

Classe Problemes Setmana 6: GRASP, inicialitzacions i DAO

Anna Puig

Enginyeria Informàtica

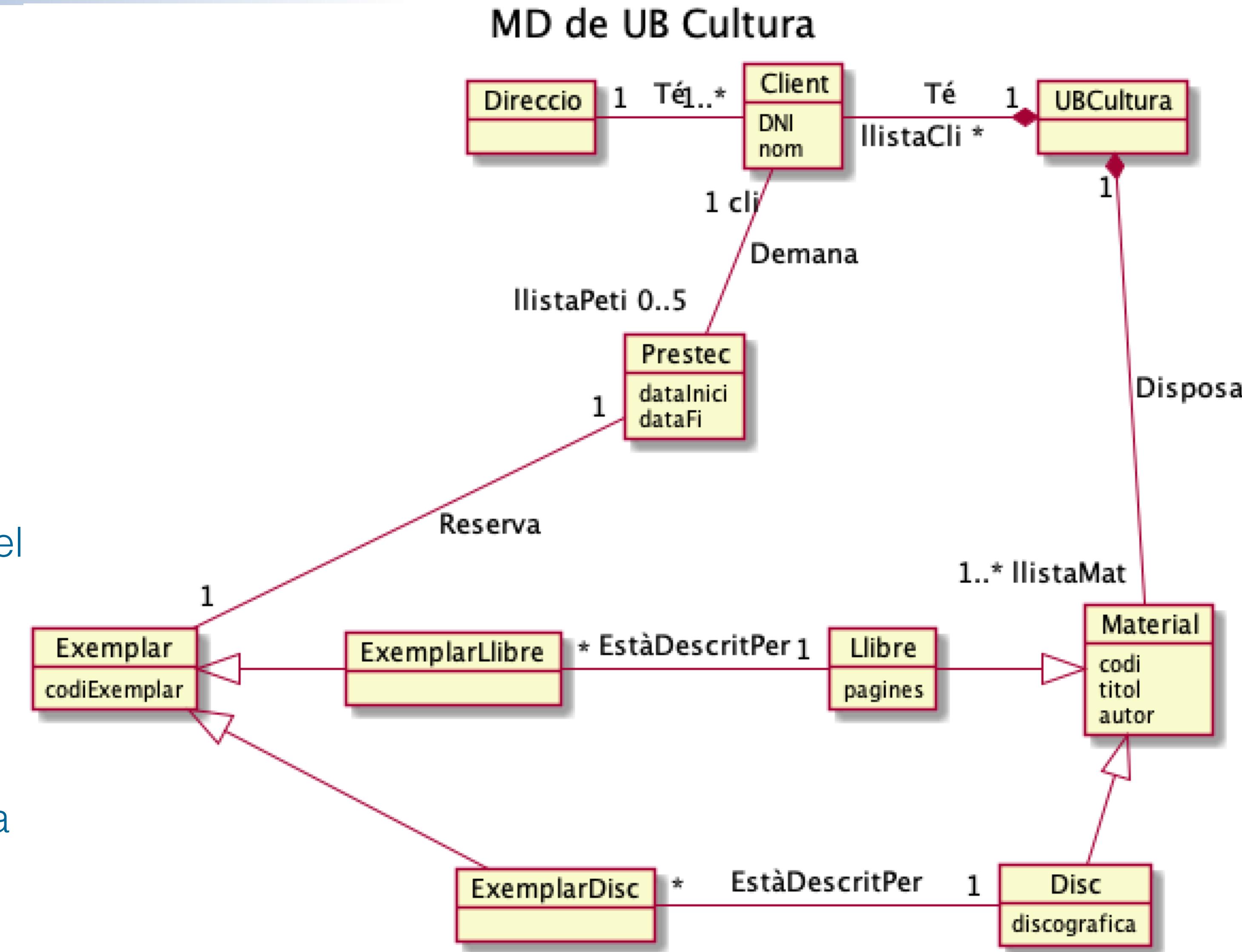
Facultat de Matemàtiques i Informàtica,
Universitat de Barcelona

Curs 2021/22

Exercici per pensar

Objectiu: Partint del Model de Domini, quin Diagrama de Classes obtens després de dissenyar/implementar les següents funcionalitats de test? Detalla els 4 Diagrames de Classes que vas obtenint... Quins patrons GRASP estàs usant?

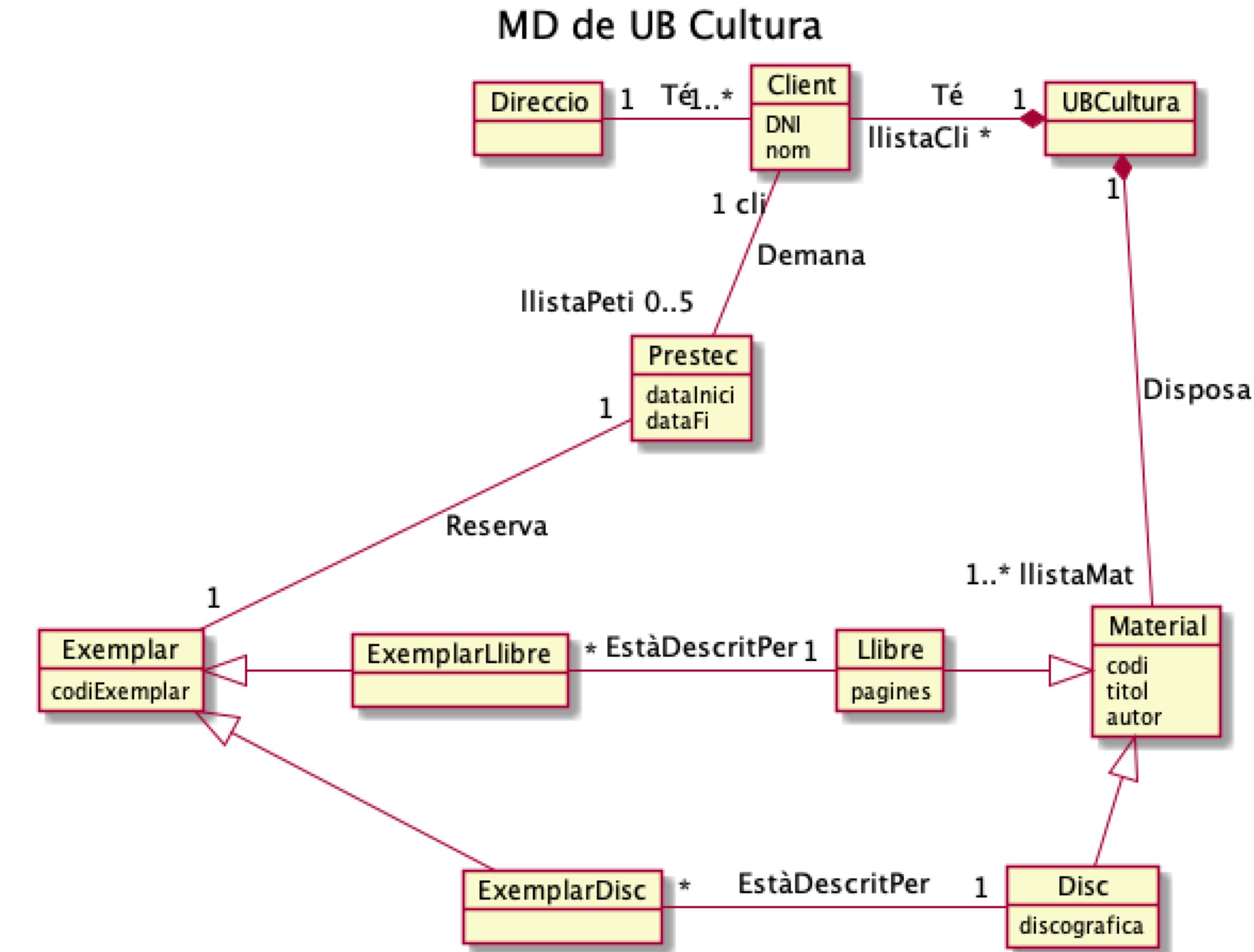
1. BuscarClient(DNI) retorna el nom del client
2. Afegir un préstec a un client, donat el DNI del client
3. Llistar tot el material en préstec d'UBCultura en un cert dia
4. Ara es volen també tenir vendes i no només préstecs, com canviaria això el teu diagrama de classes?



Exercici per pensar

Objectiu: Abans de començar, determinar:

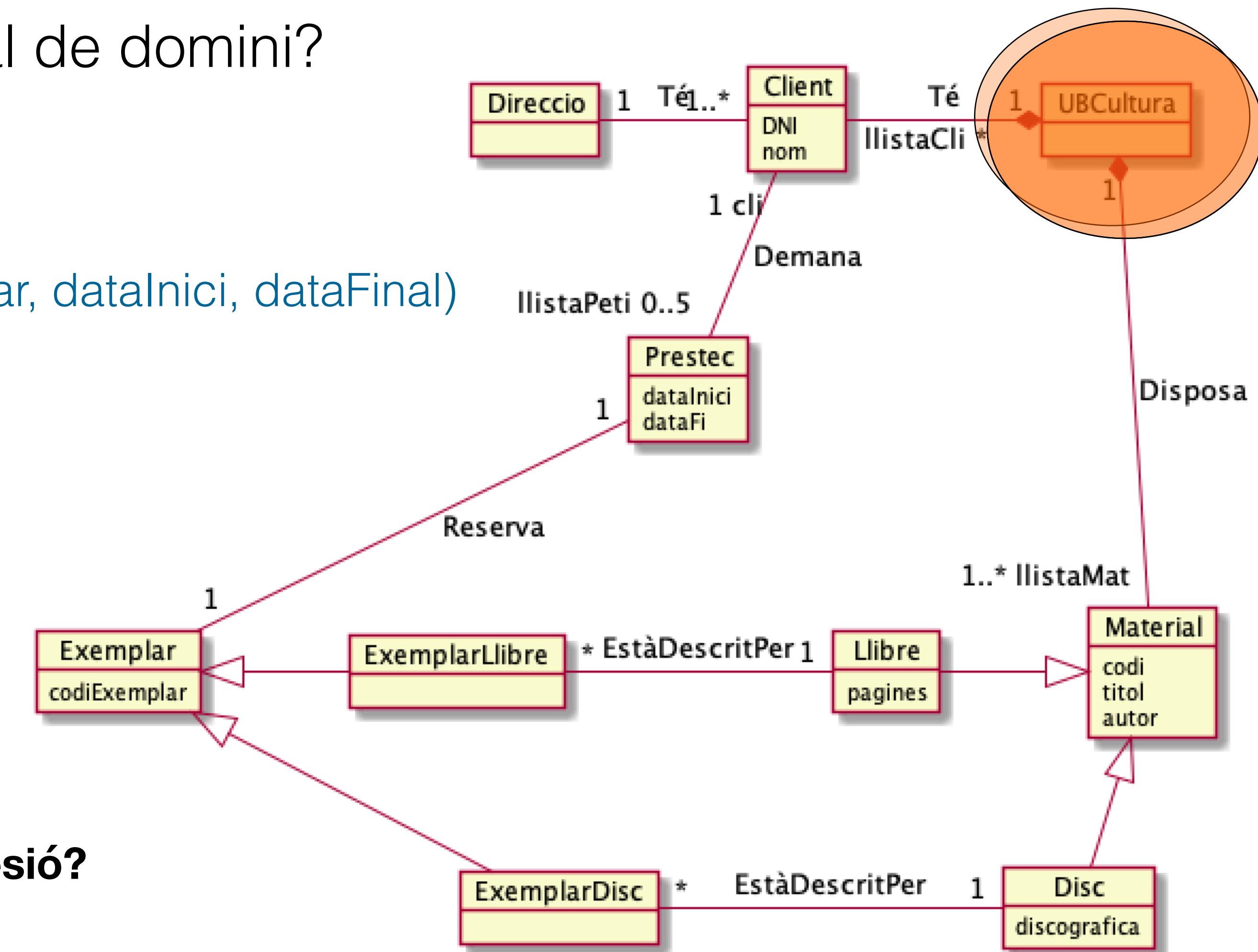
1. Quan el controlador es comunica amb el model, hi ha un **objecte principal** del Model de Domini que controli l'accés?
 - si **existeix**, quins són els candidats?
 - si **no existeix**, com desacoblo en Controlador de moltes classes del domini?
2. Qui **inicialitza les dades**?



