CoursesManagementApp

Sprint Report

Team:
Αγγελική Γκαβαρδίνα, A.M.:4042
Δημήτριος Μπαλδεράνης, A.M.:4116
Ιωάννα Κόκκαλη, A.M.:4268

VERSIONS HISTORY

Date	Version	Description	Author
2/14/22	<1.0>	Displayable project	The Team

1 Introduction

This document provides information concerning the **4st** sprint of the project.

1.1 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	A. Zarras
Scrum Master	A. Gkavardina
Development Team	The Team

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**15/4/22**	**22/4/22**	**1**	**US1, US2**
2	**22/4/22**	**29/4/22**	**1**	**US3, US4, US5**
3	**29/4/22**	**8/5/22**	**1**	**US6, US7, US8, US9**
4	**8/5/22**	**17/5/22**	**1**	**US10, US11, US12**

3 Use Cases

3.1 Login

Use case ID	UC1
Actors	Teacher
Pre conditions	The Teacher is registered and has user name and password
Main flow of events	1. The Teacher types “localhost:8080” in his browser 2. The system displays a login forum 3. The Teacher types his credentials and submit button
Alternative flow 1	If the Teacher types incorrect password - The system displays an error message
Post conditions	The Teacher can now securely use the application

3.2 ShowCourseList

Use case ID	UC2
Actors	Teacher
Pre conditions	The Teacher has logged in
Main flow of events	1. The system displays a list of courses 2. The Teacher selects an action 2.1.1. The Teacher selects to create a new course 2.1.2. The system redirects the Teacher to a creation form 2.2.1. The Teacher selects to update a course from the list 2.2.2. The system redirects the Teacher to an update form 2.3.1. The Teacher selects to delete a course 2.3.2. The system removes a selected course and refreshes the list 2.4.1. The Teacher selects to display the assigned students of a course

	2.4.2. The system redirects the Teacher to the list of assigned students
Post conditions	The Teacher can create, read, update or delete a course

3.3 ChangeCourse

Use case ID	UC3
Actors	Teacher
Pre conditions	The Teacher has selected to create or update a course at the courses list page
Main flow of events	1. The Teacher selects to create 1.1. The system displays an empty course form 1.2. The Teacher fills the forum fields and presses submit 1.3. The system stores the new course 2. The Teacher selects to update 2.1. The system displays a course form filled with the selected course fields 2.2. The Teacher changes some fields and presses submit 2.3. The system stores the updated course
Alternative flow 1	The Teacher has changed his mind - The Teacher selects to go back - The system redirects the Teacher to the previous page
Post conditions	The courses list contains the new/updated course

3.4 ShowStudentList

Use case ID	UC4
Actors	Teacher

Pre conditions	The Teacher has selected to display the students of a course
Main flow of events	1. The system displays a list of students 1.1. The system displays details of the students 1.2. The system displays grades (project, final exam), the weight of the project and the total score of the student in the selected project 2. The Teacher selects an action 2.1.1. The Teacher selects to create a new student 2.1.2. The system redirects the Teacher to a creation form 2.2.1. The Teacher selects to update a student from the list 2.2.2. The system redirects the Teacher to an update form 2.3.1. The Teacher selects to delete a student 2.3.2. The system removes a selected student and refreshes the list 2.4.1. The Teacher selects to display the statistics of a course 2.4.2. The system redirects the Teacher to a page that contains the statistics
Alternative flow 1	The Teacher has changed his mind - The Teacher selects to go back - The system redirects the Teacher to the previous page
Post conditions	The Teacher creates, updates or deletes a student or displays statistics of a course

3.5 ChangeStudent

Use case ID	UC5
Actors	Teacher
Pre conditions	The Teacher has selected to create or update a student at the students list page
Main flow of events	1. The Teacher selects to create 1.1. The system displays an empty student and grade form 1.2. The Teacher fills the forum fields and presses submit

