# CoursesManagementApp

# Sprint Report

**Γεώργιος Μπαντουβάκης 4119**

**Δημήτριος Αντωνίου 4027**

**Λάζαρος Κοσμίδης 4085**

<u>Κωδικοί για το login της εφαρμογής</u>

Username:usertest1 Password:usertest1

Username:usertest2 Password:usertest1

## VERSIONS HISTORY

| Date | Version | Description | Author |
|---|---|---|---|
| **22/03/2022** | v.1 | **Implementation of User Stories 1-5** | **Dimitris Antoniou**<br>**George Bantouvakis**<br>**Lazaros Kosmidis** |
| **22/04/2022** | v.2 | **Implementation of User Stories 6-12** | **Dimitris Antoniou**<br>**George Bantouvakis**<br>**Lazaros Kosmidis** |
| **15/05/2022** | v.3 | **Implementation of Test Classes, Uml Diagram, Package Diagram and Report** | **Dimitris Antoniou**<br>**George Bantouvakis**<br>**Lazaros Kosmidis** |

## 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

<For the user stories included in this release specify below corresponding tests using a typical tabular form.>

### 2.1. Scrum team

| | |
|---|---|
| **Product Owner** | Γεώργιος Μπαντουβάκης, Λάζαρος Κοσμίδης, Δημήτριος Αντωνίου |
| **Scrum Master** | Ζάρρας Απόστολος |
| **Development Team** | Γεώργιος Μπαντουβάκης, Λάζαρος Κοσμίδης, Δημήτριος Αντωνίου |

## 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** | **22-02-2022** | **10-03-2022** | **2** | **1-5** |
| **2** | **28-03-2022** | **14-04-2022** | **2** | **6-11** |
| **3** | **16-04-2022** | **22-04-2022** | **1** | **12** |

## 3. Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>

### 3.1. <Use Case 1>

| | |
|---|---|
| **Use case ID** | SignUpToTheApp |
| **Actors** | The Teacher |
| **Pre conditions** | The User must have an email address. |
| **Main flow of events** | 1. The use case starts when the user press sign up button.<br>2. The teacher must register himself.<br>2.1. The system asks for username.<br>2.2. The system asks for email address.<br>2.3. The systems ask for a password.<br>2.4. The teacher chooses Sign Up to complete his registration. |
| **Alternative flow 1** | At anytime the teacher can decline the registration if he has already an existing account and redirect to Sign In page. |
| **Post** | The system has been updated for the new teacher. The teacher can now sign in |

| | |
|---|---|
| **conditions** | to his account. |

## 3.2 <Use Case 2>

| | |
|---|---|
| **Use case ID** | SingInToAccount |
| **Actors** | The Teacher |
| **Pre conditions** | The user must have created an account in order to be able to log in. |
| **Main flow of events** | 1. The use case starts when the user chooses to be signed in his account.<br>2. The user isn't connected.<br>2.1. The system asks for username.<br>2.2. The system asks for password.<br>2.3. The user presses sign in.<br>2.3.1. If the system don't sign him in then the causes are :<br>1) wrong username or password ,<br>2) user not registered . |
| **Alternative flow 1** | At anytime the user can sign up if he didn't. |
| **Post conditions** | The user enters the home page of his account. |

## 3.3 <Use Case 3>

| | |
|---|---|
| **Use case ID** | BrowseTheCourses |
| **Actors** | The teacher |
| **Pre conditions** | The user must have logged in his account and registered some courses. |
| **Main flow of events** | 1. The use case starts when the user press sign in and have logged in his account. Then he can see the list of Courses. |
| **Alternative flow 1** | The user can log out at any moment when he is at home page. |
| **Post conditions** | The systems show to the user the list of Courses. |

## 3.4<Use Case 4>

| | |
|---|---|
| **Use case ID** | AddCourseAndWeights |
| **Actors** | The teacher |
| **Pre conditions** | The user must have been logged in his account. |
| **Main flow of events** | 1. The use case starts when the users press the + button at his home page.<br>2. The system asks from the teacher<br>2.1. The name of the course.<br>2.2. The CourseId.<br>2.3. The Year.<br>2.4. The Semester. |

| | |
|---|---|
| | 2.5. The Project weight.<br>2.6. The Exams weight.<br>3. After the user inserts all these information, he must press register.<br>4. The systems add the course to the list. |
| **Alternative flow 1** | The user can log out at any moment when he is at home page. |
| **Post conditions** | The system has been updated for the new course. |