Problema de delegació de crides usant el Model de Domini

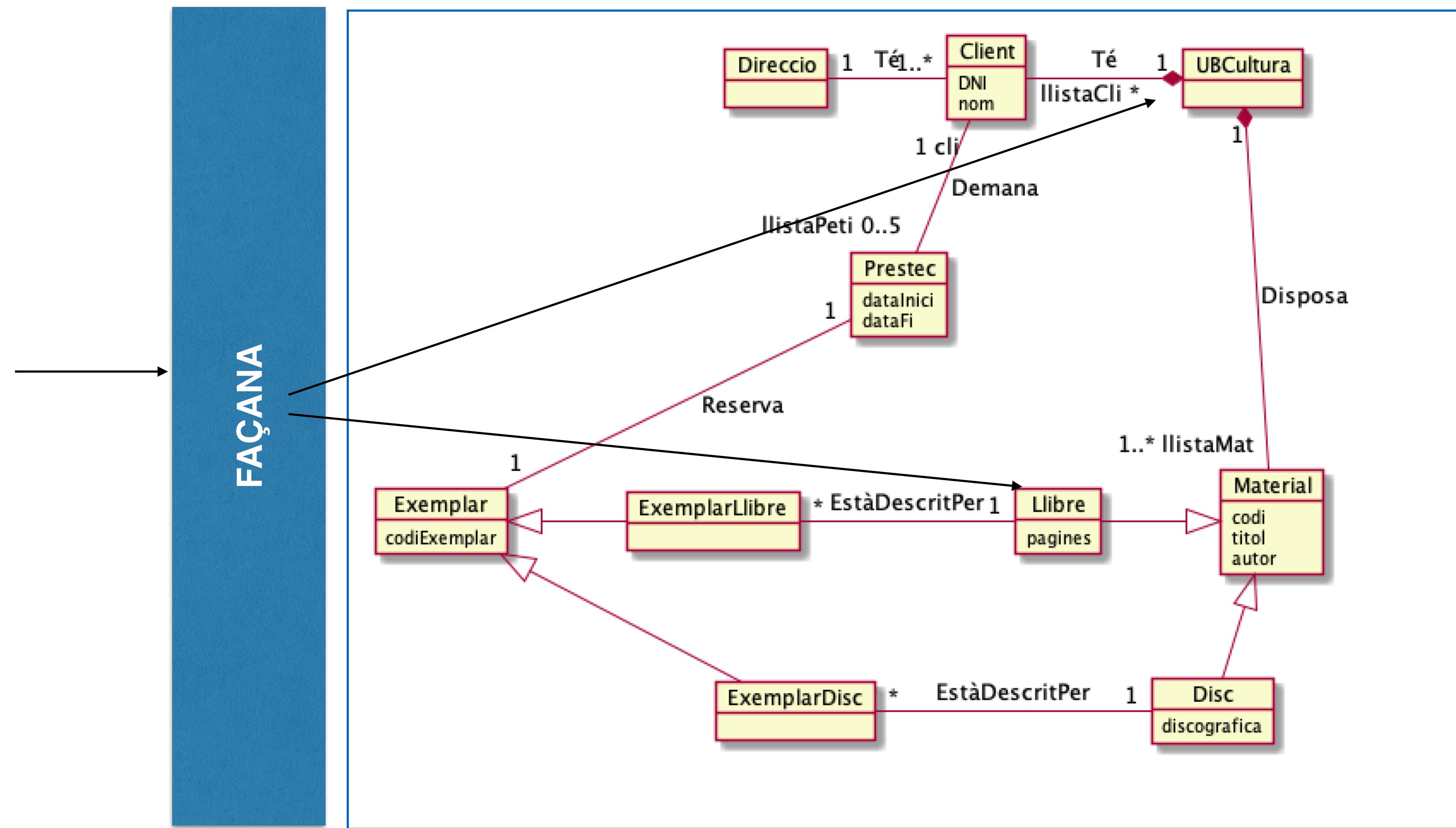
- Qui és l'objecte inicial de domini?

1. BuscarClient(DNI)
2. AfegirPréstec(DNI, codiExemplar, dataInici, dataFinal)
3. CercarNúmeroExemplars(títol)



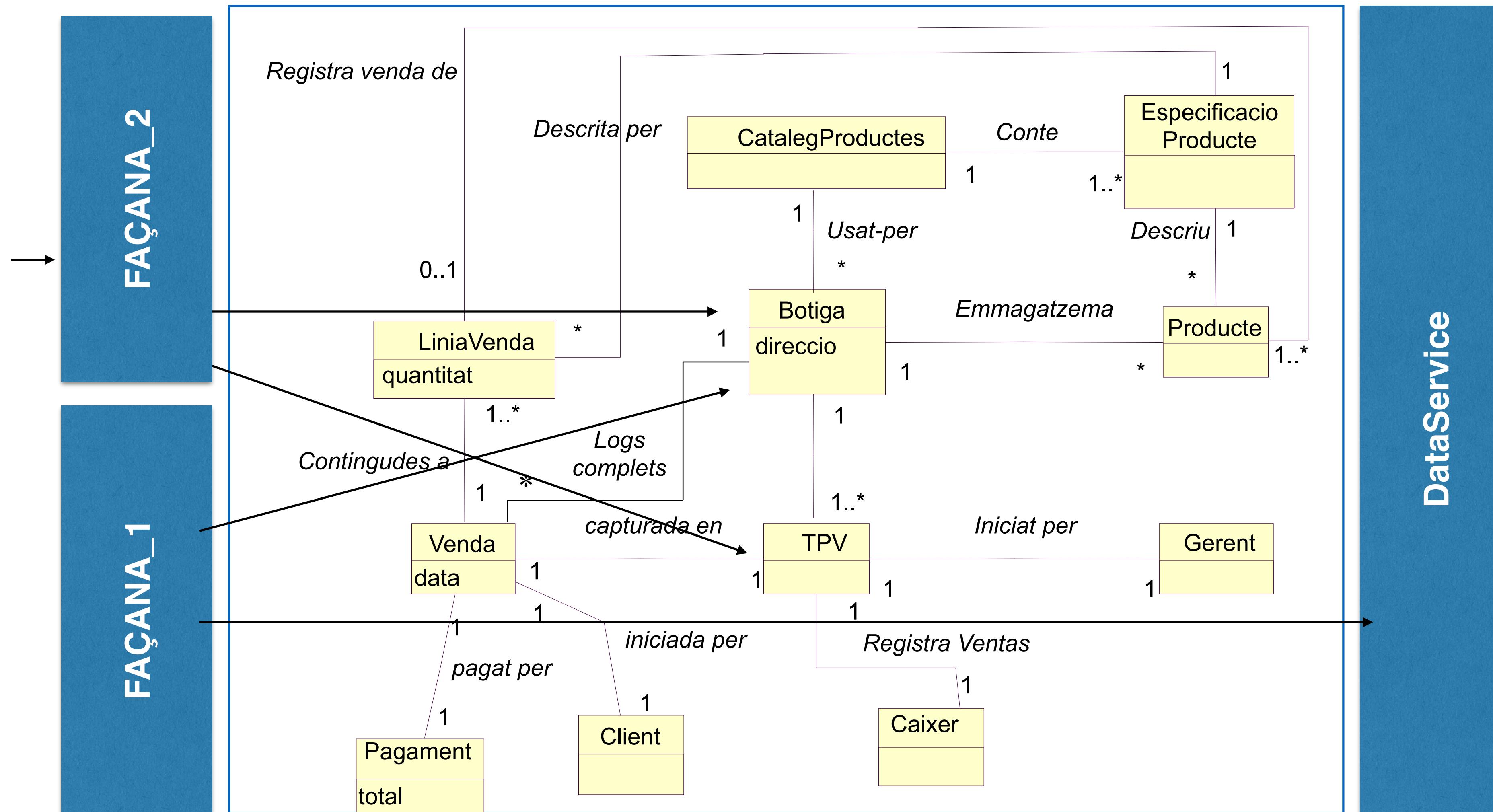
Problema de delegació de crides usant el Model de Domini

- Qui és l'objecte inicial de domini?



Problema de delegació de crides usant el Model de Domini

- Poden haver més d'una façana?



Problema d'iniciar les dades de l'aplicació

Inicialització de l'aplicació: Si es segueix un Model-Vista-Controlador:

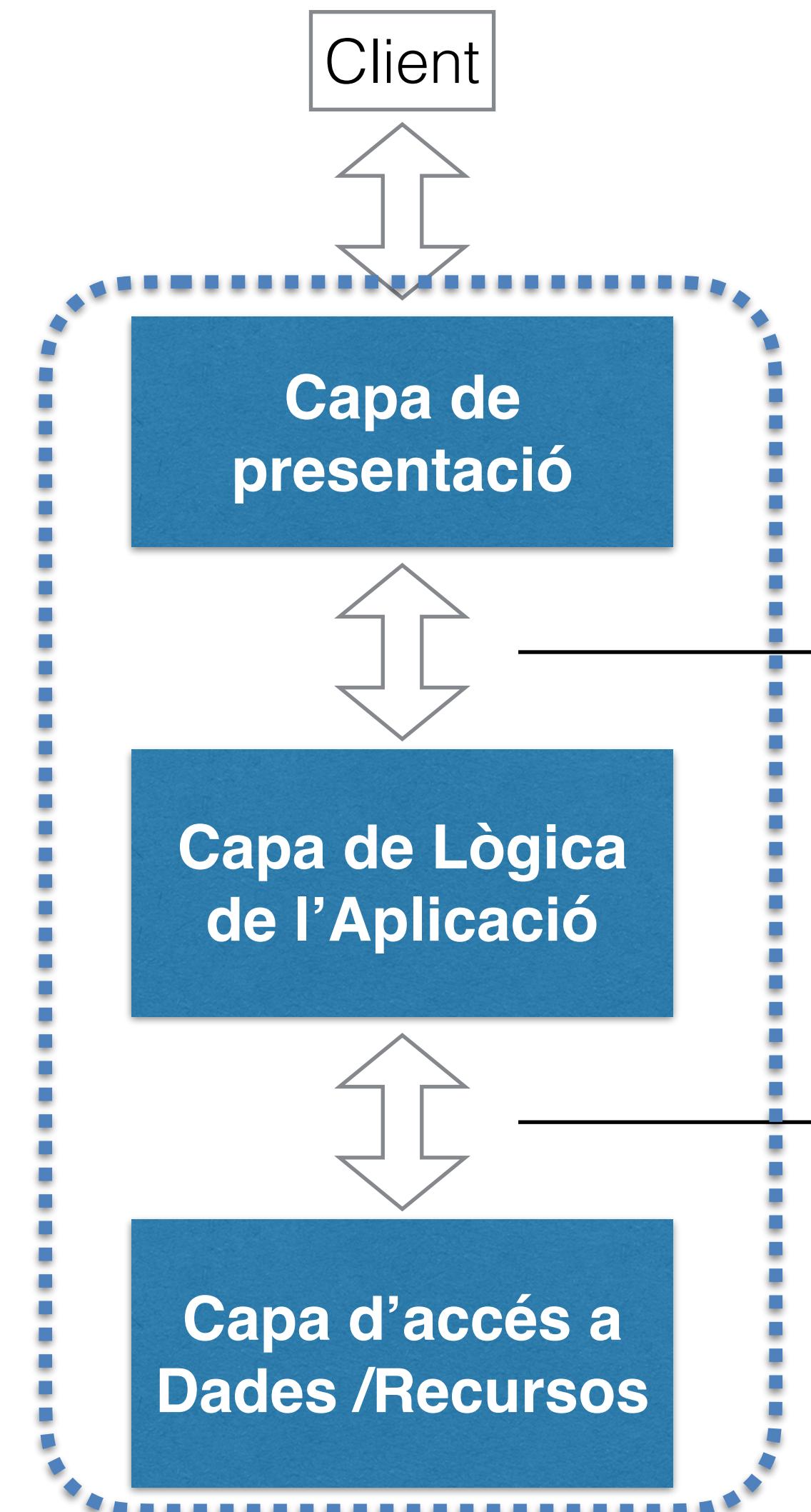
- Qui té la responsabilitat d'iniciar la creació les dades?
 - **Possibilitat 1:** L'objecte **domini** que contingui més agregacions o que sigui l'arrel de l'aplicació, i invocar un mètode *create* d'aquell objecte que serà qui delegarà la creació de les instàncies de la resta d'objectes.
 - **Possibilitat 2:** L'objecte **controlador**
 - **Possibilitat 3:** L'objecte **façana** corresponent.
 - **Possibilitat 4:** La capa de **Persistència**

