ISW – PRÀCTIQUES

SESSIO 5

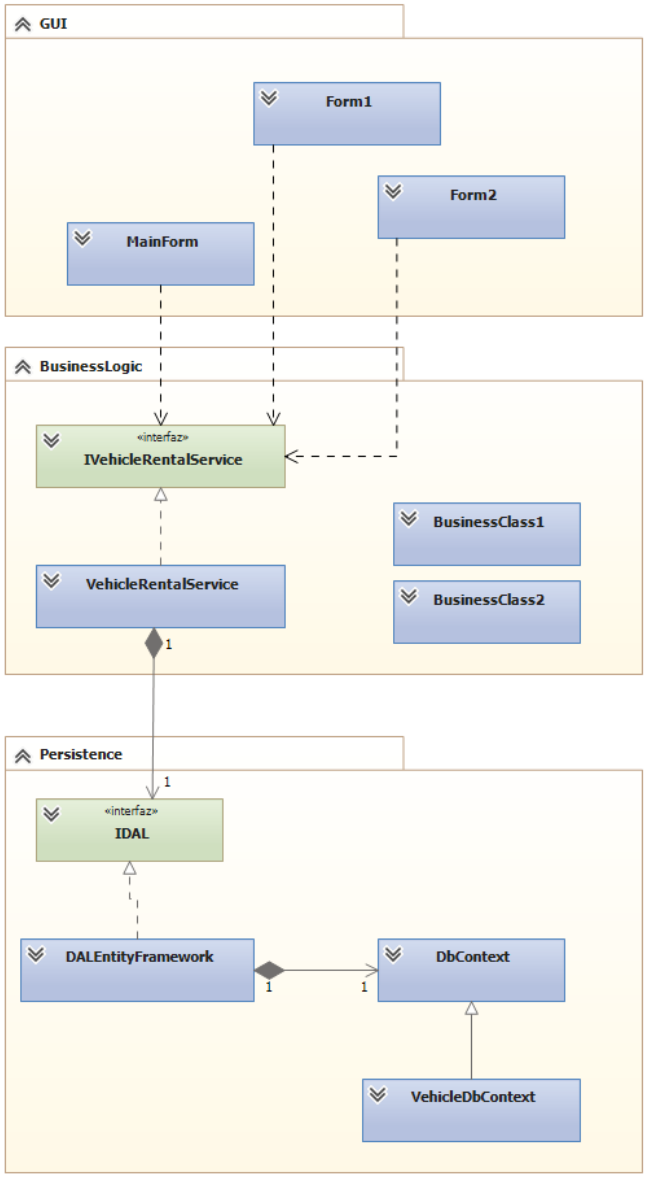
Casos d'ús– material d'ajuda

ETS Enginyeria Informàtica
DSIC – UPV

Curs 2024-2025

Arquitectura 3 capes tancada

- Seguim una arquitectura multi-capa amb:
  - Presentació (IGU)
  - Lògica de negoci
  - Persistència
- Cas d'estudi: ***VehicleRentalService***



Capa Lògica
GestAcaService

```
namespace GestAca.Services
{
    public class GestAcaService: IGestAcaService
    {
        private readonly IDAL dal;

        public GestAcaService (IDAL dal)
        {
            this.dal = dal;
        }

        public void RemoveAllData()...

        public void Commit()...

        public void DBInitialization()
```

// Un atribut que referencia al objecte DAL que se instanciarà des de el programa principal

// Si necessitem altres atributs que depenent del cas d'estudi .....

// Implementació de mètodes auxiliars per a inicialitzar la BD (amb un conjunt de dades)

// Implementació del serveis que ofereix la capa lògica, i que seran invocats des-de els formularis de la capa de presentació

ServiceException.cs

```
using System;

namespace ServiceException.Services

{
    public class ServiceException : Exception
    {
        public ServiceException()
        {
        }

        public ServiceException(string message)
        : base(message)
        {
        }

        public ServiceException(string message, Exception inner)
        : base(message, inner)
        {
        }
    }

}
```

