UNIVERSIDAD POLITECNICA DE VALENCIA

Pràctiques de Laboratori

Sessions 5 i 6 – Implementació de Casos d'ús

- GestAca -

Enginyeria del Programari

ETS Enginyeria Informàtica
DSIC – UPV

Curs 2024-2025

Implementació de Casos d'Ùs

Abans de començar la implementació dels casos d'ús, l'equip haurà d'haver finalitzat el treball de les sessions prèvies: implementació del disseny de classes de la capa lògica (sessions 2-3) i de la persistència amb *Entity Framework* (sessió 4) i verificació del seu correcte funcionament mitjançant els test de prova (es a dir, tot el Lliurament 1).

Es proporcionarà codi, en Poliformat, per a inicialitzar la base de dades amb informació d'alguns professors i aules. Este codi es deu incloure en el vostre projecte e invocar-lo al inicialitzar el sistema.

Els casos d'ús que han de desenvolupar-se són els següents:
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)

Al costat de cada cas d'ús hem indicat entre parèntesi el nom de l'actor iniciador.

NOTA: la funcionalitat d'aquests casos d'ús s'haurà d'implementar en les sessions 5 i 6. El lliurament es farà a la sessió 9 de laboratori (que en el grup 3A correspon al dia 11 de desembre de 2024 (data provisional fins que es confirme el nou calendari de exàmens parcials)).

A continuació es presenta el model de casos d'ús a implementar junt amb les seues plantilles de especificació, així com alguns detalls d'implementació a tindre en compte. A partir de la descripció inicial del cas d'estudi hi han molts mes casos d´ús, però no es demana la seua implantació, per lo que no apareixen a aquest butlletí.