NOTA:

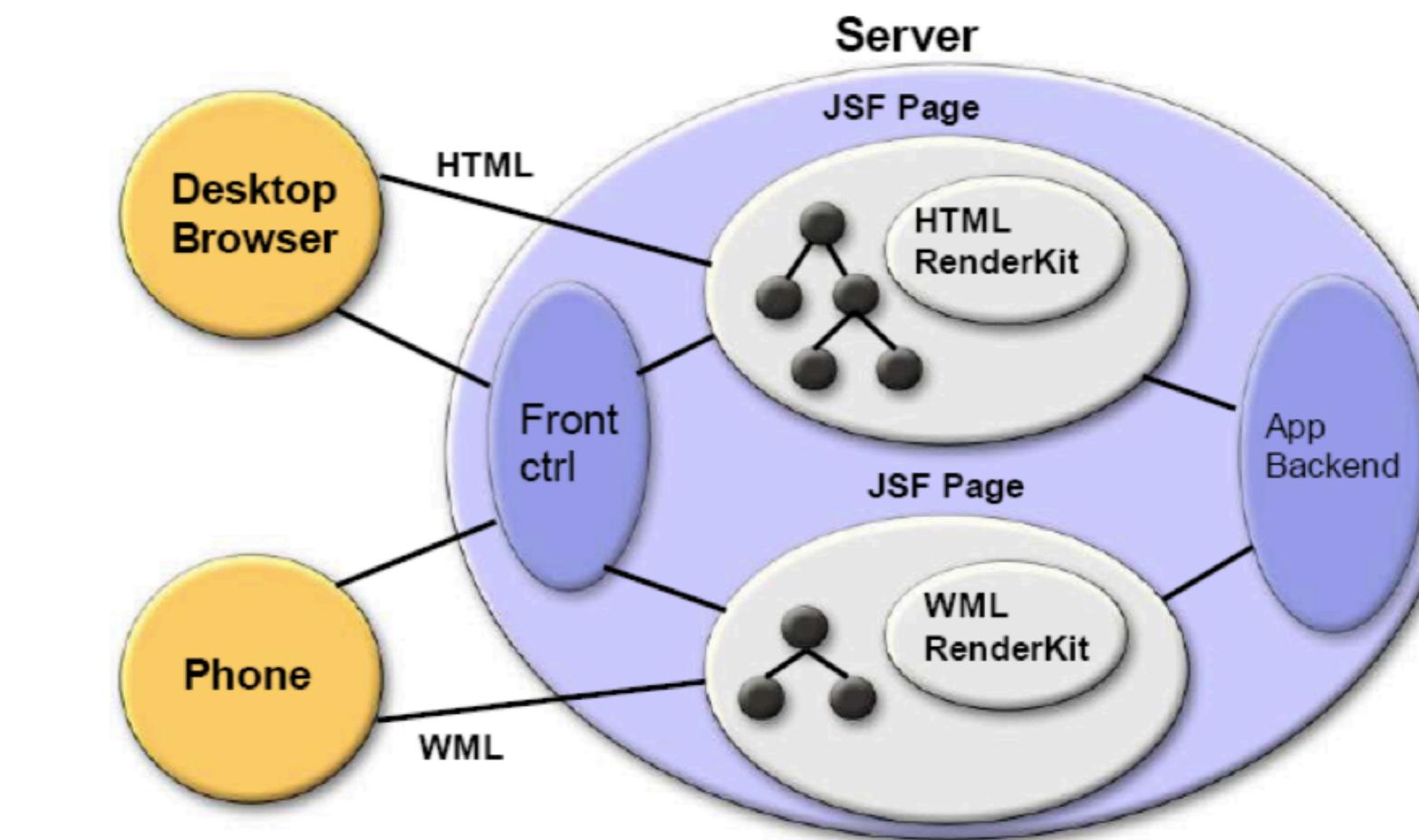
- Si s'usen dades persistents (Bases de Dades, fitxers,...) potser el controlador serà l'encarregat de delegar a algun servei la càrrega de la Base de Dades o d'un fitxer (usant el patró **DAO**)

3.2. Patrons arquitectònics

Patró per capes:



Patró Model-Vista-
Controlador



Patró Data Access
Object



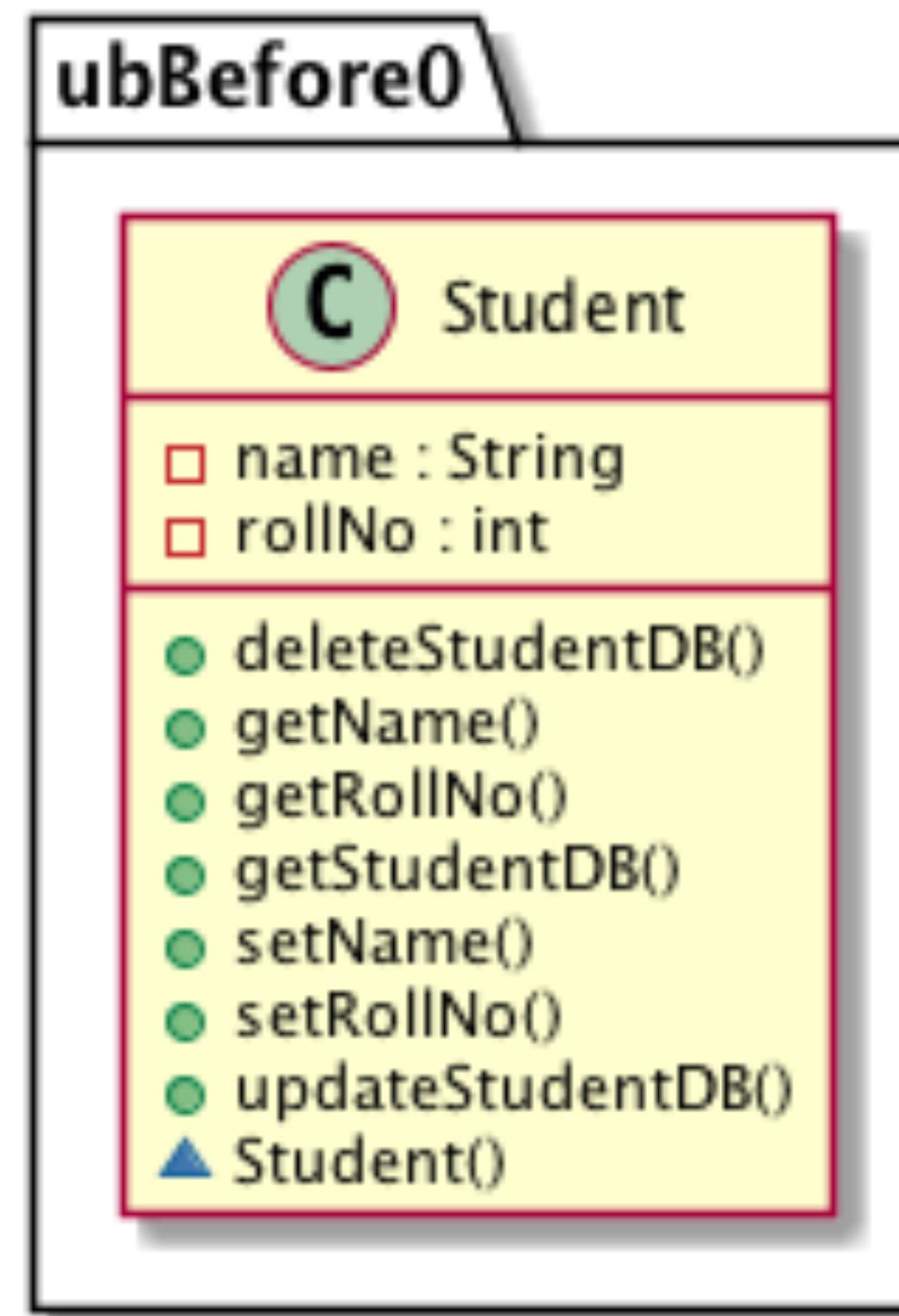
HIBERNATE



JDBC Templates

3.2. Patrons arquitectònics

Comunicació entre la capa de lògica i la capa de persistència:



Cohesió?
Acoblament?

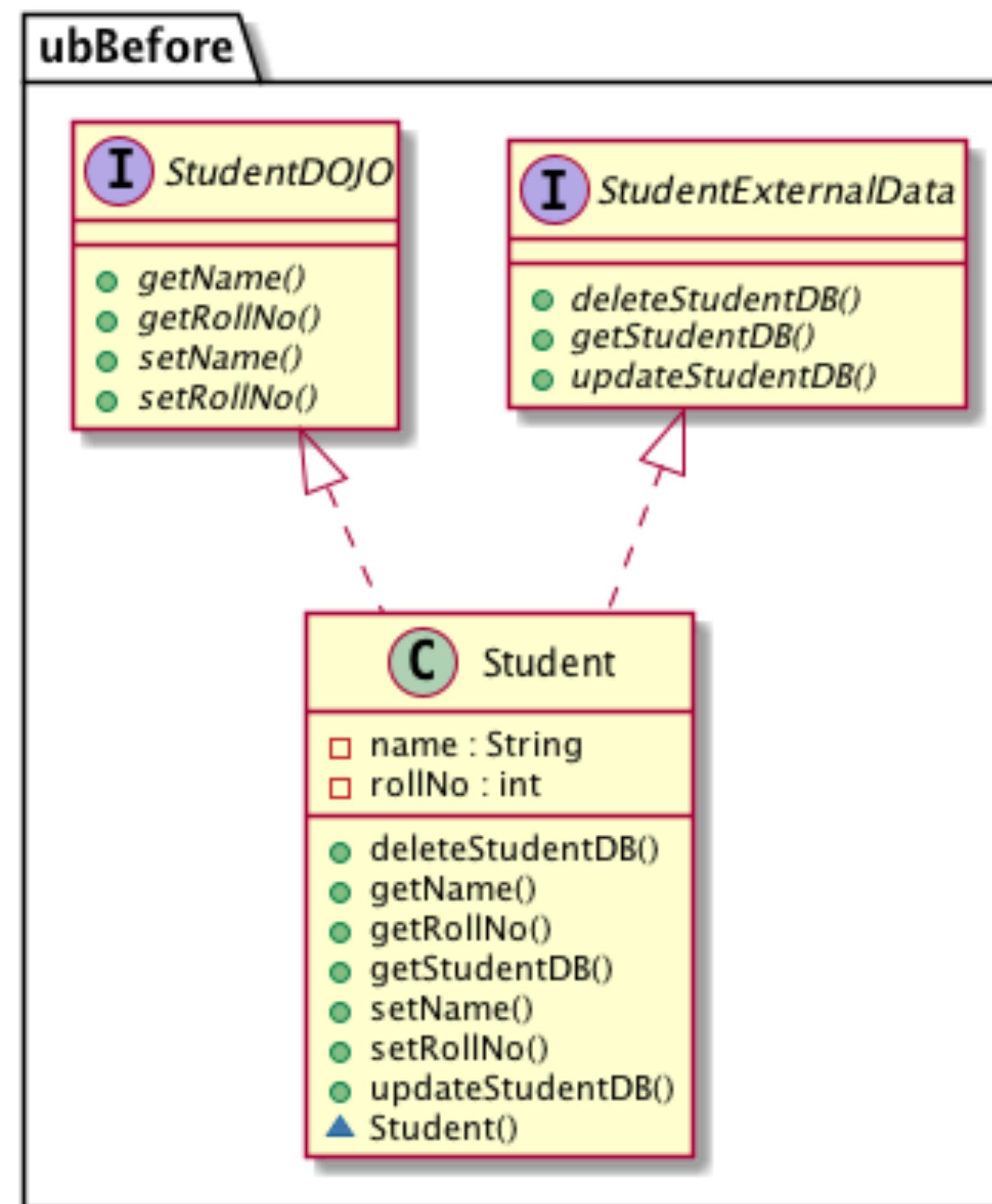
Inmobilitat

Viscositat

Rigidesa

3.2. Patrons arquitectònics

Comunicació entre la capa de lògica i la capa de persistència:



Cohesió?
Acoblament?

3.2. Patrons arquitectònics

Nom del patró: **Patró Data Access Object (DAO)**

Problema

- Accés als components de baix nivell acoblat a la lògica de l'aplicació
- Poc flexible a canvis de components de baix nivell (rigidesa)
- Viscositat a nivell de disseny de l'aplicació

Solució

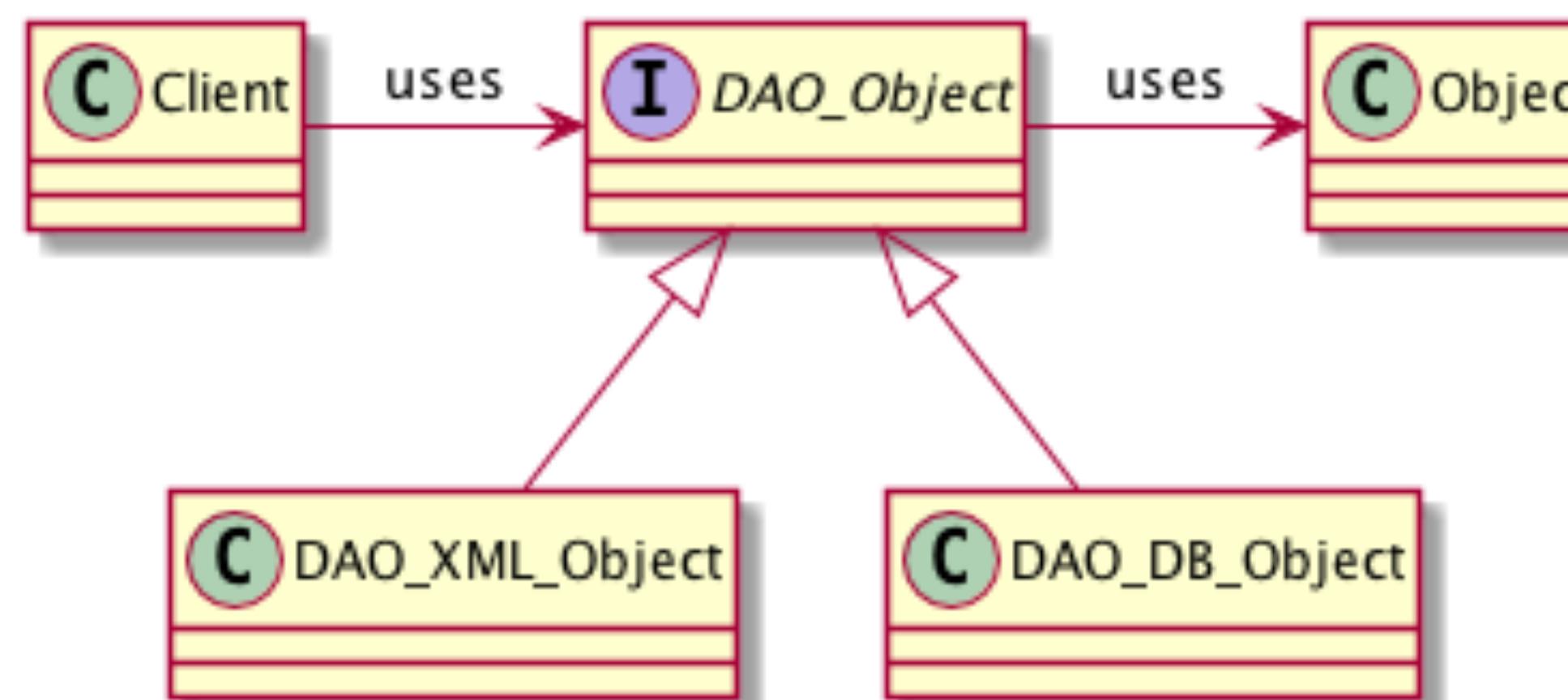
- **Aspecte estàtic:** Tot accés a Bases de Dades es fa mitjançant DAO. Encapsula les operacions CRUD dels objectes de l'aplicació
- **Aspecte dinàmic:** col·laboració i desacoblamet de la capa d'aplicació a la capa de persistència

3.2. Patrons arquitectònics

Patró Data Access Object: (DAO)

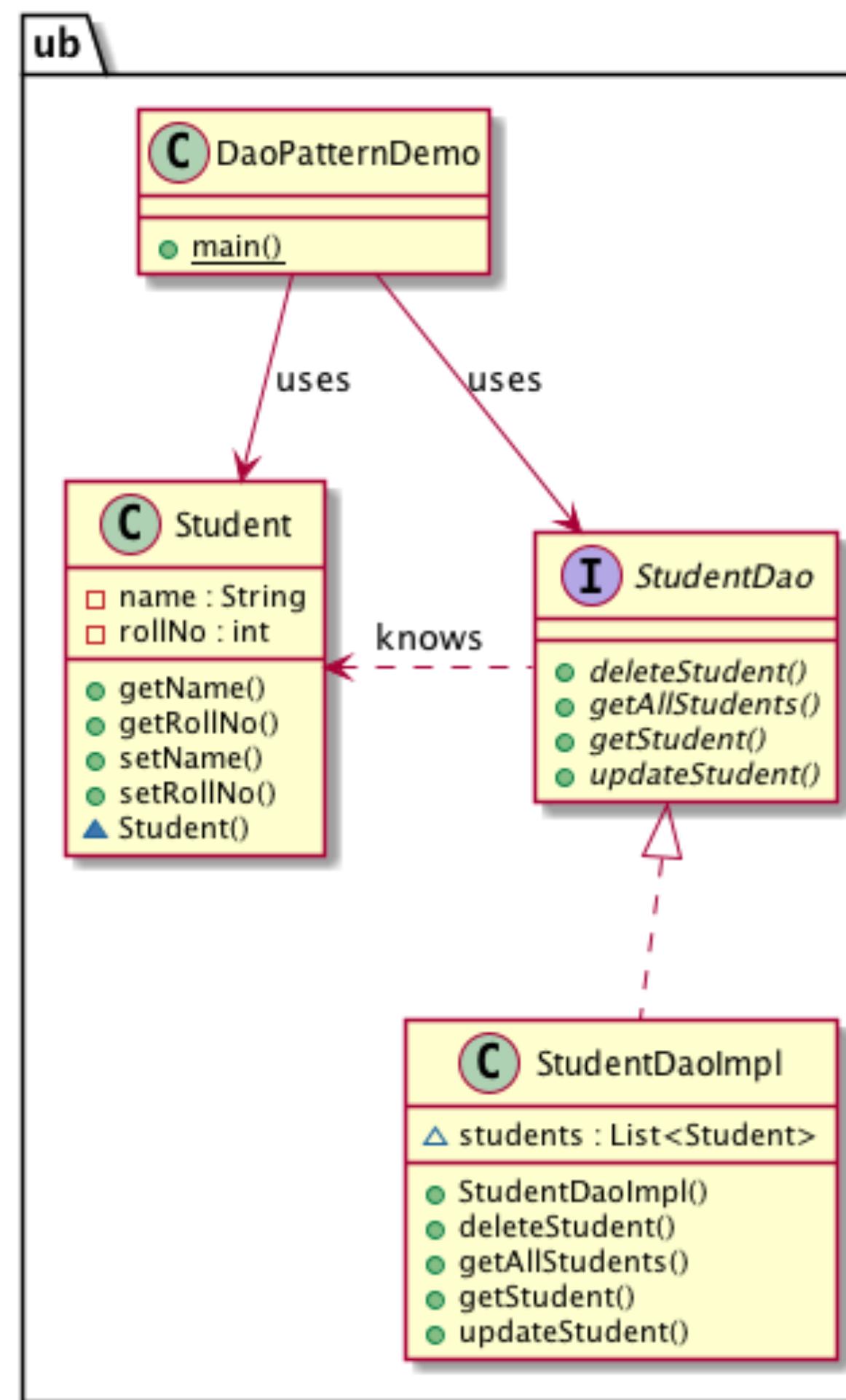
Utilitzat per separar l'accés a dades o recursos a baix nivell des de les funcionalitats. Els participants són:

- **Data Access Object Interface** - Defineix les operacions stàndard que s'han de fer en els objectes del model
- **Data Access Object classe d'implementació** - Realitza la implementació concreta sobre la base de dades /XML o la tecnologia concreta de persistència.
- **Object**: (POJO: Plain Old Java Object) - Objecte que conté les dades extretes de la classe DAO, i permet mètodes de get/set.