// Tractament d'excepcions

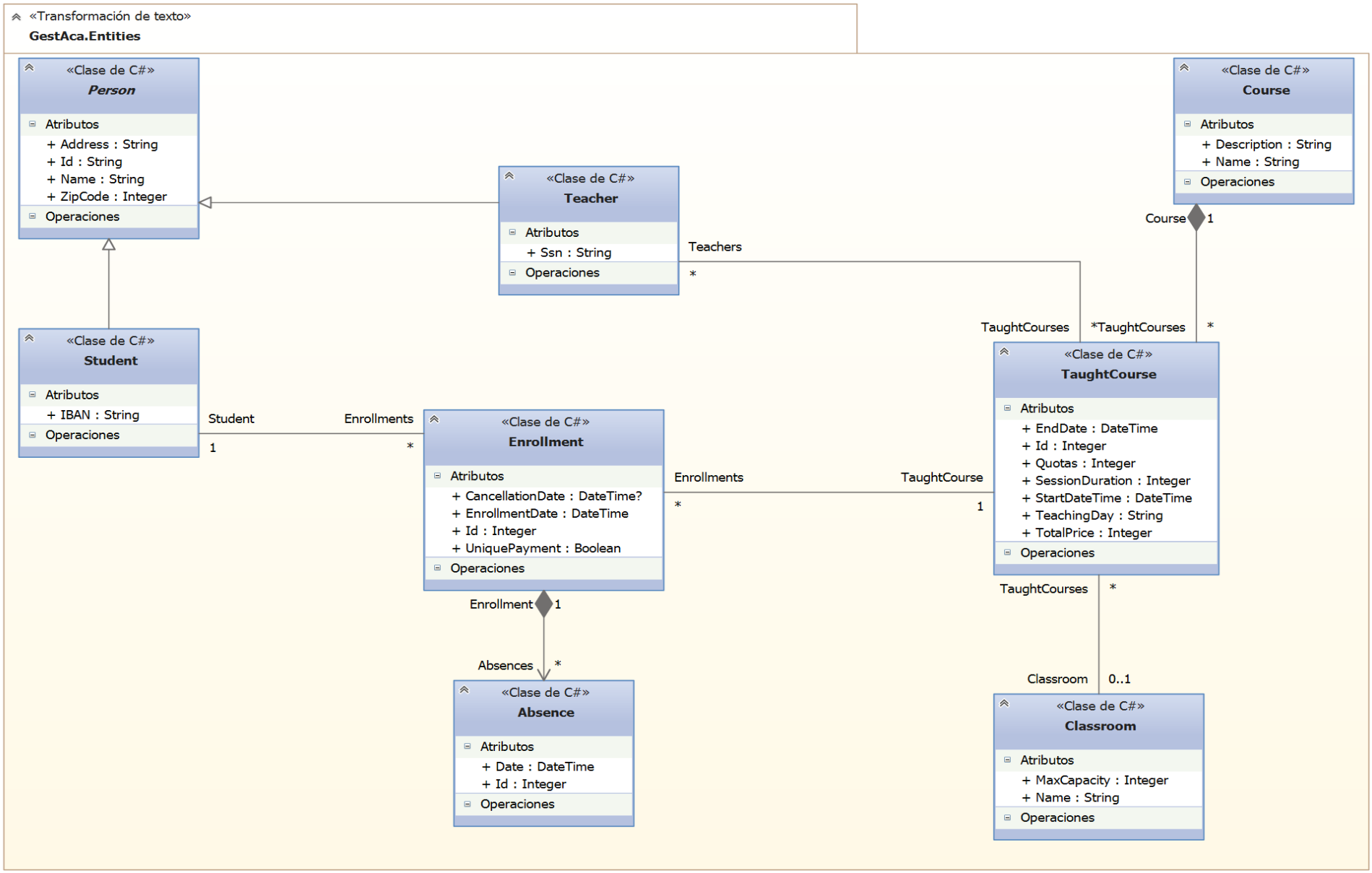
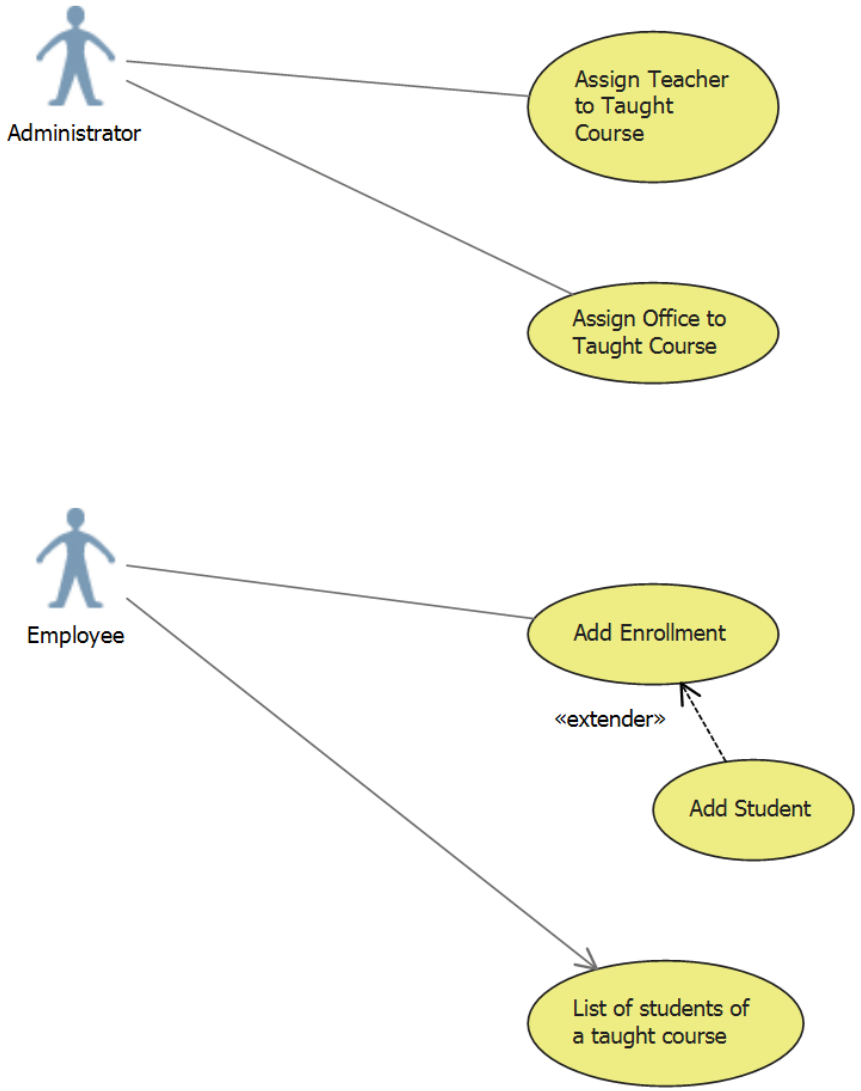


