UNIVERSIDAD POLITECNICA DE VALENCIA

Labs

Lab Bulletin
Sessions 5 & 6

Use Cases Implementation.

GestAca

Software Engineering
Computer Science School
DSIC – UPV

Year 2024-2025

Use cases Implementation

Before implementing the use cases the team must have finished the work of the previous sessions: objects design and persistence with Entity Framework validating the implementation with the tests provided.

Additional code to initialize the database with data will be available in PoliformaT. This code should be included in your project and run to initialize the system.

The use cases to be developed are:

- **Assign teacher to taught course** (actor: Administrator)
- **Assign classroom to taught course** (actor: Administrator)
- **Add Enrollment** (actor: Employee)
- **List students of taught course** (actor: Employee)

Next to each user story we have indicated the name of the actor who initiates it. See the figure below for the graphical description of the use cases to implement.

> **NOTE**: the functionality of these use cases must be implemented in sessions 5 and 6. The final delivery will be made in session 9 (including the graphical user interface). Check the course syllabus for the exact delivery date.

Next, we present the use cases that must be implemented with their specification templates and some implementation details to consider.

Description of the use cases to be developed

The description of the use cases is additional information to that already given in the general system description.

Next, a partial UML use cases diagram is provided containing the main functionalities to be implemented.

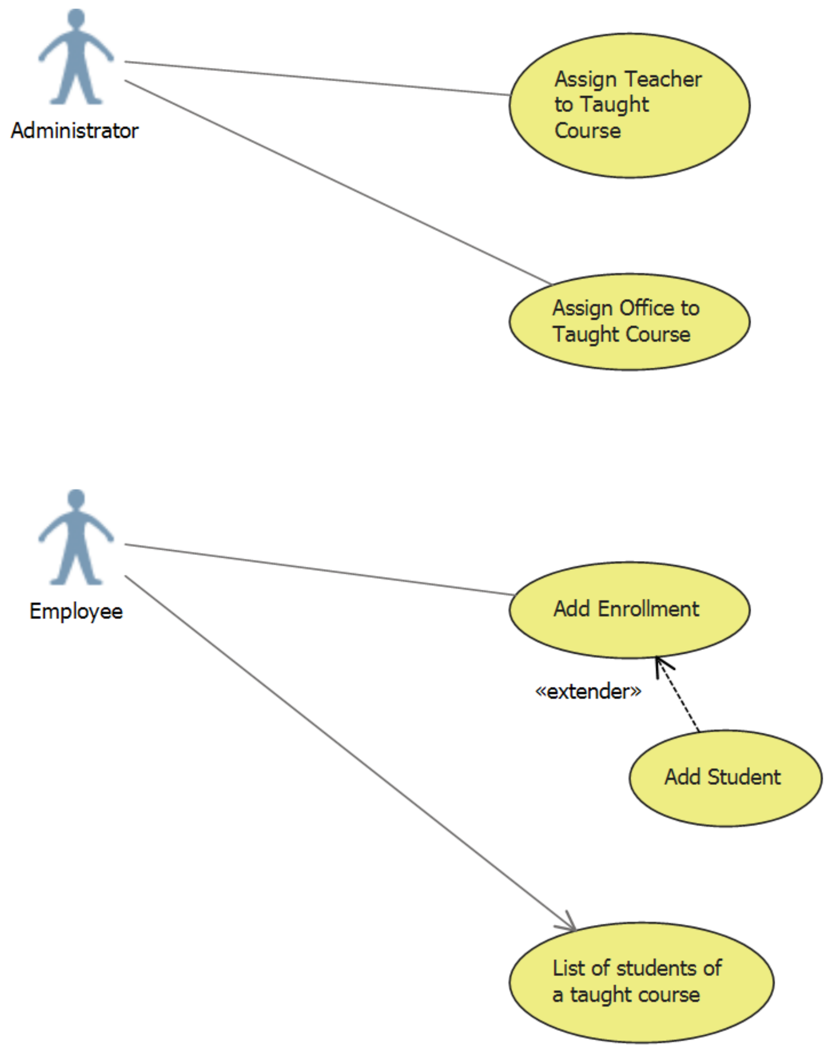
Administrator
Assign Teacher to Taught Course
Assign Office to Taught Course
Employee
Add Enrollment
«extender»
Add Student
List of students of a taught course

Figure 1 Use cases model

Next, we provide the description templates of each use case.