## 3.5<Use Case 5>

| | |
|---|---|
| **Use case ID** | DeleteCourse |
| **Actors** | The teacher |
| **Pre conditions** | The user must be logged in and the list of courses must have some courses. |
| **Main flow of events** | 1. The use case starts when the users press the 3 lines and after he clicks Delete.<br>2. The system deletes the course from the list and the students that are registered to this course. |
| **Alternative flow 1** | 1. The user can log out at any moment.<br>2. The user can edit the course that he chooses.<br>3. The user can browse the list of students of a specific course. |
| **Post conditions** | The system has been updated for the deletion of the course. |

## 3.6<Use Case 6>

| | |
|---|---|
| **Use case ID** | UpdateDescriptionOfCourse |
| **Actors** | The teacher |
| **Pre conditions** | The user must have been logged in and the course that needs to be updated must be on the list of courses. |
| **Main flow of events** | 1. The use case starts when the user chooses the 3 lines and after he clicks Edit. |

| | |
|---|---|
| | 2. The system gives to the user the following possibilities<br>2.1. Edit name of course.<br>2.2. Edit CourseId.<br>2.3. Edit Year.<br>2.4. Edit Semester.<br>2.5. Edit Project weight.<br>2.6. Edit Exams weight.<br>3. After he finishes with the edit, user press Save.<br>4. The system stores the changes of the course that user made |
| **Alternative flow 1** | 1. The user can delete a course, he can see the details of a course, and he can make no changes to the course he presses to edit. |
| **Post conditions** | The system has been updated for the edit that user made on that specific course. |

## 3.7 <Use Case 7>

| | |
|---|---|
| **Use case ID** | BrowseStudentsOfCourse |
| **Actors** | The teacher . |
| **Pre conditions** | The user must have been logged in and the list of course must have some the course that he needs to browse students. |
| **Main flow of events** | 1. The use case starts when the user chooses the 3 lines and after he clicks Details.<br>2. The system shows the list of students and their final grade that auto calculated from the System based on their final exam and project grade multiplied with the weights of the course.<br>3. The statistics of their grades from that specific course that the user chooses. |
| **Alternative flow 1** | The user can log out, he can delete or edit the course before he enters the details. Finally, he can press the backwards key up left to go back to home page if he chooses the wrong course. |
| **Post conditions** | The system shows the List of Students and the statistics of their grades. |

## 3.8<Use Case 8>

| | |
|---|---|
| **Use case ID** | AddStudentOnACourseAndHisGrades |
| **Actors** | The teacher |
| **Pre conditions** | The user must have logged in, the specific course that he needs must be registered and he must browse the student list of this course. |
| **Main flow of events** | 1. The use case starts when the user presses the '+' button.<br>2. The system asks from the user the student's information<br>2.1. The email.<br>2.2. The First name.<br>2.3. The Second name.<br>2.4. The Am.<br>2.5. The Exams grade.<br>2.6. The Project grade.<br>3. After the user is done, he must press the button register.<br>4. The system adds the new student to the course. |
| **Alternative flow 1** | The user can return to the home page if he is in the wrong course. |
| **Post conditions** | The system has been updated for the new student in that specific course. |

## 3.9<Use Case 9>

| | |
|---|---|
| **Use case ID** | RemoveStudent |
| **Actors** | The teacher |
| **Pre conditions** | The course must exist, the students at this course must be registered and the user must have been logged in. |
| **Main flow of events** | 1. The use case starts when the user chooses the 3 lines and after he clicks Delete.<br>2. The system deletes the student from that specific course. |
| **Alternative flow 2** | The user can return to the home page if he is in the wrong course. |
| **Post conditions** | The system has been updated for the deletion of the student. |