Diagrama de Classes de Disseny de **GestAca**

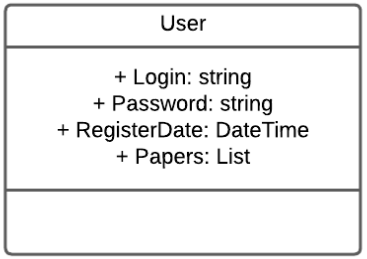
Diagrama de casos d'ús



Exemple:

- Identificar els serveis/mètodes necessaris per a implementar els casos d'ús

ID	XXX	
Use case	Register User	
Actors	User	
Objective	Registration of a new user	
Abstract	The user provides a username and a password. The register date is the current data and the list of paper is empty.	
Precondition	-	
Postcondition	The user is registered in the system	
Includes		
Extends		
Inherits from		
Steps	User intentions	System obligations
	1. The user chooses the Register option in the system.	2. The system asks for the registration information.
	3. The user provides the required information: username, password.	4. The system checks that the information is valid and stores the new user into the system.
Synchronous extensions		xxxx
Asynchronous extensions	-	

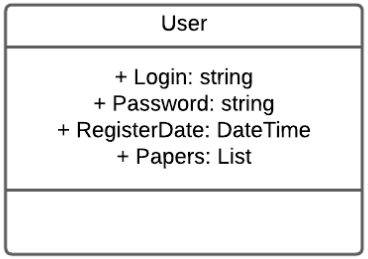


- Servei : `?? RegisterUser (????);`

Exemple:

- Identificar els serveis/mètodes necessaris per a implementar els casos d'ús

- Servei : `?? RegisterUser (????);`



➡ **Comunicació entre capes: Presentació-Lògica**

** S'intercanvien únicament les dades necessàries

- Servei : `?? RegisterUser (string login, string pass);`

** S'intercanvien únicament les dades necessàries encapsulades en un objete DTO

- Servei : `?? RegisterUser (UserDTO userDTO);`

** S'intercanvien objectes de la lògica de negoci

- Servei : `?? RegisterUser(User user);`

Implementació de casos d'ús

- **Identificar els serveis/mètodes necessaris per a implementar els casos d'ús.**

* Punt de partida: Diagrama de Casos d'Us i les plantilles de descripció.

- Consideracions:
  - Treballem amb un llenguatge orientat a objectes (reutilització i encapsulació).
  - Les operacions pròpies de les classes (per exemple, comprovar la igualtat) **han d'implementar-se a les classes, i no als serveis**. Així com accedir a les **col·leccions mitjançant els objectes de la capa de lògica** de negoci que mantenen aquestes col·leccions *(accedir mitjançant el DAL sols quan siga estrictament necessari)*

Exemples:

- Programació OO

1. Calcular si una persona es major d'edat

```
class Program
{
    static void Main(string[] args)
    {
        Persona p = new Persona(DateTime.Parse("12/10/2010"));
        DateTime hoy = DateTime.Now;
        DateTime fn = p.Fecha_nacimiento;

        if ((hoy.Year - fn.Year) < 18) Console.WriteLine("No es mayor de edad");
        else if (((hoy.Year - fn.Year) == 18) & (hoy.DayOfYear < fn.DayOfYear))
            Console.WriteLine("No es mayor de edad");
        else Console.WriteLine("Es mayor de edad.");
        Console.ReadKey();
    }
}
```

Métode de la classe Persona

Exemples:

- Programació OO

```
class Program
{
    static void Main(string[] args)
    {
        Persona p = new Persona(DateTime.Parse("12/10/2010"));

        if (p.MayorDeEdad())
            Console.WriteLine("Es mayor de edad");
        else Console.WriteLine("No es mayor de edad.");
        Console.ReadKey();
    }
}
```

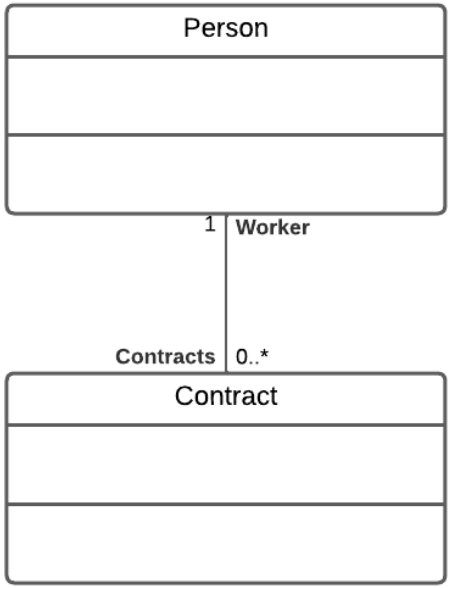
```
class Persona
{
    public DateTime Fecha_nacimiento { get; set; }

    public bool MayorDeEdad()
    {
        DateTime hoy = DateTime.Now;
        if ((hoy.Year - Fecha_nacimiento.Year) < 18) return false;
        else if (((hoy.Year - Fecha_nacimiento.Year) == 18) & (hoy.DayOfYear < Fecha_nacimiento.DayOfYear))
            return false;
        else return true;
    }
```

Exemples:

- Programació OO

2. Afegir un contracte a una persona



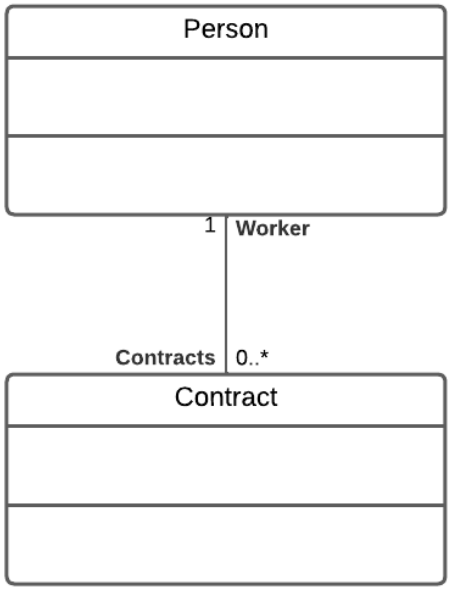

```
Person p = dal.GetById<Person>("12345678X");
…
Contract c = new Contract(…);
…
p.Contracts.Add(c);
```

```
p.AddContract(c);
```

Exemples:

- Programació OO

3. Obtindre els contractes d'una persona

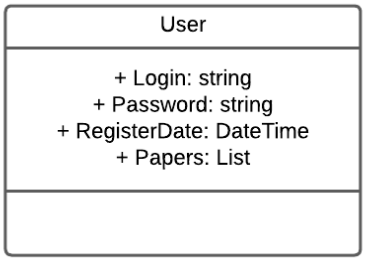



```
Person p = dal.GetById<Person>("12345678X");
...
List<Contract> contracts = p.GetContracts();
...
```

Exemple:

- Identificar els serveis/mètodes necessaris per a implementar els casos d'ús

ID	XXX	
Use case	Register User	
Actors	User	
Objective	Registration of a new user	
Abstract	The user provides a username and a password. The register date is the current data and the list of paper is empty.	
Precondition	-	
Postcondition	The user is registered in the system	
Includes		
Extends		
Inherits from		
Steps	User intentions	System obligations
	1. The user chooses the Register option in the system.	2. The system asks for the registration information.
	3. The user provides the required information: username, password.	4. The system checks that the information is valid and stores the new user into the system.
Synchronous extensions		xxxx
Asynchronous extensions	-	

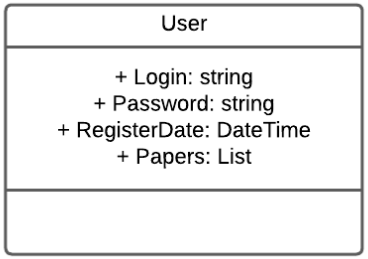


- Servei : `?? RegisterUser (????);`

Exemple:

- Identificar els serveis/mètodes necessaris per a implementar els casos d'ús

- Servei : `?? RegisterUser (????);`



➡ Comunicació entre capes: Presentació-Lògica

** S'intercanvien únicament les dades necessàries

- Servei : `?? RegisterUser (string login, string pass);`

** S'intercanvien únicament les dades necessàries encapsulades en un objete DTO

- Servei : `?? RegisterUser (UserDTO userDTO);`

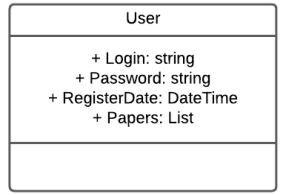
** S'intercanvien objectes de la lògica de negoci

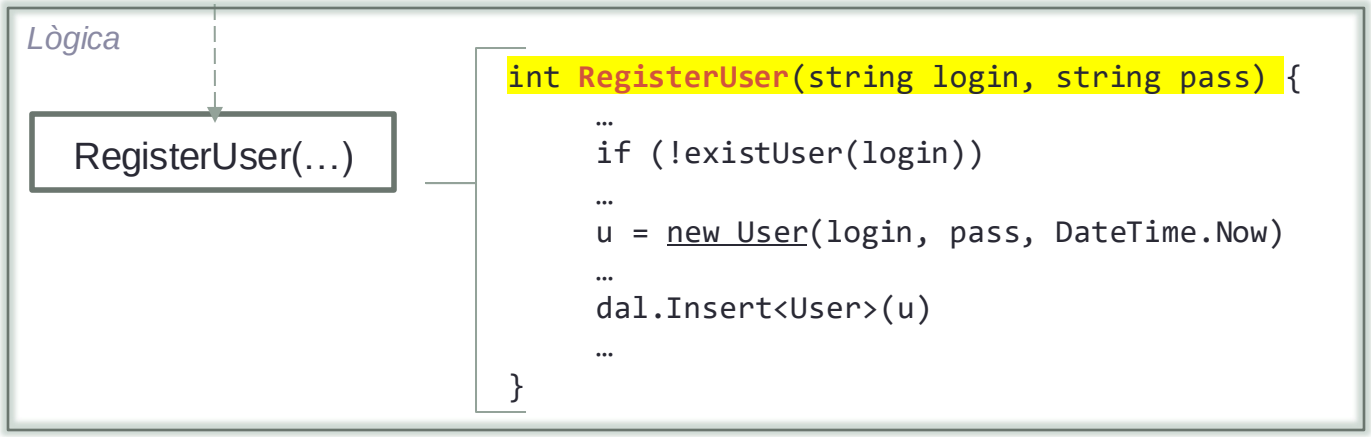
- Servei : `?? RegisterUser(User user);`

