0.9845b	Course Manager App	The Team

1 Introduction

This document provides information concerning the **<X>** sprint of the project.

1.1 Purpose

1.2 Document Structure

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

2 Scrum team and Sprint Backlog

2.1 Scrum team

Product Owner	The Team
Scrum Master	The Team
Development Team	The Team

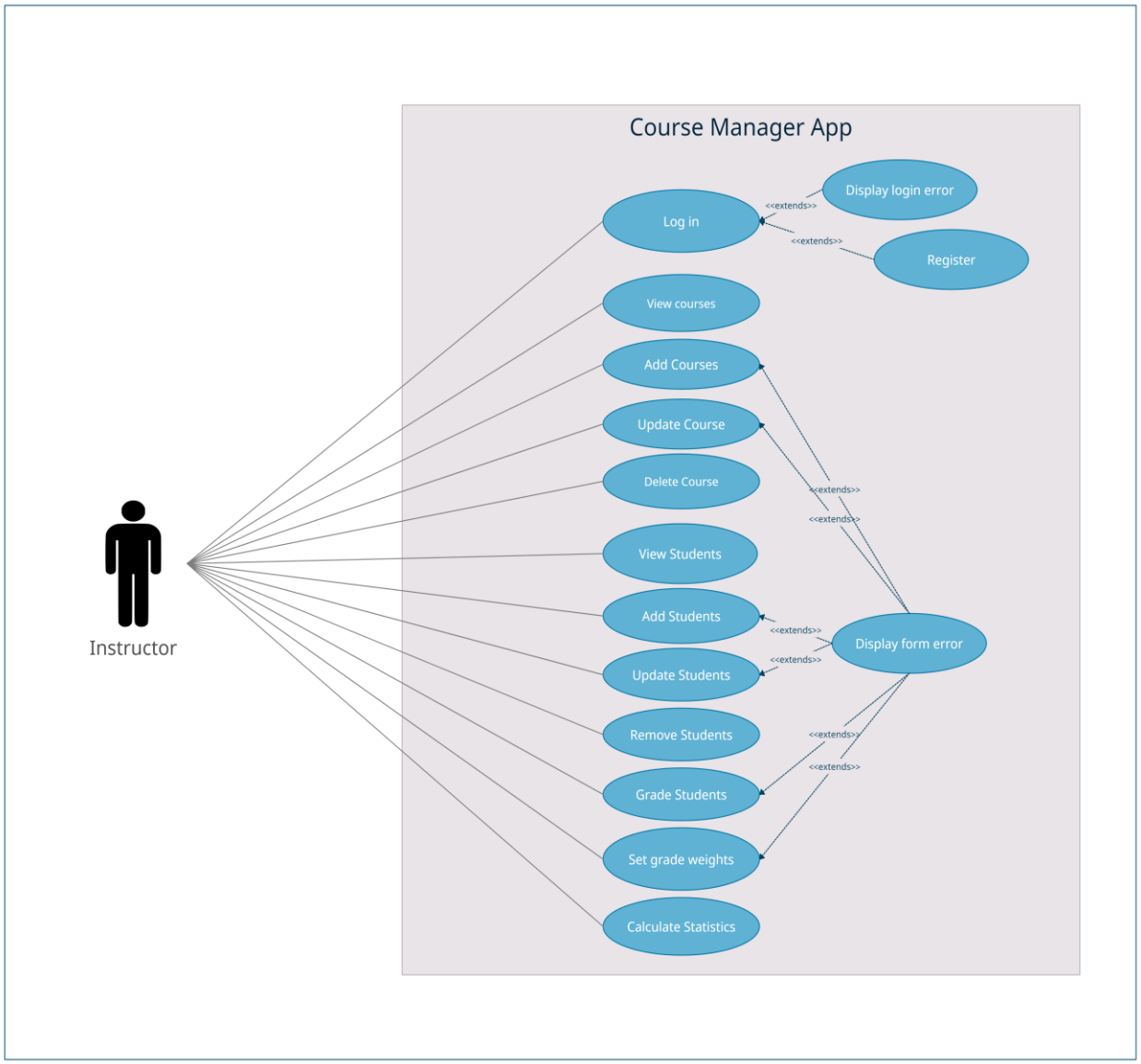
2.2 Sprints

<List below the sprints that you performed and the user stories that have been realized in each Sprint>

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**1/5/2022**	**3/5/2022**		**Project creation / dependencies installation / mySql connect to project**
2	**3/5/2022**	**5/5/2022**		**US2, US3, US4, US5 (Entity course)**
3	**5/5/2022**	**6/5/2022**		**US6, US8, US9 (Entity student)**