3.2. Patrons arquitectònics

Comunicació entre la capa de lògica i la capa de persistència:



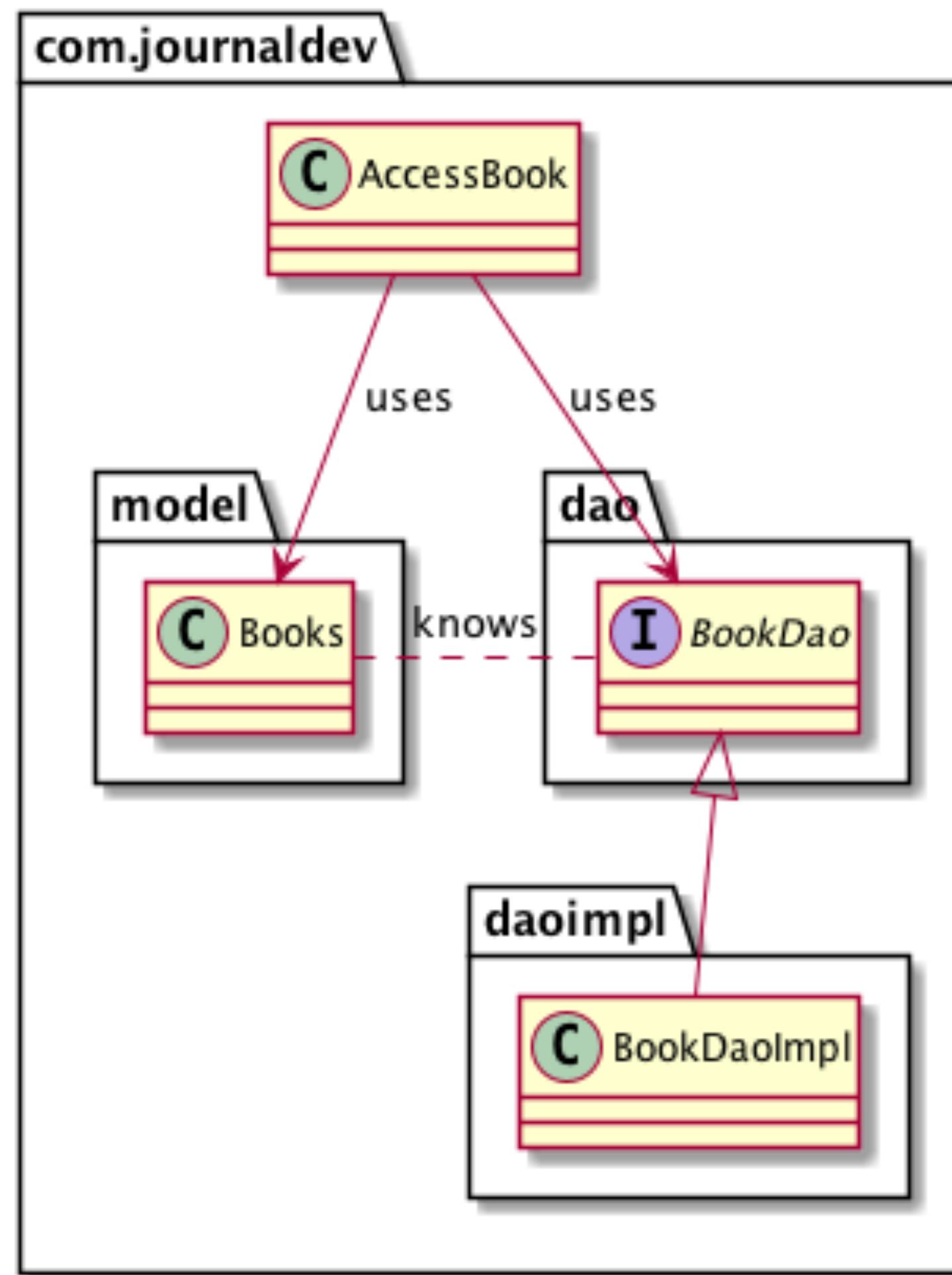
3.2. Patrons arquitectònics

Patró Data Access Object: (DAO) Interfície DAO

```
public interface GenericDao<T> {  
  
    /** Persist the newInstance object into database */  
    public void add(T newInstance);  
  
    /** Retrieve an object that was previously persisted to the database */  
    public T read();  
  
    public List<T> getAll();  
  
    /** Save changes made to a persistent object. */  
    public void update(T transientObject);  
  
    /** Remove an object from persistent storage in the database */  
    public void delete(T persistentObject);  
}
```

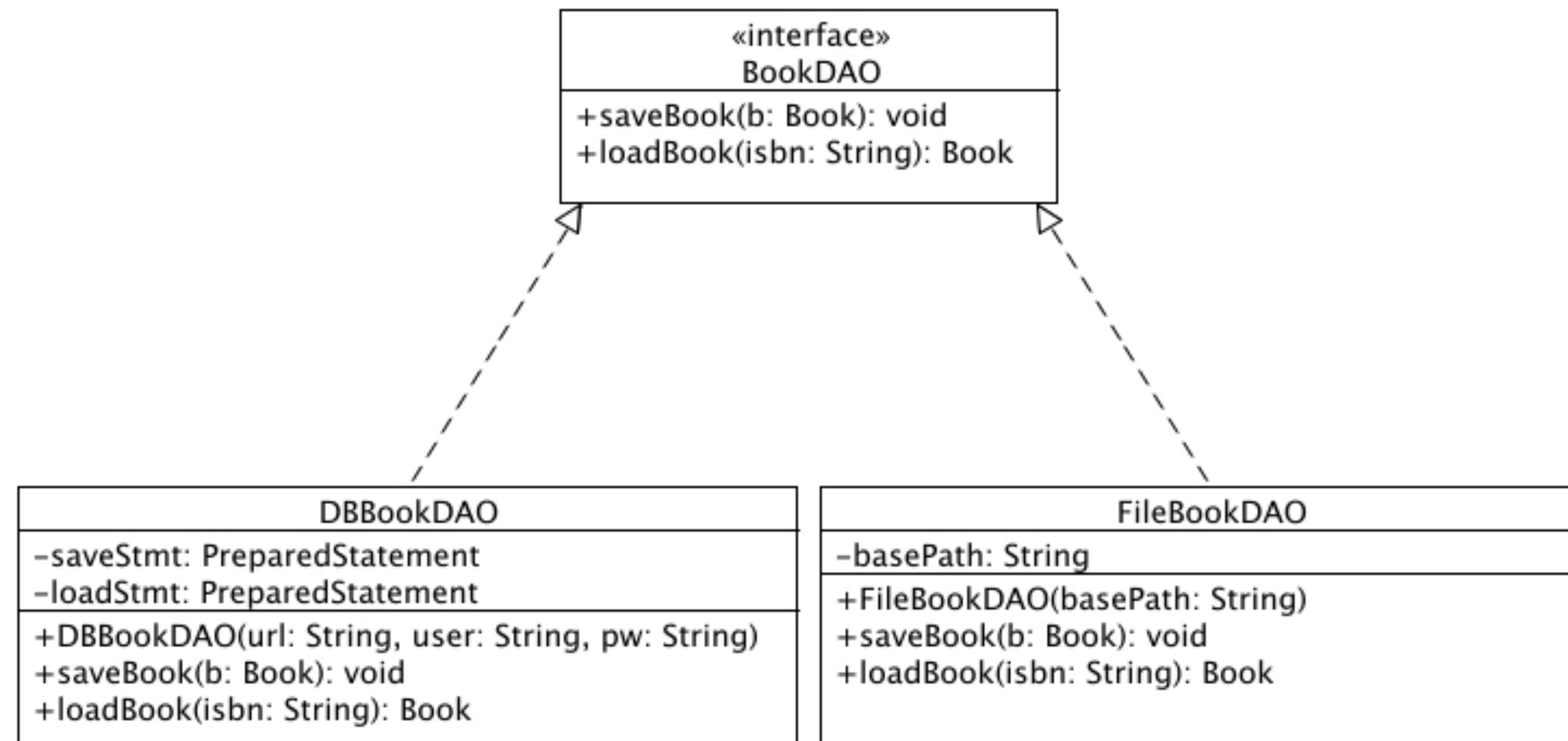
3.2. Patrons arquitectònics

Patró Data Access Object: (DAO) Exemples



3.2. Patrons arquitectònics

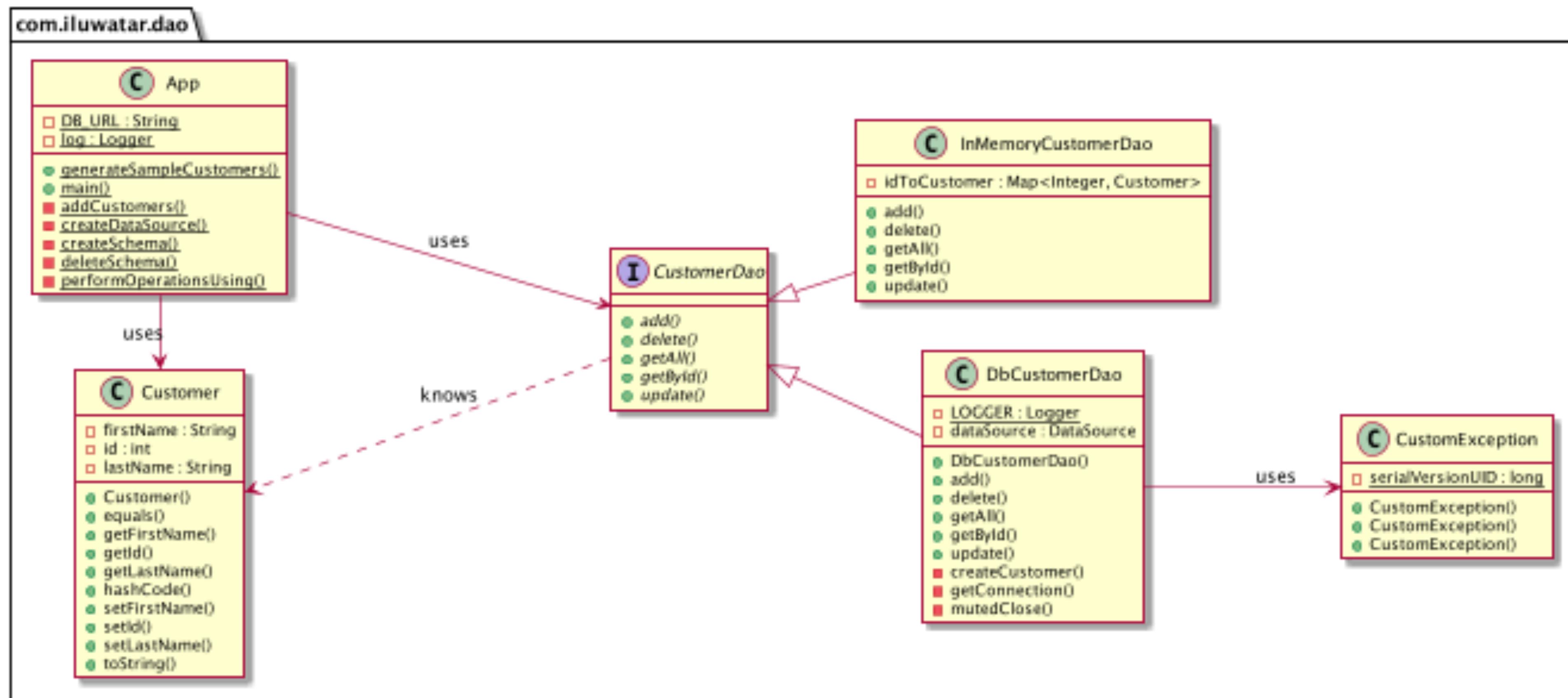
Patró Data Access Object: (DAO) Exemples