Descripció dels casos d'ús a implementar

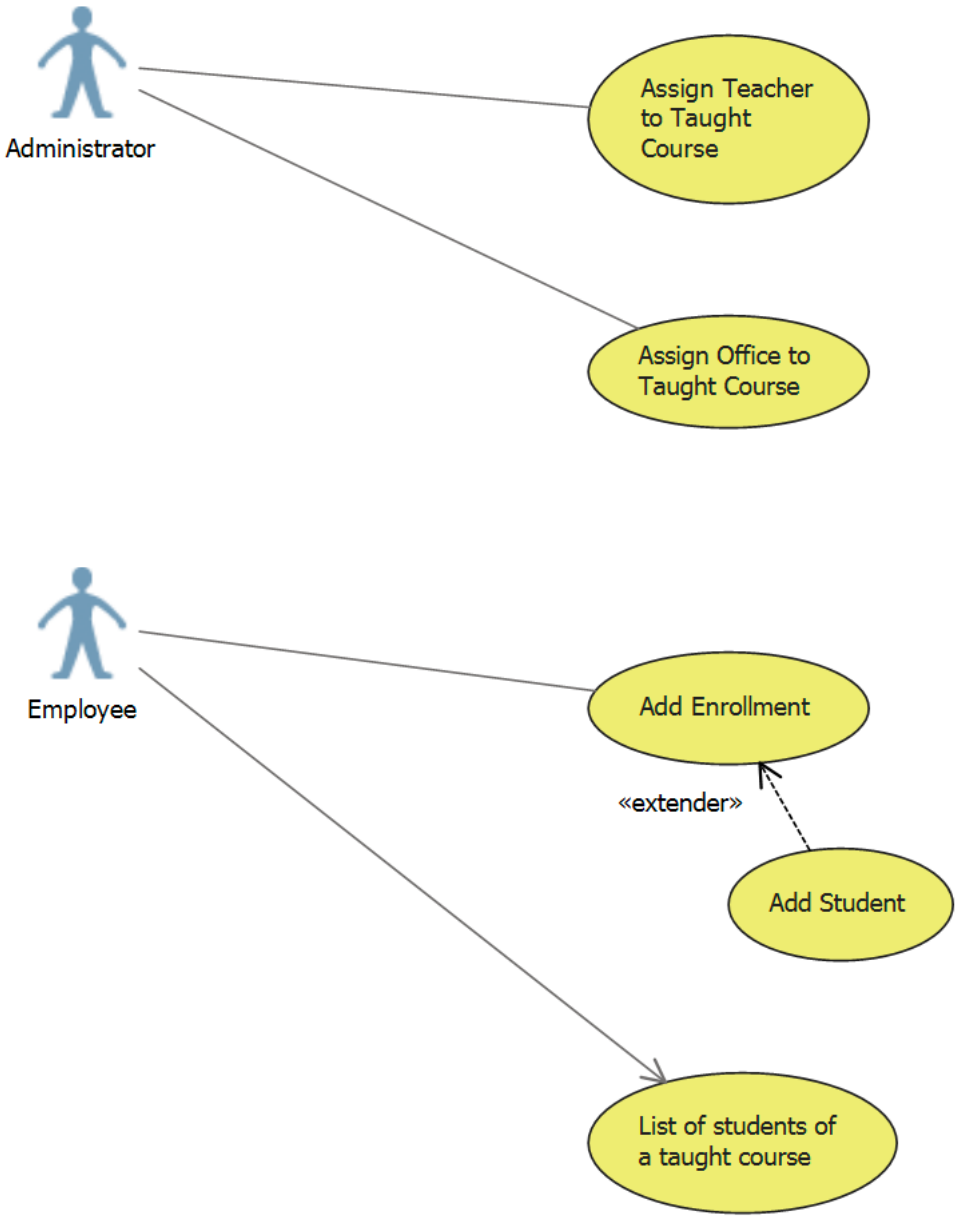

Figura 1. Fragment del Model de Casos d'Ús de GestAca

A continuació es detalla la descripció mitjançant una plantilla textual dels casos d'ús. Aquestes sessions només es demanen les capes de negoci i persistència. **No es demana la capa d'interfície gràfica**, que es farà per a les següent sessions.

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 a classroom to a taught course	2. The system shows all the existing 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 select 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	-	-

Detalls de la capa lògica

A les sessions anteriors de pràctiques se ha construït la part de la funcionalitat relacionada amb el disseny de les classes i el seu emmagatzemament persistent.

Per a aquestes sessions (5-6) **s'ha d'implementar el controlador (o proveïdor de serveis) de la lògica de negoci.** Aquest controlador haurà d'oferir a la capa superior (Interficie d'Usuari (IU)) accés a tota la funcionalitat que aquesta necessite.

El conjunt de tots els serveis oferits a la IU ha d'estar definit en una interfície anomenada, `IGestAcaService`. Els mètodes d'aquesta interfície s'implementaran en una classe anomenada `GestAcaService.` La interfície i la seua implementació han d'estar a la carpeta `BusinessLogic/Services` i formar part de l'espai de noms `GestAca.Services.` La classe `GestAcaService` farà ús dels serveis del DAL per a poder comunicar-se amb la capa de persistència.

Tots els errors que puguen aparèixer durant l'execució dels mètodes implementats s'hauran de reportar mitjançant l'ús d'excepcions. Per a reportar els errors generats en la capa lògica, a la carpeta `BusinessLogic/Services` haurà de crear-se una subclasse d'Exception anomenada `ServiceException`. Quan es creen objectes d'aquesta classe per informar d'algun error, la propietat `Message` d'aquesta classe contindrà el missatge de text amb l'errada generada.

Per exemple, en el cas d'estudi de referència que hi ha a PoliformaT anomenat *VehicleRental_ISW_2017,* quan es vol afegir una persona cal tindre en compte si ja existeix una amb el mateix DNI que la que es vol crear. A continuació es pot veure un fragment de codi de mètode corresponent, on abans d'afegir una persona es comprova que no existisca cap amb el mateix DNI. Si hi haguera alguna, s'envia una excepció.

```
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.");
}
```

Bones pràctiques d'escriptura de codi

És important utilitzar bones pràctiques d'escriptura de codi, tant en Enginyeria del Programari com en qualsevol altre context, ja que, entre altres coses, facilita el manteniment del codi i el fa més intel·ligible.

Per tant, el codi escrit per a implementar els casos d'ús definits en aquest butlletí ha de

seguir aquestes bones pràctiques, incloent la **delegació** de codi de la classe de serveis a les altres classes de la capa de lògica, o la **reutilització** de codi quan siga possible. Aquest serà un criteri tingut en compte a l'hora de valorar el projecte final.

Data de lliurament

La data de lliurament serà la corresponent a la setmana 9 de pràctiques (l'última sessió) al laboratori[1]. **TOTS els membres del grup han d'acudir al lliurament** ja que poden ser requerits per a respondre preguntes sobre el projecte.

[1] Cal tindre en compte que el lliurament final ha de contenir les interfícies d'usuari i que aquestes s'implementen durant les últimes dues sessions de laboratori (sessions 7 i 8).