4	**6/5/2022**	**8/5/2022**		**US7 (connect Course-Student)**
5	**8/5/2022**	**10/5/2022**		**US10, US11 (grading and calculate final grades)**
6	**10/5/2022**	**11/5/2022**		**US12(Statistics dependency)**
7	**11/5/2022**	**18/5/2022**		**Testing / report / video**

Use Cases Diagram



2.3 <US1><LOGIN-REGISTRATION>

Use case ID	US1
Actors	Instructor
Main flow of events	1.The use case starts when the user types the URL of the application in the browser 2.The user fill in the boxes with the username and password 3.The user press the log in button
Alternative flow 1	1.The user is new 2.The user press the (Register here) button 3.The user fill the registration form 4.The user press the registration button 5.The user is notified of the successful registration 6.The user press the (login here) button
Post conditions	The user has log in the application successfully

2.4 <US2><BROWSE_COURSES>

Use case ID	US2
Actors	Instructor
Preconditions	The application is up and running
Main flow of events	1.The use case starts when the user presses the (Log in) button 2.The user can browse the list of courses and see the name, the year, the semester and the syllabus of the courses
Post conditions	The user can add a new course update information of an existing course, delete a course from the list or view the students of a course.

2.5 <US3><ADD_COURSE>

Use case ID	US3
Actors	Instructor
Preconditions	The user must be logged in
Main flow of events	1. The use case starts when the user presses the (Add Course) button 2. The user fills the form with the information of the Course 1.The user enters a course name 2. The user selects the year of the course 3.the user selects the semester of the course 4.The user enters the syllabus of the course 3. The user presses the button save 4. The course added successfully to the course list 5. The view returns to the list of the courses
Post conditions	The new course has added successfully and can be viewed and further modified
Alternative flow	1.The form is filled incorrectly or is empty 2. The application indicates the fields that are empty or incorrect when the user presses save 3.The user corrects the fields as indicated 4. The user presses the button save
Post conditions	The use case continues from step 4 of the main flow

2.6 <US4><DELETE_COURSE>

Use case ID	US4
Actors	Instructor
Preconditions	The user must be logged in The user must have at least a course in his course list
Main flow of events	1.The use case starts when the user is browses the list of courses 2.The user clicks on the (Delete) Button of the course

	3.The course gets deleted from the base and the corresponding list of Students for this course 4.The page refreshes and shows the updated list of courses
Post conditions	The course has been deleted

2.7 <US3><UPDATE_COURSE>

Use case ID	US5
Actors	Instructor
Pre conditions	1.The user must be logged in 2.The user must have at least a course in his course list
Main flow of events	1.the use case starts when the user presses the (update) button on a course 2.The user is redirected to the form filled with the current information of the course 3.The user makes changes to the form as needed 4. The user presses the button save 5. The course updated successfully to the course list 6. The view returns to the list of the courses
Post conditions	The course has been successfully updated
Alternative flow 1	1.The changes to the form are not correct 2. The application indicates the fields that are incorrect when the user presses save 3. The user corrects the fields as indicated 4. The user presses the button save
Post conditions	The use case continues from step 5 of the main flow

2.8 <US6><BROWSE_STUDENTS>

Use case ID	US6
Actors	Instructor
Preconditions	The application is up and running

	The user has at least a course in their course list
Main flow of events	1.The use case starts when the user presses the (View Students) button 2.The user can browse the list of Students and see the First name, the Last name, the year, the semester, the Project grade, the Exam grade and the Final grade of the students
Post conditions	The user can add a new students update information of an existing student, delete a student from the list, set the grades for a student.

2.9 <US7><ADD_STUDENT>

Use case ID	US7
Actors	Instructor
Preconditions	The user must be logged in The application is up and running The user has at least a course in their course list
Main flow of events	1.The use case starts when the user presses the (Add Student) button 2.The user fills the form with the information of the student 1. The user enters the student's first name 2. The user enters the student's last name 3. The user selects the year of Registration of the student 4. The user selects the semester of the student 3.The user presses the button save 4.The course added successfully to the course list 5.The view returns to the list of the courses
Post conditions	The new student has added successfully and can be viewed and further modified
Alternative flow	1.The form is filled incorrectly or is empty 2. The application indicates the fields that are empty or incorrect when the user presses save 3.The user corrects the fields as indicated 4. The user presses the button save

Post conditions	The use case continues from step 4 of the main flow

2.10 <US8><DELETE_STUDENT>

Use case ID	US8
Actors	Instructor
Preconditions	The user must be logged in The user must have at least a course in his course list The course must have at least a student in the student list
Main flow of events	1.The use case starts when the user is browses the list students 2.The user clicks on the (Delete) Button of the student 3.The student gets deleted from the list 4.The page refreshes and shows the updated list of students
Post conditions	The student has been deleted