## 3.10<Use Case 10>

| | |
|---|---|
| **Use case ID** | UpdateStudentInformation |
| **Actors** | The teacher |
| **Pre conditions** | The user must have logged in, the course is created and there students in this course. |
| **Main flow of events** | 1. The use case starts when the user chooses the 3 lines and after he clicks Edit.<br>2. The system gives to the user the following possibilities<br>2.1. Edit Email of the student.<br>2.2. Edit First name. |

| | |
|---|---|
| | 2.3. Edit Second name.<br>2.4. Edit Am.<br>2.5. Edit Project grade.<br>2.6. Edit exams grade.<br>3. After the user finishes with the edit, he presses 'Save'.<br>4. The system saves all the changes that user made. |
| **Alternative flow 1** | The user can anytime press the backwards button to return to home page and select another course if he made a mistake. |
| **Post conditions** | The system has been updated for the changes that have been on student. |

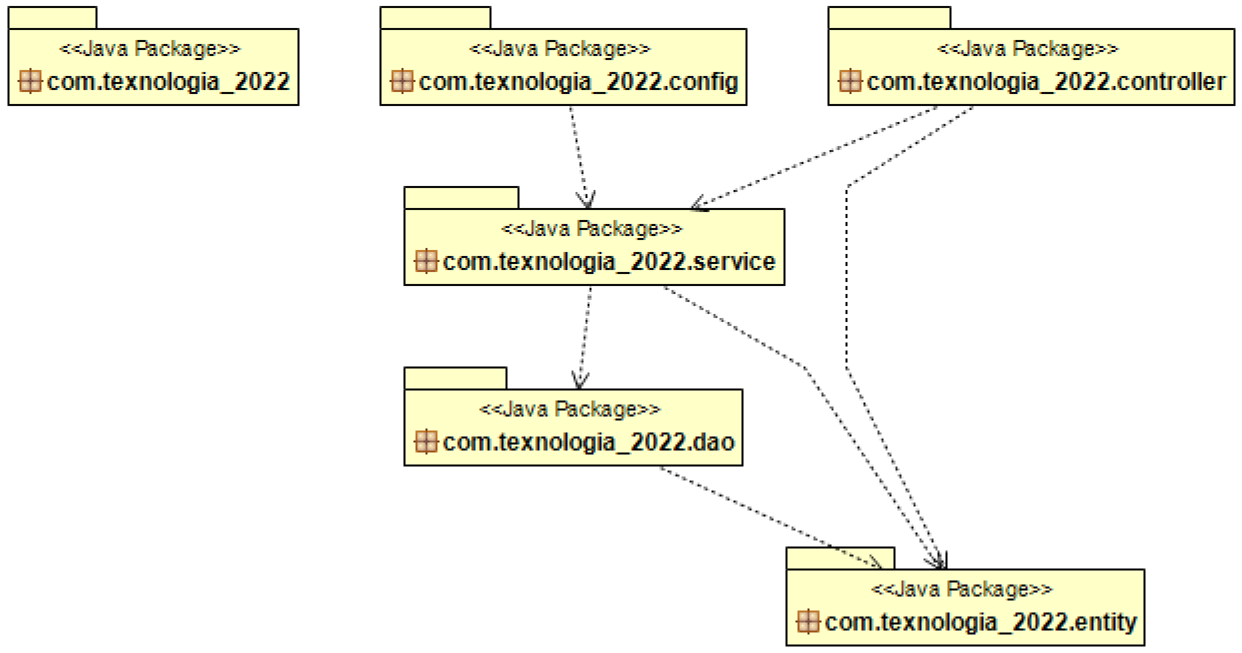
## 3.11<Use Case 11>

| | |
|---|---|
| **Use case ID** | CalculateStatistics |
| **Actors** | The System's Calculator and time |
| **Pre conditions** | The user must have logged in and the course must have some students in it. |
| **Main flow of events** | 1. The use case starts when the user has students in the course.<br>2. The system automatically calculates some statistics<br>2.1. About min grade.<br>2.2. About max grade.<br>2.3. About the mean of the grades.<br>2.4. About standard deviation.<br>2.5. About Variance.<br>2.6. About skewness. |
| **Alternative flow 1** | If there are no students the board of the statistics is Nan. |
| **Post conditions** | The system calculates every statics that the teacher needs to see. |