ID	1	
Use Case	Assign teacher to taught course	
Actors	Administrator	
Goal	Adding a teacher to an existing taught course	
Summary	An administrator adds a new teacher for a taught course	
Precond	-	
Postcond	A lecturer is assigned to a taught course	
Includes		
Extends		
Inherits from		
Steps	**User intentions**	**System obligations**
	1. The user requests assigning a teacher to a taught course	2. The system shows all existing taught courses
	3. The user selects a taught course	4. The system shows detailed information of the taught course including whether it already has a teacher assigned
		5. The system shows the list of teachers that can teach the course attending to the time and date restrictions that may exist. A teacher may not teach a taught course if any of its sessions collide with another taught course he or she is already assigned to.
	6. The user selects the teacher	7. The system assigns the teacher to the taught course
Synchronous Extensions		
Asynchronous Extensions		

ID	2	
Use Case	Assign classroom to taught course	
Actors	Administrator	
Goal	Have a taught course with a classroom for lectures	
Summary	An adequate classroom is assigned to an existing taught course	
Precond		
Postcond	The classroom is assigned to the taught course	
Includes		
Extends		
Inherits from		
Steps	**User intentions**	**System obligations**
	1. The user requests assigning	2. The system shows all the existing

	a classroom to a taught course	taught courses
	3. The user selects a taught course	4. The system shows the detailed information of the taught course including whether it already has an assigned classroom or not
		5. The system shows all the classrooms where the course may be taught attending to the possible time and space restrictions. (A course cannot be taught in a classroom if it is already occupied in the period and time slots the course will be taught. In addition, the classroom is not valid if its capacity is lower than the number of enrolled students.
	6. The user selects the classroom	7. The system records the assigned classroom
Synchronous Extensions		
Asynchronous Extensions		

ID	3	
Use Case	Add enrollment	
Actors	Employee	
Goal	Enroll a student in a taught course	
Summary	A student is enrolled in an existing course that is going to be taught.	
Precond		
Postcond	The student is enrolled in the course	
Includes		
Extends	Add student	
Inherits from		
Steps	**User intentions**	**System obligations**
	1. The user requests enrolling a student	2. The system shows the list of taught courses whose start date is after the current date
	3. The user selects a taught course	4. The system requests the student id
	5. The user provides the student id	6. The system finds the student and shows its detailed information
	7. The user confirms the student enrollment in the selected taught course	8. The system records the enrollment
Synchronous Extensions	If at 5 the student is not found the system will ask whether a new student must be added. If so, the student info will be requested, and the student will be added	
	At 5 if the user is already enrolled in the taught course the system will display an error message	

	At 6, if there is an assigned classroom and its capacity is exceeded the system will display an error message and another taught course may be selected	
Asynchronous Extensions	-	-

ID	4	
Use Case	List students of taught course	
Actors	Employee	
Goal	Obtain the list of all students enrolled in a taught course	
Summary	The user selects a taught course and the system show all the enrolled students.	
Precond		
Postcond		
Includes		
Extends		
Inherits from		
Steps	**User intentions**	**System obligations**
	1. The user requests listing all the enrolled students	2. The system shows the list of all taught courses
	3. The user selects a taught course	4. The system shows the enrolled students showing their names and whether the payment is with monthly quotas or with a unique payment
Synchronous Extensions		
Asynchronous Extensions		

Business Logic Implementation Details

In the previous sessions you must have built the functionality related to the design of classes and their persistence.

In this sprint (sessions 5 to 6) you must implement a controller (a services provider) of the business logic layer. This controller must offer the upper layer (UI) access to any needed functionality.

The set of services offered to the UI must be defined as an interface called `IGestAcaService`. The methods of this interface must be implemented by a class `GestAcaService`. The interface and the class must be placed inside a folder `BusinessLogic/Services` and must have as namespace `GestAca.Services`.

All errors that may arise during the execution of the implemented methods must be reported by means of exceptions. To report the errors of the business logic layer you must create a subclass of `Exception` called `ServiceException` in the `BusinessLogic/Services` folder. When you create objects of this class, the `Message` property of this class will hold the error message.

For instance, in the case study available in PoliformaT *VehicleRental_ISW_2017*, when adding a person it would be an error to add another person with the same Id as an already existing person. Next, a fragment of code is shown to illustrate how an exception would be raised in such a situation.

```
public void addPerson(Person person)
{
    if (dal.GetById<Person>(person.Dni) == null)
    {
        dal.Insert<Person>(person);
        dal.Commit();
    }
    else throw new ServiceException("Person already exists.");
}
```

Good coding practices

It is important to write code following good practices to ease code maintenance and make code more readable.

The code written must follow good practices such as “code delegation” from the service class to the other classes of the business logic layer or “code reuse” when this is possible. These practices will be considered when grading the final project.

.

Delivery date

The delivery date will be in the last laboratory session[1]. **ALL** team members must **be present during the evaluation** to answer questions about the project development.

[1] Consider that the final delivery must contain the graphical user interfaces and that they will be implemented during the final two lab sessions.