2.11 <US9><UPDATE_STUDENT>

Use case ID	US9
Actors	Instructor
Preconditions	1.The user must be logged in 2.The course must have at least a student in the student list
Main flow of events	1.The use case starts when the user presses the (Update) button on a student 2.The user is redirected to the form filled with the current information of the student 3.The user makes changes to the form as needed 4. The user presses the button save 5. The student updated successfully to the student list 6. The view returns to the list of the students
Post conditions	The student has been successfully updated

Alternative flow 1	1.The changes to the form are not correct 2. The application indicates the fields that are incorrect when the user presses save 3. The user corrects the fields as indicated 4. The user presses the button save
Post conditions	The use case continues from step 5 of the main flow

2.12 <US10><GRADING>

Use case ID	US10
Actors	Instructor
Preconditions	1.The user must be logged in 2.The course must have at least a student in the student list
Main flow of events	1.The use case starts when the user presses the (Grading) button on a student 2.The user is redirected to the form to fill the grades for the student (project and exam grades) 3. The user presses the button save 4. The students grades saved and can be viewed in the list of students 5. The view returns to the list of the students
Post conditions	The student has been successfully graded
Alternative flow 1	1.The user made a mistake and want to change an existing grade of a student 2. The user presses the (Grading) button on a student which has already been graded 3. The user change the grades 4. The user presses the button save
Post conditions	The use case continues from step 4 of the main flow

2.13 <US11><OVERALL_GRADES>

Use case ID	US11
Actors	Instructor

Preconditions	1.The user must be logged in 2.The user must have at least a course in their list
Main flow of events	1.The use case starts when the user presses the (Calculate Grades) button 2.The user is redirected to the form to fill. 1.The user fills the minimum Project grade. 2.The user fills the minimum exam grade. 3.The user fills the Project weight (%) grade. 4.The user fills the exam weight (%) grade. 3. The user presses the button save 4. The values are saved 5. The view returns to the list of the students
Post conditions	The weights for the course are set and the students will show their final grade when the will be graded
Alternative flow 1	1.The user made a mistake and want to change the weights 2. The user presses the (Calculate Grades) button 3. The user change the grades 4. The user presses the button save
Post conditions	The use case continues from step 4 of the main flow

2.14 <US12><CALCULATE_STATISTICS>

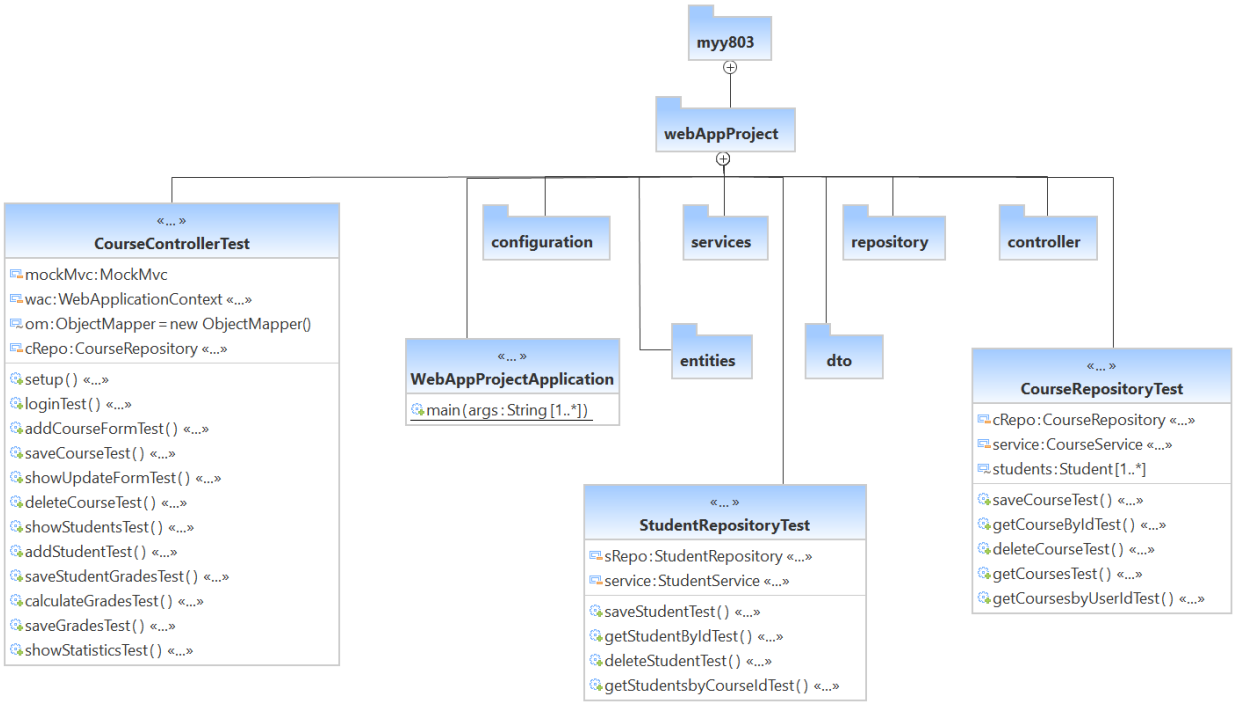
Use case ID	US12
Actors	Instructor
Pre conditions	All the students must be graded The overall grades should have been calculated
Main flow of events	1.The use case begins when the user wants to see the statistics for the course 2. The user presses the (Grade Statistics) button 3.The application is redirecting the user to a page with Statistics
Post conditions	The user can study the distribution of the student's grades in the course.