# Design

## 4.1Architecture

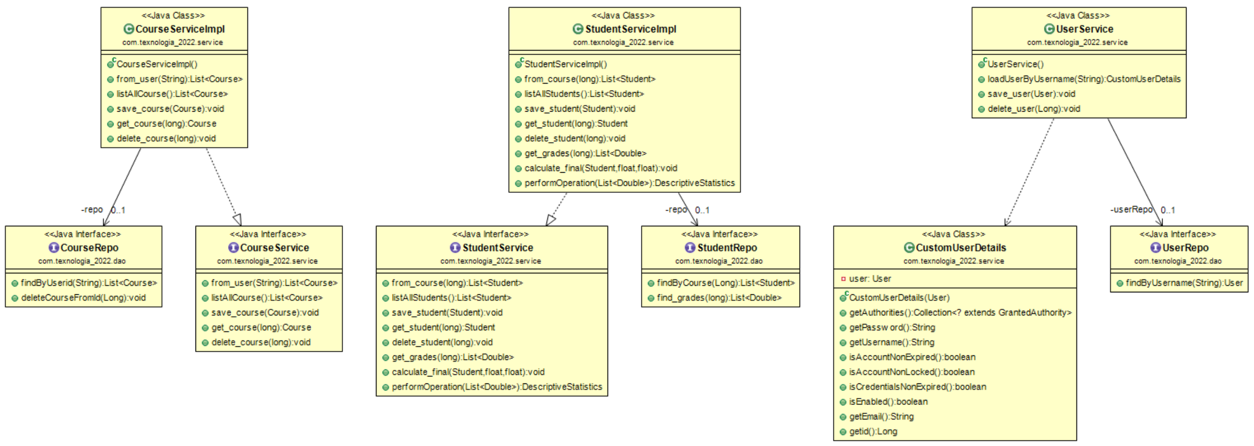
<Specify the overall architecture for this release in terms of a **UML package diagram**.>