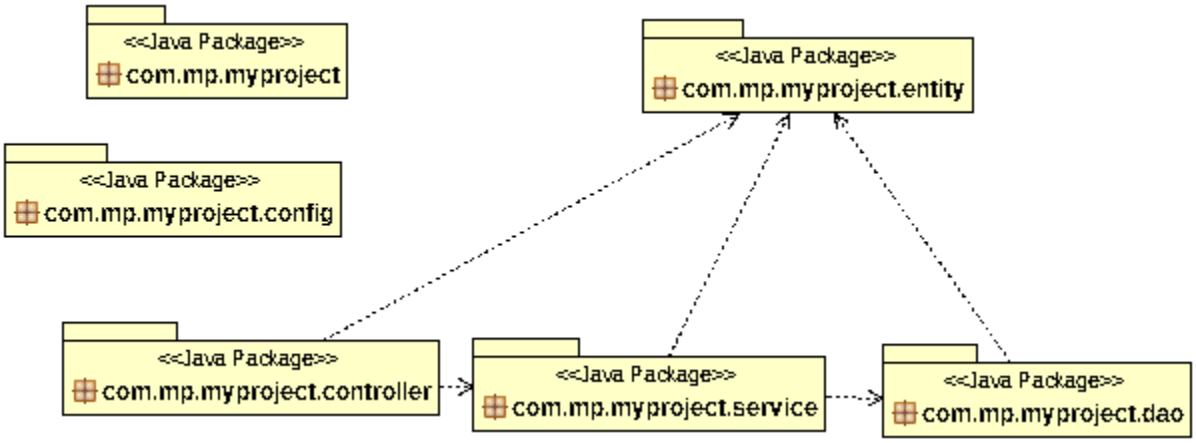
	1.3. The system stores the new student 2. The Teacher selects to update 2.1. The system displays a student and grade form filled with the selected student fields 2.2. The Teacher changes some fields and presses submit 2.3. The system stores the updated student
Alternative flow 1	The Teacher has changed his mind - The Teacher selects to go back - The system redirects the Teacher to the previous page
Post conditions	The students list contains the new/updated course

3.6 CalculateStatistics

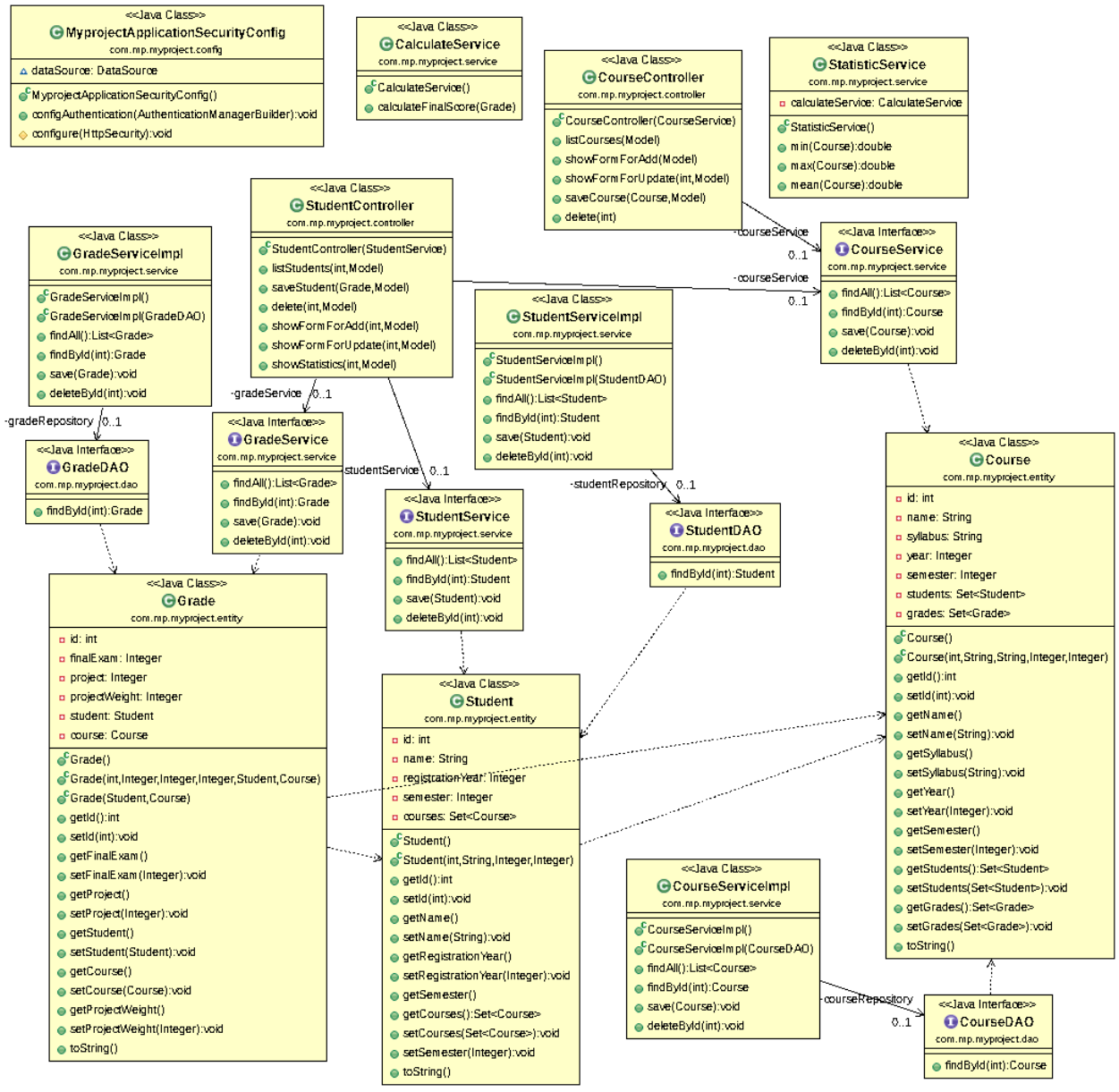
Use case ID	UC6
Actors	Teacher
Pre conditions	The Teacher is at the student-list
Main flow of events	1. The Teacher selects to show the statistics of the shown course 2. The System shows a list of statistics
Alternative flow 1	The Teacher has changed his mind - The Teacher selects to go back - The system redirects the Teacher to the previous page
Post conditions	The teacher can use the shown statistics