Comunicació entre capes: Presentació-Lògica

** S'intercanvien únicament les dades necessàries

- Servei : `?? RegisterUser(string login, string pass);`





S'intercanvien únicament les dades necessàries

Tipus bàsics

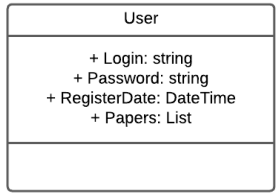
Acoblament mínim
Màxima cohesió: cada capa es responsabilitza de les seues funcions
La capa de presentació no coneix objectes de la lògica de negoci

Màxima independència entre capes

Comunicació entre capes: Presentació-Lògica

** S'intercanvien objectes de la Lògica

- Servei : `?? RegisterUser (string login, string pass);`



S'intercanvien objectes de la lògica de negoci

Acoblament màxim
Menor cohesió: la capa de presentació realitza funcions de la capa lògica
La capa de presentació coneix i manipula objectes de la lògica de negoci

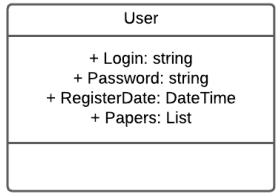
Menys codi a implementar

Mínima independència

Comunicació entre capes: Presentació-Lògica

** S'intercanvien únicament les dades necessàries

- Servei : `?? RegisterUser (UserDTO userDTO);`



Presentació

RegisterUserForm

```
void RegisterUserForm() {string login, pass;
…
login=…
pass=…
…
RegisterUser(new UserDTO(login, pass));
…
}
```

Lògica

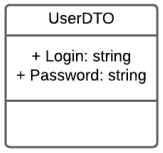
RegisterUser(…)

```
int RegisterUser(UserDTO userDTO) {
      …
      if (!existUser(userDTO.login))
      …
      u = new User(UserDTO.login, userDTO.pass,
                   DateTime.Now);
      …
      dal.Insert<User>(u);
      …
}
```

S'intercanvien les dades necessàries encapsulades en un objecte DTO

Acoblament major
Màxima cohesió: cada capa es responsabilitza de les seues funcions
La capa de presentació no coneix objectes de la lògica de negoci

Útil per a reduir el nombre de paràmetres
Requereix implementar els DTOs necessaris per als serveis



Menys independència

GestAcaService

```
/// <summary>
    /// Persiste un profesor
    /// </summary>
    /// <param name="teacher"></param>
    /// <exception cref="ServiceException"></exception>

    public void AddTeacher(Teacher teacher)
    {
      // Restricción: No puede haber dos personas con el mismo Id (dni)

      if (dal.GetById<Teacher>(teacher.Id) == null)
      {
        dal.Insert<Teacher>(teacher);
        dal.Commit();
      }
      else
        throw new ServiceException
          ("There is another person with Id " + teacher.Id);
    }
```

```
/// <summary>
    /// Persiste un profesor
    /// </summary>
    /// <param name="teacher"></param>
    /// <exception cref="ServiceException"></exception>

    public void AddTeacher(string address, string id, string name, int zipcode, string ssn)
    {
      // Restricción: No puede haber dos personas con el mismo Id (dni)

      if (dal.GetById<Teacher>(id) == null)
      {
        teacher = new Teacher (address, id, name, zipcode, ssn);
        dal.Insert<Teacher>(teacher);
        dal.Commit();
      }
      else
        throw new ServiceException
              ("There is another person with Id " + teacher.Id);
    }
```