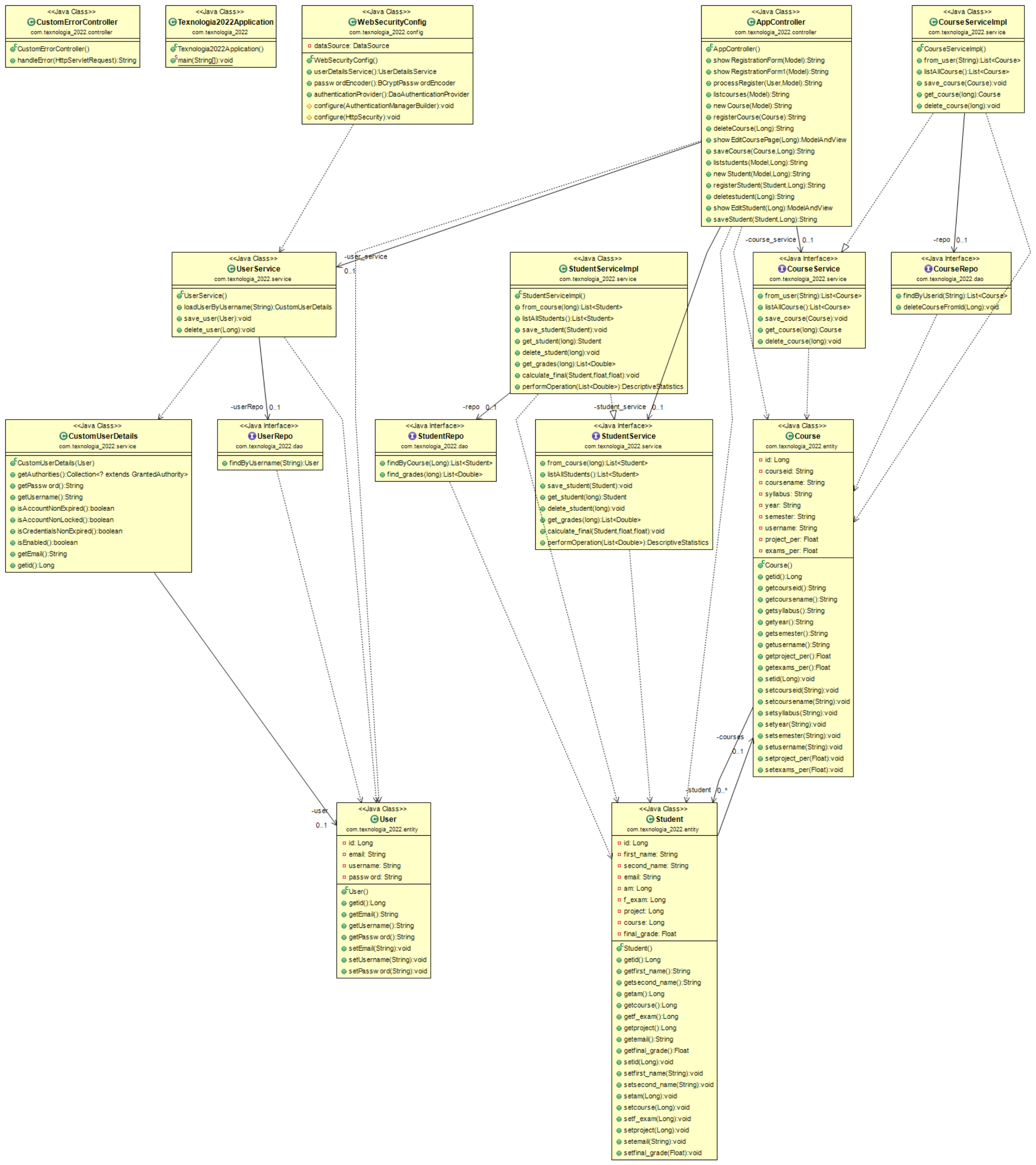
## 4.2Design

<Specify the detailed design for this release in terms of **UML class diagrams**.>

### 1.**UML of Service and Dao**



<<Java Class>>
CourseServiceImpl
com.texnologia_2022.service
CourseServiceImpl()
from_user(String):List<Course>
listAllCourse():List<Course>
save_course(Course):void
get_course(long):Course
delete_course(long):void
<<Java Class>>
StudentServiceImpl
com.texnologia_2022.service
StudentServiceImpl()
from_course(long):List<Student>
listAllStudents():List<Student>
save_student(Student):void
get_student(long):Student
delete_student(long):void
get_grades(long):List<Double>
calculate_final(Student,float,float):void
performOperation(List<Double>):DescriptiveStatistics
<<Java Class>>
UserService
com.texnologia_2022.service
UserService()
loadUserByUsername(String):CustomUserDetails
save_user(User):void
delete_user(Long):void
-repo
0..1
<<Java Interface>>
CourseRepo
com.texnologia_2022.dao
findByUserid(String):List<Course>
deleteCourseFromId(Long):void
<<Java Interface>>
CourseService
com.texnologia_2022.service
from_user(String):List<Course>
listAllCourse():List<Course>
save_course(Course):void
get_course(long):Course
delete_course(long):void
<<Java Interface>>
StudentService
com.texnologia_2022.service
from_course(long):List<Student>
listAllStudents():List<Student>
save_student(Student):void
get_student(long):Student
delete_student(long):void
get_grades(long):List<Double>
calculate_final(Student,float,float):void
performOperation(List<Double>):DescriptiveStatistics
-repo
0..1
<<Java Interface>>
StudentRepo
com.texnologia_2022.dao
findByCourse(Long):List<Student>
find_grades(long):List<Double>
<<Java Class>>
CustomUserDetails
com.texnologia_2022.service
user: User
CustomUserDetails(User)
getAuthorities():Collection<? extends GrantedAuthority>
getPassword():String
getUsername():String
isAccountNonExpired():boolean
isAccountNonLocked():boolean
isCredentialsNonExpired():boolean
isEnabled():boolean
getEmail():String
getid():Long
-userRepo
0..1
<<Java Interface>>
UserRepo
com.texnologia_2022.dao
findByUsername(String):User