4 Design

4.1 Architecture



4.2 Design


<<Java Class>>
MyprojectApplicationSecurityConfig
com.mp.myproject.config
dataSource: DataSource
MyprojectApplicationSecurityConfig()
configAuthentication(AuthenticationManagerBuilder):void
configure(HttpSecurity):void
<<Java Class>>
CalculateService
com.mp.myproject.service
CalculateService()
calculateFinalScore(Grade)
<<Java Class>>
CourseController
com.mp.myproject.controller
CourseController(CourseService)
listCourses(Model)
showFormForAdd(Model)
showFormForUpdate(int,Model)
saveCourse(Course,Model)
delete(int)
<<Java Class>>
StatisticService
com.mp.myproject.service
calculateService: CalculateService
StatisticService()
min(Course):double
max(Course):double
mean(Course):double
<<Java Class>>
StudentController
com.mp.myproject.controller
StudentController(StudentService)
listStudents(int,Model)
saveStudent(Grade,Model)
delete(int,Model)
showFormForAdd(int,Model)
showFormForUpdate(int,Model)
showStatistics(int,Model)
-courseService
0..1
-courseService
0..1
<<Java Interface>>
CourseService
com.mp.myproject.service
findAll():List<Course>
findById(int):Course
save(Course):void
deleteById(int):void
<<Java Class>>
GradeServiceImpl
com.mp.myproject.service
GradeServiceImpl()
GradeServiceImpl(GradeDAO)
findAll():List<Grade>
findById(int):Grade
save(Grade):void
deleteById(int):void
-gradeRepository
0..1
-gradeService
0..1
<<Java Class>>
StudentServiceImpl
com.mp.myproject.service
StudentServiceImpl()
StudentServiceImpl(StudentDAO)
findAll():List<Student>
findById(int):Student
save(Student):void
deleteById(int):void
<<Java Interface>>
GradeDAO
com.mp.myproject.dao
findById(int):Grade
<<Java Interface>>
GradeService
com.mp.myproject.service
findAll():List<Grade>
findById(int):Grade
save(Grade):void
deleteById(int):void
studentService
0..1
-studentRepository
0..1
<<Java Interface>>
StudentService
com.mp.myproject.service
findAll():List<Student>
findById(int):Student
save(Student):void
deleteById(int):void
<<Java Interface>>
StudentDAO
com.mp.myproject.dao
findById(int):Student
<<Java Class>>
Course
com.mp.myproject.entity
id: int
name: String
syllabus: String
year: Integer
semester: Integer
students: Set<Student>
grades: Set<Grade>
Course()
Course(int,String,String,Integer,Integer)
getId():int
setId(int):void
getName()
setName(String):void
getSyllabus()
setSyllabus(String):void
getYear()
setYear(Integer):void
getSemester()
setSemester(Integer):void
getStudents():Set<Student>
setStudents(Set<Student>):void
getGrades():Set<Grade>
setGrades(Set<Grade>):void
toString()
<<Java Class>>
Grade
com.mp.myproject.entity
id: int
finalExam: Integer
project: Integer
projectWeight: Integer
student: Student
course: Course
Grade()
Grade(int,Integer,Integer,Integer,Student,Course)
Grade(Student,Course)
getId():int
setId(int):void
getFinalExam()
setFinalExam(Integer):void
getProject()
setProject(Integer):void
getStudent()
setStudent(Student):void
getCourse()
setCourse(Course):void
getProjectWeight()
setProjectWeight(Integer):void
toString()
<<Java Class>>
Student
com.mp.myproject.entity
id: int
name: String
registrationYear: Integer
semester: Integer
courses: Set<Course>
Student()
Student(int,String,Integer,Integer)
getId():int
setId(int):void
getName()
setName(String):void
getRegistrationYear()
setRegistrationYear(Integer):void
getSemester()
getCourses():Set<Course>
setCourses(Set<Course>):void
setSemester(Integer):void
toString()
<<Java Class>>
CourseServiceImpl
com.mp.myproject.service
CourseServiceImpl()
CourseServiceImpl(CourseDAO)
findAll():List<Course>
findById(int):Course
save(Course):void
deleteById(int):void
-courseRepository
0..1
<<Java Interface>>
CourseDAO
com.mp.myproject.dao
findById(int):Course