3 Design

3.1 Architecture

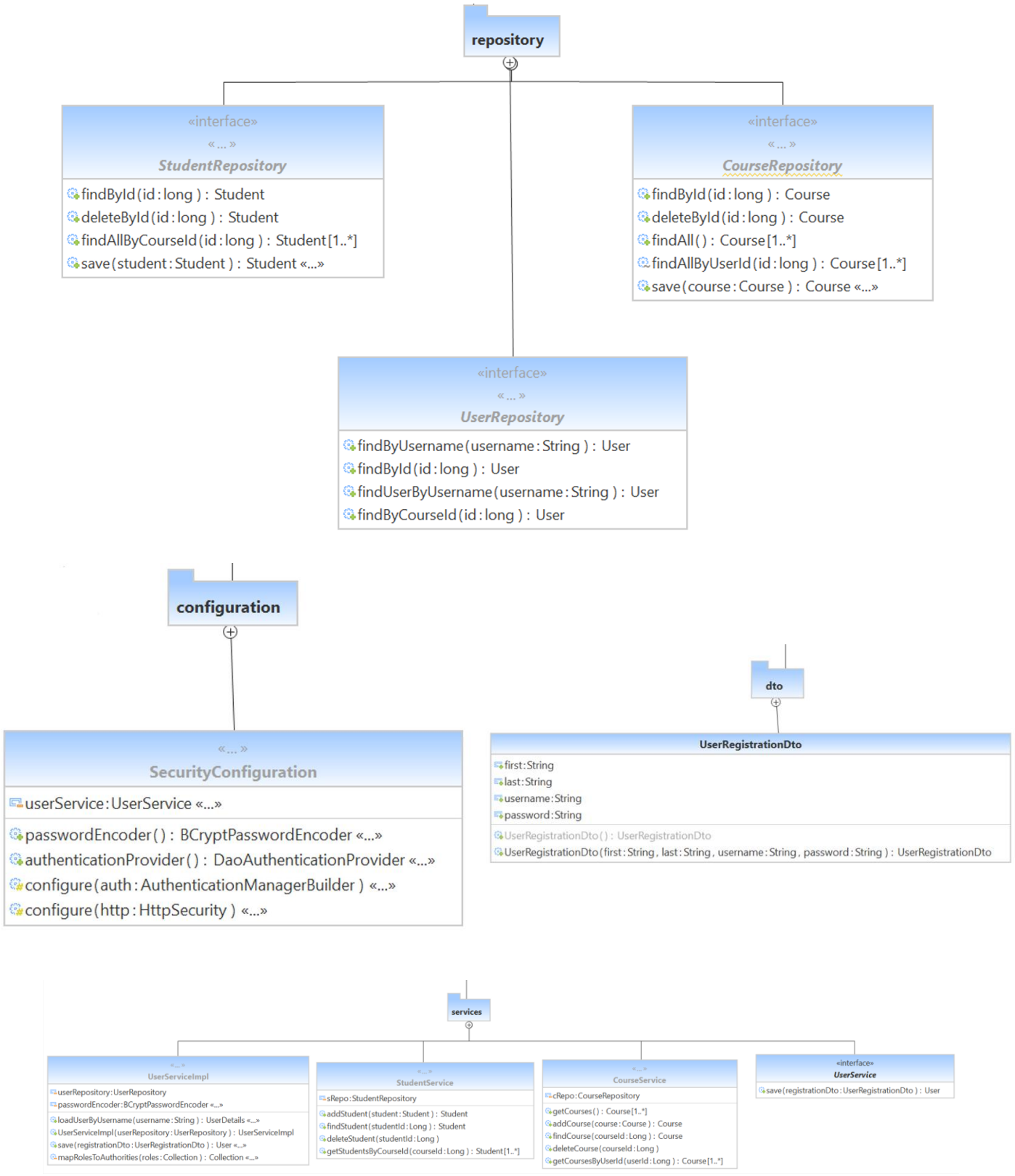
<Specify the overall architecture for this release in terms of a **UML package diagram**.>