<<Java Class>>
CustomErrorController
com.texnologia_2022.controller
CustomErrorController()
handleError(HttpServletRequest):String
<<Java Class>>
Texnologia2022Application
com.texnologia_2022
Texnologia2022Application()
main(String[]):void
<<Java Class>>
WebSecurityConfig
com.texnologia_2022.config
dataSource: DataSource
WebSecurityConfig()
userDetailsService():UserDetailsService
passw ordEncoder():BCryptPassw ordEncoder
authenticationProvider():DaoAuthenticationProvider
configure(AuthenticationManagerBuilder):void
configure(HttpSecurity):void
<<Java Class>>
AppController
com.texnologia_2022.controller
AppController()
show RegistrationForm(Model):String
show RegistrationForm1(Model):String
processRegister(User,Model):String
listcourses(Model):String
new Course(Model):String
registerCourse(Course):String
deleteCourse(Long):String
show EditCoursePage(Long):ModelAndView
saveCourse(Course,Long):String
liststudents(Model,Long):String
new Student(Model,Long):String
registerStudent(Student,Long):String
deletestudent(Long):String
show EditStudent(Long):ModelAndView
saveStudent(Student,Long):String
<<Java Class>>
CourseServiceImpl
com.texnologia_2022.service
CourseServiceImpl()
from_user(String):List<Course>
listAllCourse():List<Course>
save_course(Course):void
get_course(long):Course
delete_course(long):void
-user_service
0..1
<<Java Class>>
UserService
com.texnologia_2022.service
UserService()
loadUserByUsername(String):CustomUserDetails
save_user(User):void
delete_user(Long):void
<<Java Class>>
StudentServiceImpl
com.texnologia_2022.service
StudentServiceImpl()
from_course(long):List<Student>
listAllStudents():List<Student>
save_student(Student):void
get_student(long):Student
delete_student(long):void
get_grades(long):List<Double>
calculate_final(Student,float,float):void
performOperation(List<Double>):DescriptiveStatistics
-course_service
0..1
<<Java Interface>>
CourseService
com.texnologia_2022.service
from_user(String):List<Course>
listAllCourse():List<Course>
save_course(Course):void
get_course(long):Course
delete_course(long):void
-repo
0..1
<<Java Interface>>
CourseRepo
com.texnologia_2022.dao
findByUserid(String):List<Course>
deleteCourseFromId(Long):void
-userRepo
0..1
-repo
0..1
-student_service
0..1
<<Java Class>>
CustomUserDetails
com.texnologia_2022.service
CustomUserDetails(User)
getAuthorities():Collection<? extends GrantedAuthority>
getPassw ord():String
getUsername():String
isAccountNonExpired():boolean
isAccountNonLocked():boolean
isCredentialsNonExpired():boolean
isEnabled():boolean
getEmail():String
getid():Long
<<Java Interface>>
UserRepo
com.texnologia_2022.dao
findByUsername(String):User
<<Java Interface>>
StudentRepo
com.texnologia_2022.dao
findByCourse(Long):List<Student>
find_grades(long):List<Double>
<<Java Interface>>
StudentService
com.texnologia_2022.service
from_course(long):List<Student>
listAllStudents():List<Student>
save_student(Student):void
get_student(long):Student
delete_student(long):void
get_grades(long):List<Double>
calculate_final(Student,float,float):void
performOperation(List<Double>):DescriptiveStatistics
<<Java Class>>
Course
com.texnologia_2022.entity
id: Long
courseid: String
coursename: String
syllabus: String
year: String
semester: String
username: String
project_per: Float
exams_per: Float
Course()
getid():Long
getcourseid():String
getcoursename():String
getsyllabus():String
getyear():String
getsemester():String
getusername():String
getproject_per():Float
getexams_per():Float
setid(Long):void
setcourseid(String):void
setcoursename(String):void
setsyllabus(String):void
setyear(String):void
setsemester(String):void
setusername(String):void
setproject_per(Float):void
setexams_per(Float):void
-courses
0..1
-student
0..*
-user
0..1
<<Java Class>>
User
com.texnologia_2022.entity
id: Long
email: String
username: String
passw ord: String
User()
getid():Long
getEmail():String
getUsername():String
getPassw ord():String
setEmail(String):void
setUsername(String):void
setPassw ord(String):void
<<Java Class>>
Student
com.texnologia_2022.entity
id: Long
first_name: String
second_name: String
email: String
am: Long
f_exam: Long
project: Long
course: Long
final_grade: Float
Student()
getid():Long
getfirst_name():String
getsecond_name():String
getam():Long
getcourse():Long
getf_exam():Long
getproject():Long
getemail():String
getfinal_grade():Float
setid(Long):void
setfirst_name(String):void
setsecond_name(String):void
setam(Long):void
setcourse(Long):void
setf_exam(Long):void
setproject(Long):void
setemail(String):void
setfinal_grade(Float):void

<Document the classes that are included in this release in terms of CRC cards according to the template that is given below.>

| **Class Name: User** | |
|---|---|
| **Responsibilities:** | **Collaborations:** |
| ▪ Initialize an object of type User.<br>▪ Create getters.<br>▪ Create setters. | ▪ Appcontroller<br>▪ UserService<br>▪ UserRepo<br>▪ CustomUserDetails |

| **Class Name: Course** | |
|---|---|
| **Responsibilities:** | **Collaborations:** |
| ▪ Initialize an object of type Course.<br>▪ Create getters.<br>▪ Create setters. | ▪ Appcontroller<br>▪ CourseService<br>▪ CourseServiceImpl<br>▪ CourseRepo<br>▪ Student |

| **Class Name: Student** | |
|---|---|
| **Responsibilities:** | **Collaborations:** |
| ▪ Initialize an object of type Student.<br>▪ Create getters.<br>▪ Create setters. | ▪ Appcontroller<br>▪ StudentService<br>▪ StudentServiceImpl<br>▪ StudentRepo<br>▪ Course |

| **Class Name: UserRepo** | |
|---|---|
| **Responsibilities:** | **Collaborations:** |
| ▪ Define the database operations which map domain objects to database table row data. | ▪ UserService<br>▪ User |

| **Class Name: CourseRepo** | |
|---|---|
| **Responsibilities:** | **Collaborations:** |
| ▪ Define the database operations which map domain objects to database table row data. | ▪ CourseServiceImpl<br>▪ Course |

| **Class Name: StudentRepo** | |
|---|---|
| **Responsibilities:** | **Collaborations:** |
| ▪ Define the database operations which map domain objects to database table row data. | ▪ StudentServiceImpl<br>▪ Student |

| **Class Name: UserService** | |
|---|---|
| **Responsibilities:** | **Collaborations:** |
| ▪ Define the operations of the services that are provided by the application to the users. | ▪ WebSecurityConfig<br>▪ AppController<br>▪ CustomUserDetails<br>▪ UserRepo<br>▪ User |

| **Class Name: CustomUserDetails** | |
|---|---|
| **Responsibilities:**<br>▪ Define the operations of the services that are provided by the application to the users. | **Collaborations:**<br>▪ UserService<br>▪ User |

| **Class Name: CourseService** | |
|---|---|
| **Responsibilities:**<br>▪ Define the operations of the services that are provided by the application to the users. | **Collaborations:**<br>▪ Course<br>▪ AppController<br>▪ CourseServiceImpl |

| **Class Name: CourseServiceImpl** | |
|---|---|
| **Responsibilities:**<br>▪ Implemantation of CourseService. | **Collaborations:**<br>▪ CourseService<br>▪ CourseRepo<br>▪ Course |

| **Class Name: StudentService** | |
|---|---|
| **Responsibilities:**<br>▪ Define the operations of the services that are provided by the application to the users. | **Collaborations:**<br>▪ StudentServiceImpl<br>▪ AppController<br>▪ Student |

| Class Name: StudentServiceImpl | |
|---|---|
| **Responsibilities:** <br> ▪ Implemantation of StudentService. | **Collaborations:** <br> ▪ StudentRepo <br> ▪ StudentService <br> ▪ Student |

| Class Name: WebSecurityConfig | |
|---|---|
| **Responsibilities:** <br> ▪ Configuration of security parameters that already defined in spring boot framework. | **Collaborations:** <br> ▪ UserService <br> ▪ .... |

| Class Name: AppController | |
|---|---|
| **Responsibilities:** <br> ▪ Handling all http requests and direct to the correst pages. | **Collaborations:** <br> ▪ UserService <br> ▪ User <br> ▪ StudentService <br> ▪ Student <br> ▪ Course <br> ▪ CourseService |

| Class Name: CustomErrorController | |
|---|---|
| **Responsibilities:** <br> ▪ Redirecting of error 404 and 500 to custom pages. | **Collaborations:** <br> ▪ ... |

| Class Name: Texnologia2022Application | |
|---|---|
| **Responsibilities:** <br> ▪ main class of the project. | **Collaborations:** <br> ▪ ... |