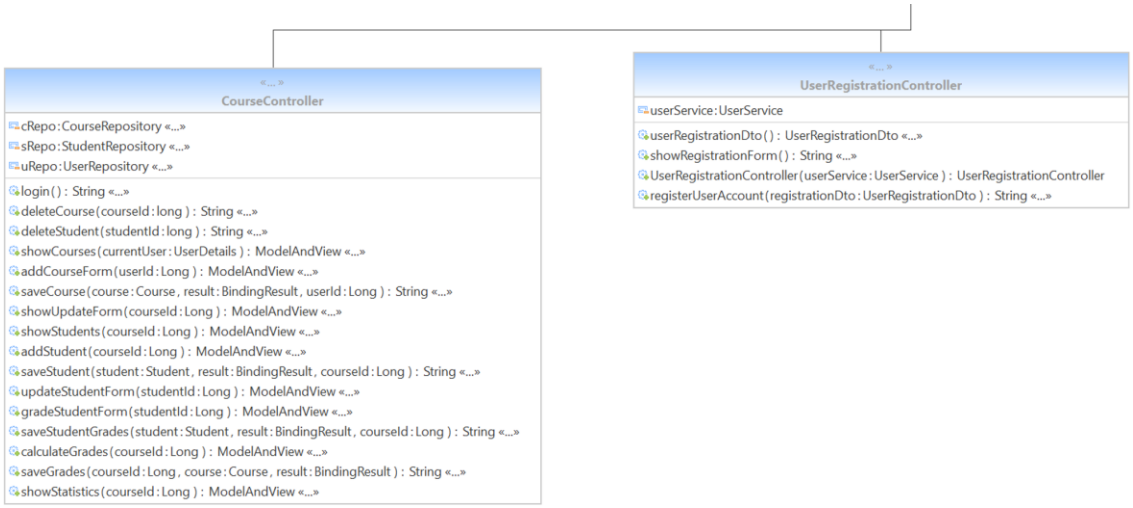
UML package diagram



3.2 Design

UML class diagrams


repository
«interface»
«...»
StudentRepository
findById(id:long):Student
deleteById(id:long):Student
findAllByCourseId(id:long):Student[1..*]
save(student:Student):Student «...»
«interface»
«...»
CourseRepository
findById(id:long):Course
deleteById(id:long):Course
findAll():Course[1..*]
findAllByUserId(id:long):Course[1..*]
save(course:Course):Course «...»
«interface»
«...»
UserRepository
findByUsername(username:String):User
findById(id:long):User
findUserByUsername(username:String):User
findByCourseId(id:long):User
configuration
«...»
SecurityConfiguration
userService:UserService «...»
passwordEncoder():BCryptPasswordEncoder «...»
authenticationProvider():DaoAuthenticationProvider «...»
configure(auth:AuthenticationManagerBuilder) «...»
configure(http:HttpSecurity) «...»
dto
UserRegistrationDto
first:String
last:String
username:String
password:String
UserRegistrationDto():UserRegistrationDto
UserRegistrationDto(first:String, last:String, username:String, password:String):UserRegistrationDto
services
«...»
UserServiceImpl
userRepository:UserRepository
passwordEncoder:BCryptPasswordEncoder «...»
loadUserByUsername(username:String):UserDetails «...»
UserServiceImpl(userRepository:UserRepository):UserServiceImpl
save(registrationDto:UserRegistrationDto):User «...»
mapRolesToAuthorities(roles:Collection):Collection «...»
«...»
StudentService
sRepo:StudentRepository
addStudent(student:Student):Student
findStudent(studentId:Long):Student
deleteStudent(studentId:Long)
getStudentsByCourseId(courseId:Long):Student[1..*]
«...»
CourseService
cRepo:CourseRepository
getCourses():Course[1..*]
addCourse(course:Course):Course
findCourse(courseId:Long):Course
deleteCourse(courseId:Long)
getCoursesByUserId(userId:Long):Course[1..*]
«interface»
UserService
save(registrationDto:UserRegistrationDto):User



Full uml diagram of the project with all the connections .