<http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/dao.html>

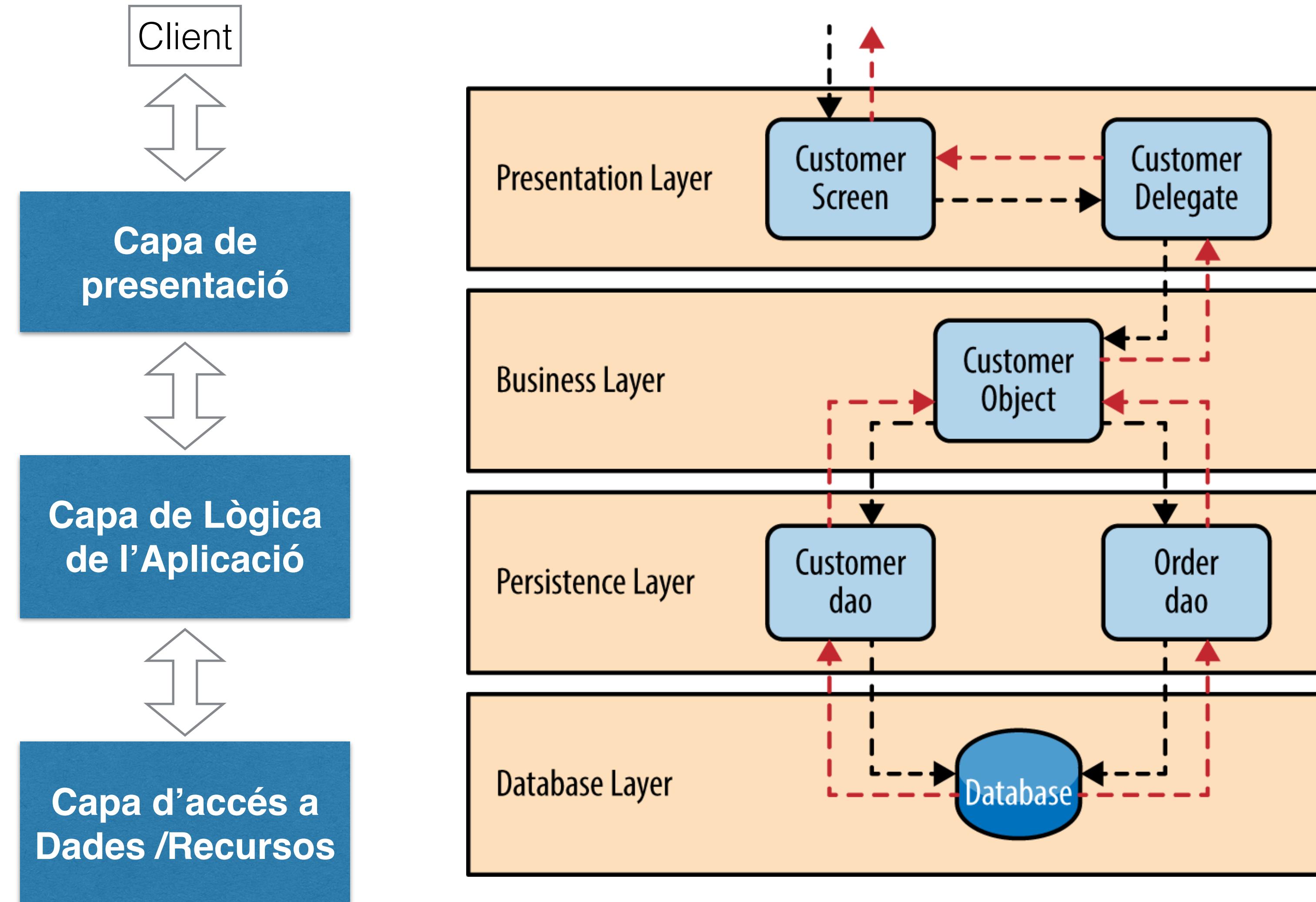
3.2. Patrons arquitectònics

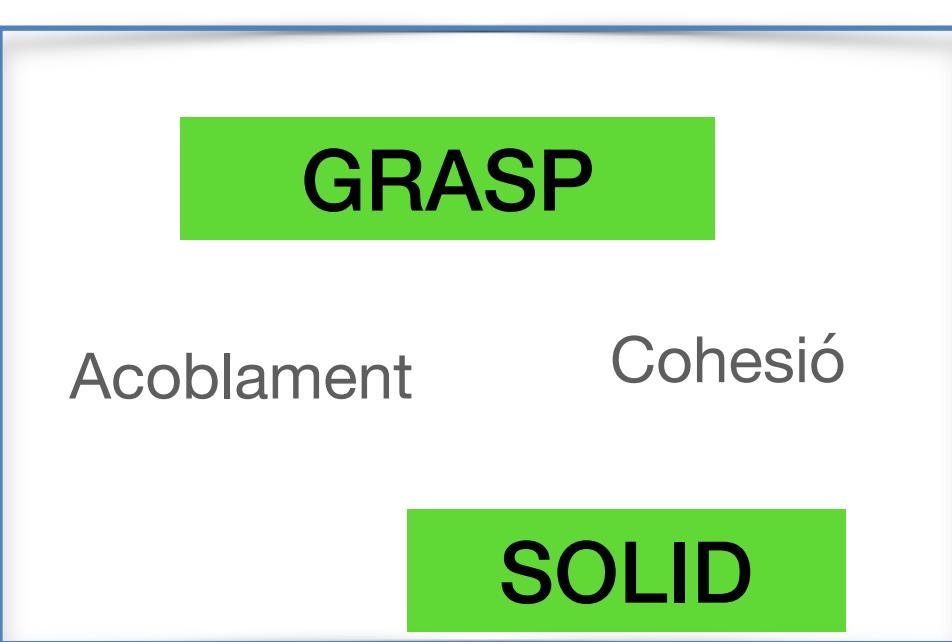
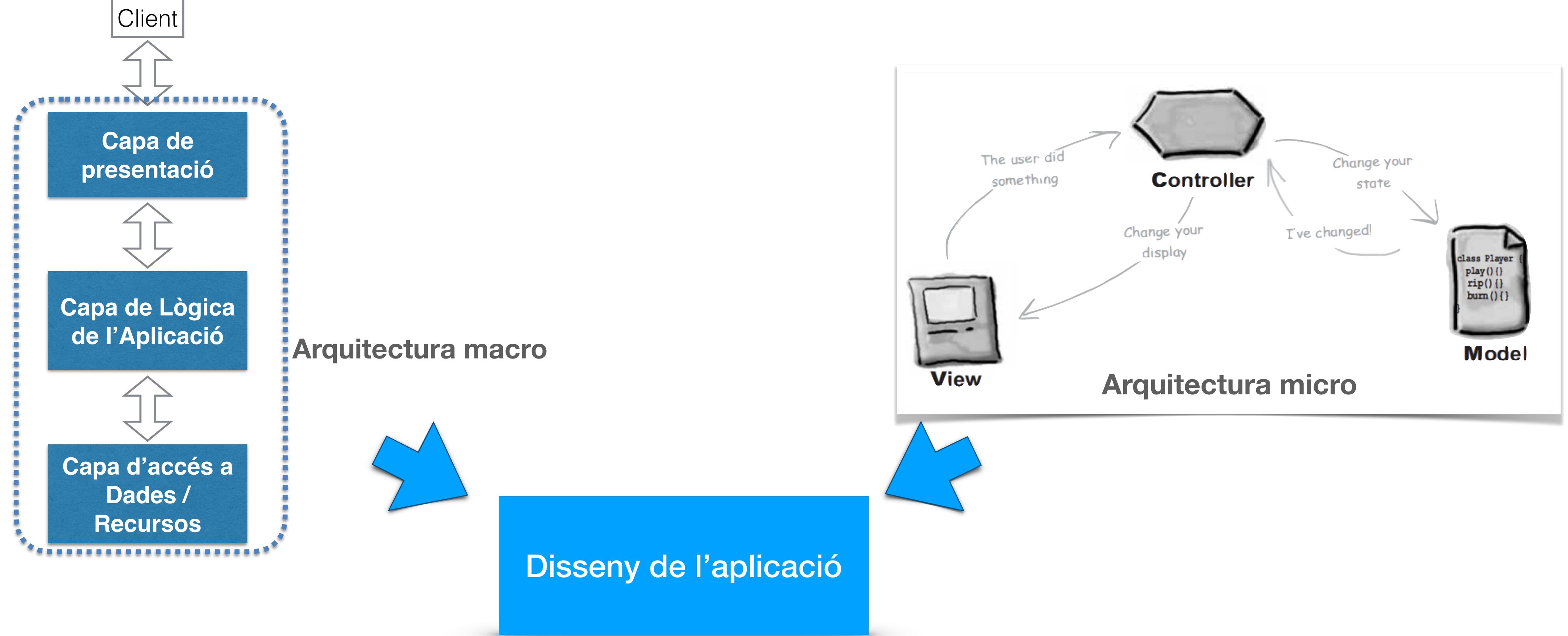
Patró Data Access Object: (DAO) Exemples



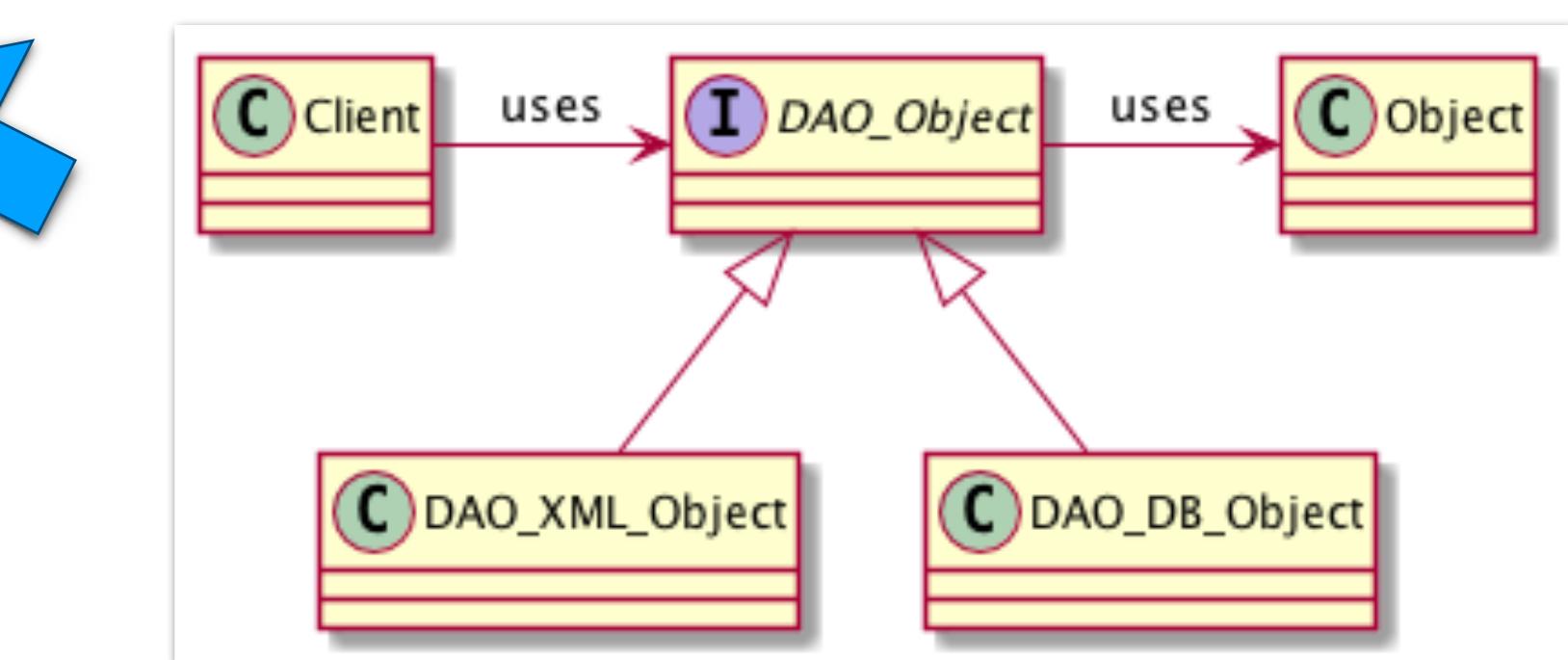
3.2. Patrons arquitectònics

Patró per capes:





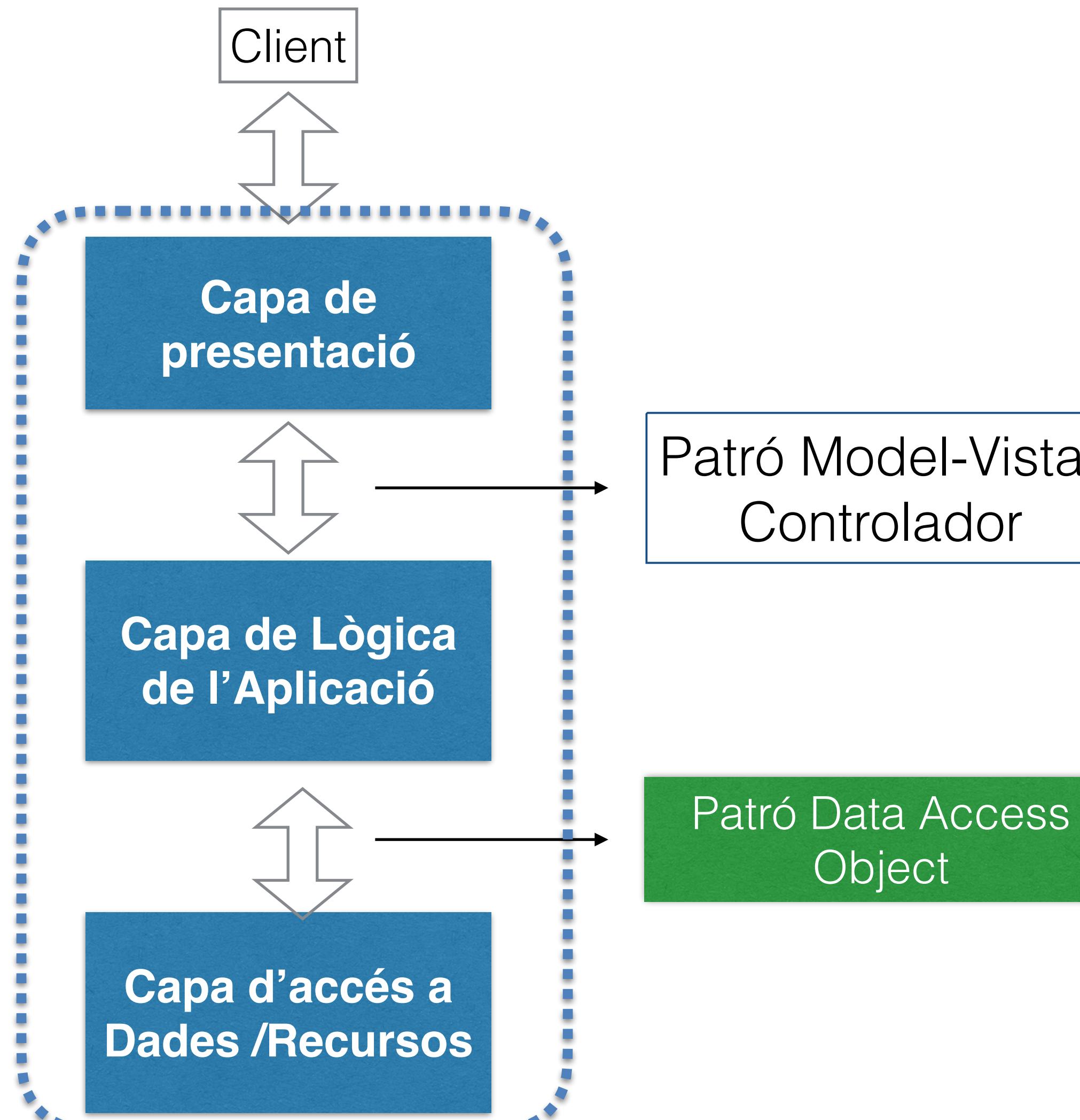
Criteris de disseny



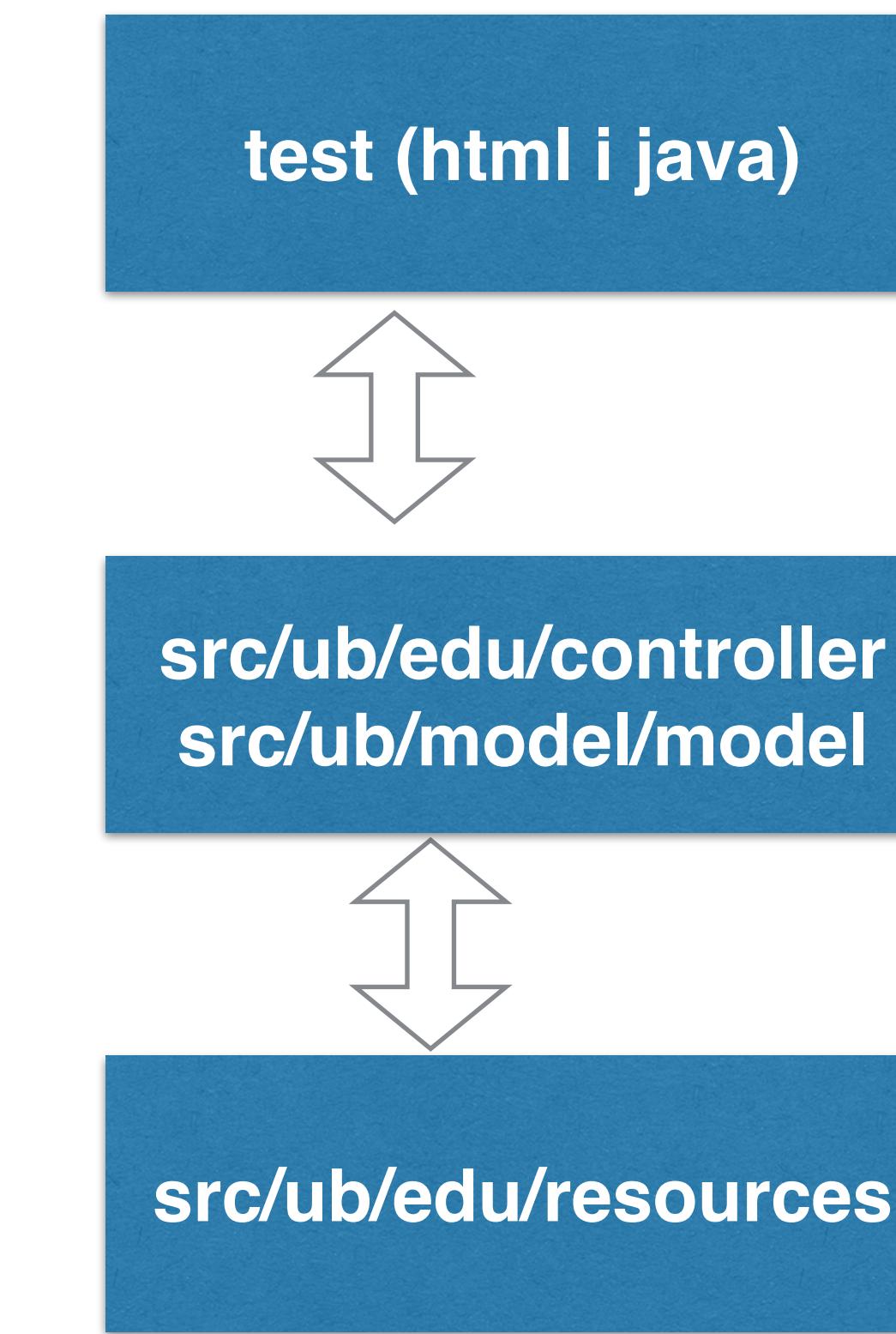
Patrons de disseny

3.2. Patrons arquitectònics

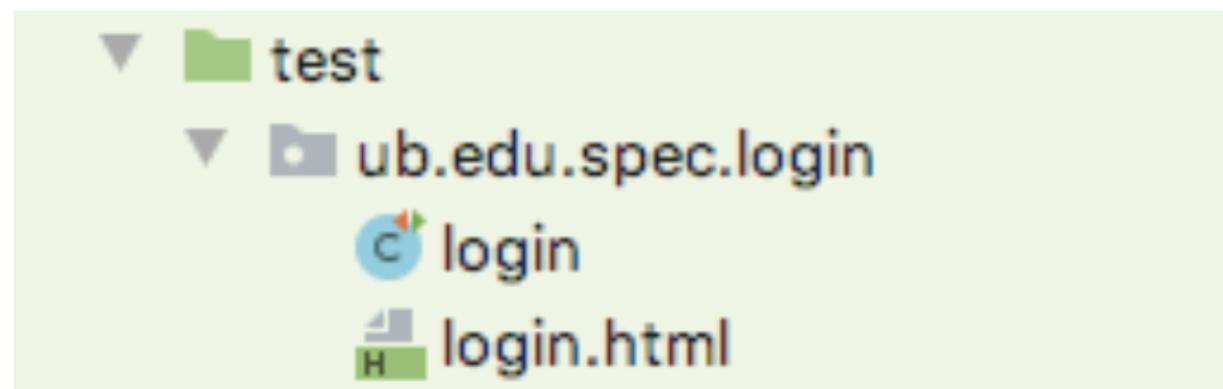
Patró per capes:



Pràctica 2



Capa de vista



Acceptance Criteria 14: Registre Soci: Registres correctes

El sistema dona opció d'enregistrar i els registres són correctes

Soci name	Password	Valid
ajaleonew@gmail.com	12wreTry	Soci Validat
pepito@domini.cat	qwerfrdsE1	Soci Validat
ajaleo@gmail.com	qwerfrdsE1	Soci Duplicat
pepitonew@domini.cat	qwert	Format incorrecte

```
@RunWith(ConcordionRunner.class)
public class registreSoci {
    private Controller controlador;

    @BeforeExample
    private void init() { controlador = new Controller(); }

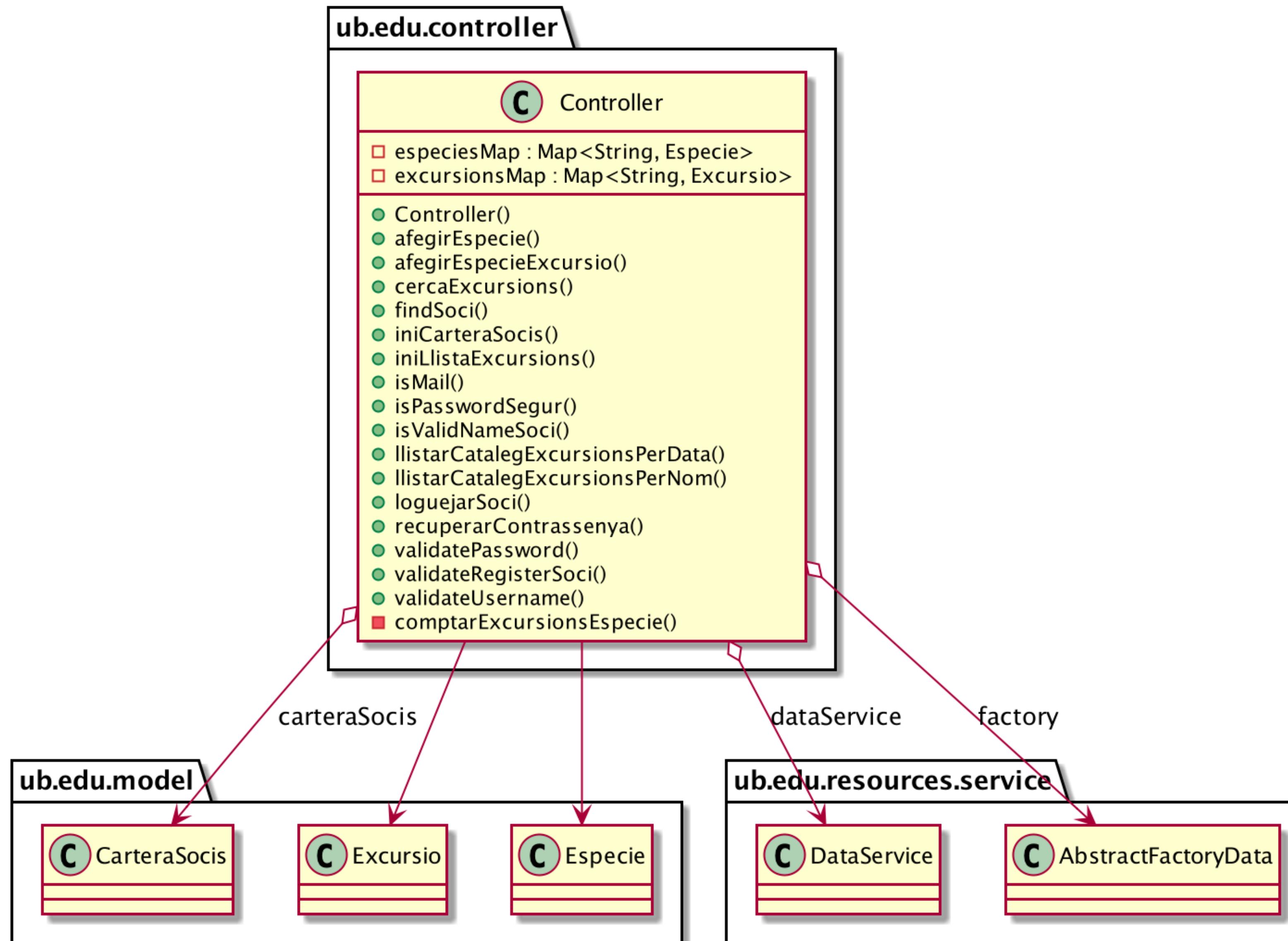
    public String getSoci(String a) { return controlador.findSoci(a); }

    public String validatePassword(String b) { return controlador.validatePassword(b); }

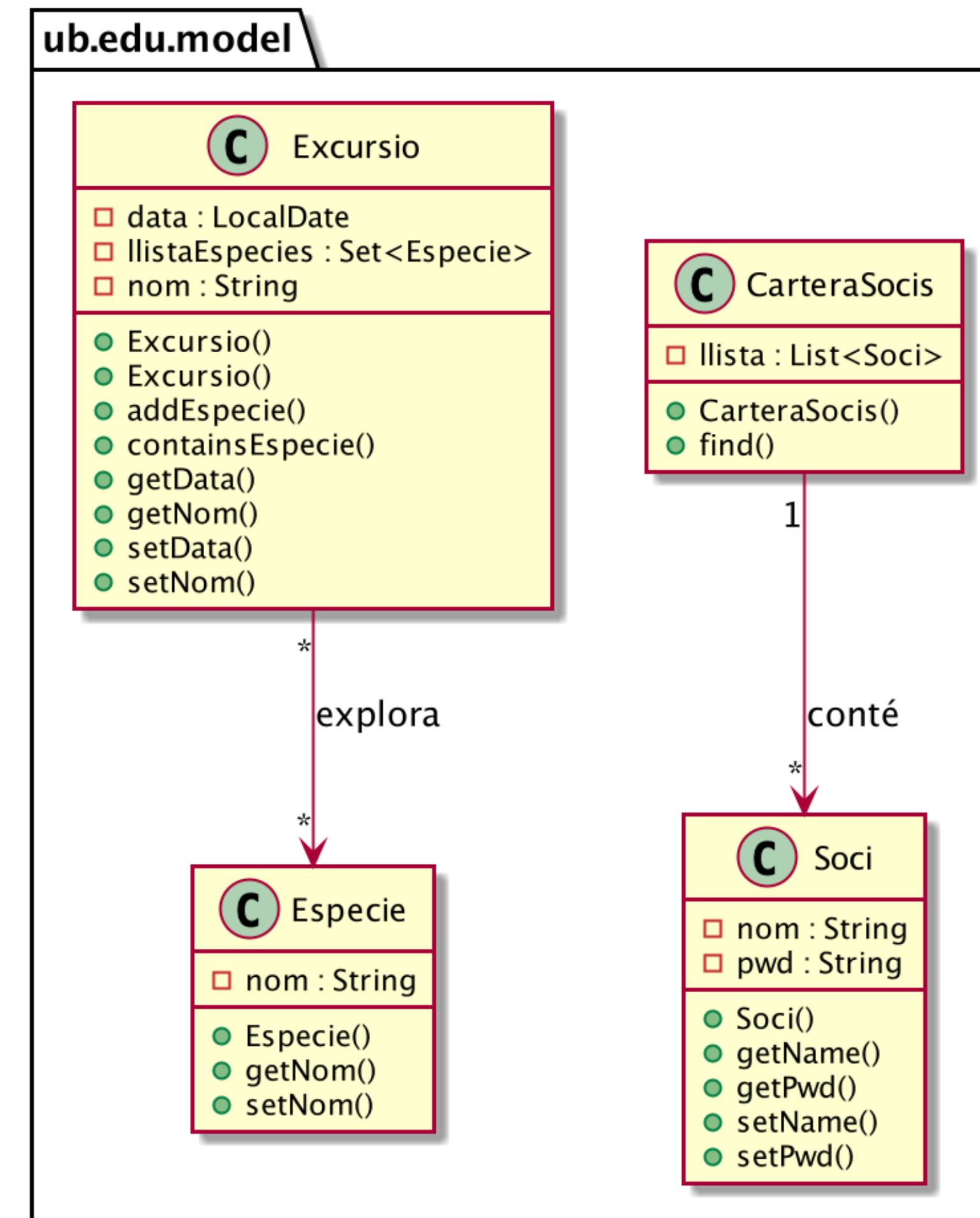
    public String validateUsername(String a) { return controlador.validateUsername(a); }

    public String isValidRegistre(String sociName, String password) {
        return controlador.validateRegisterSoci(sociName, password);
    }
}
```

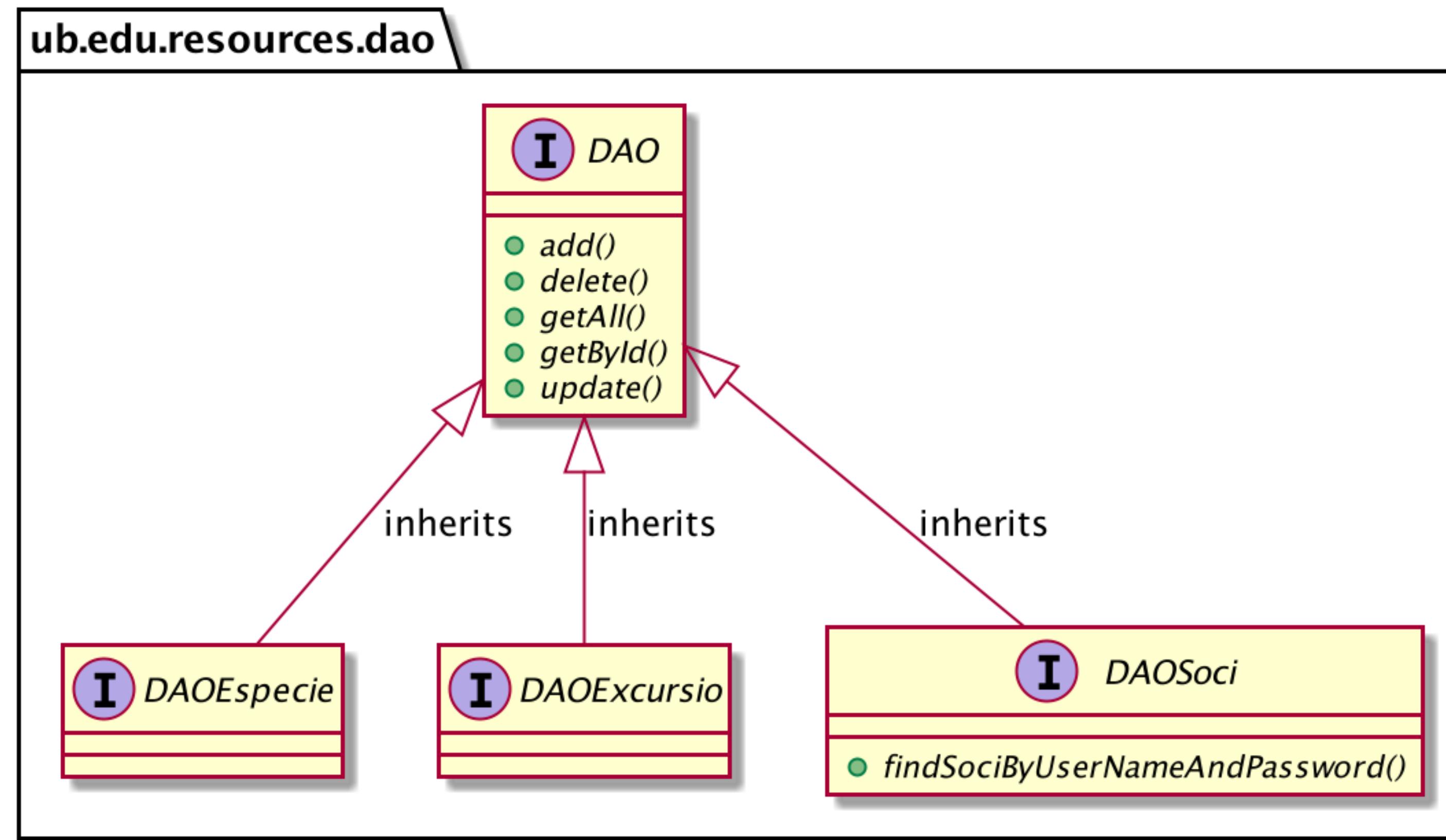
Capa de lògica (controller)



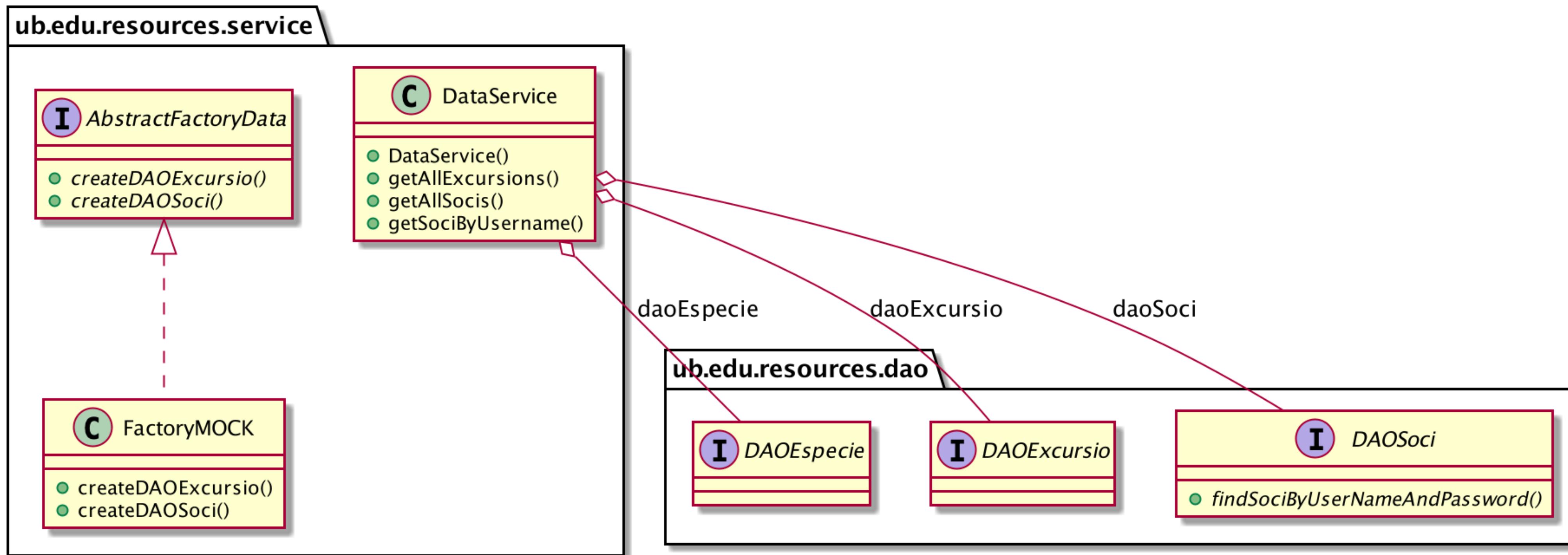
Capa de lògica (model)



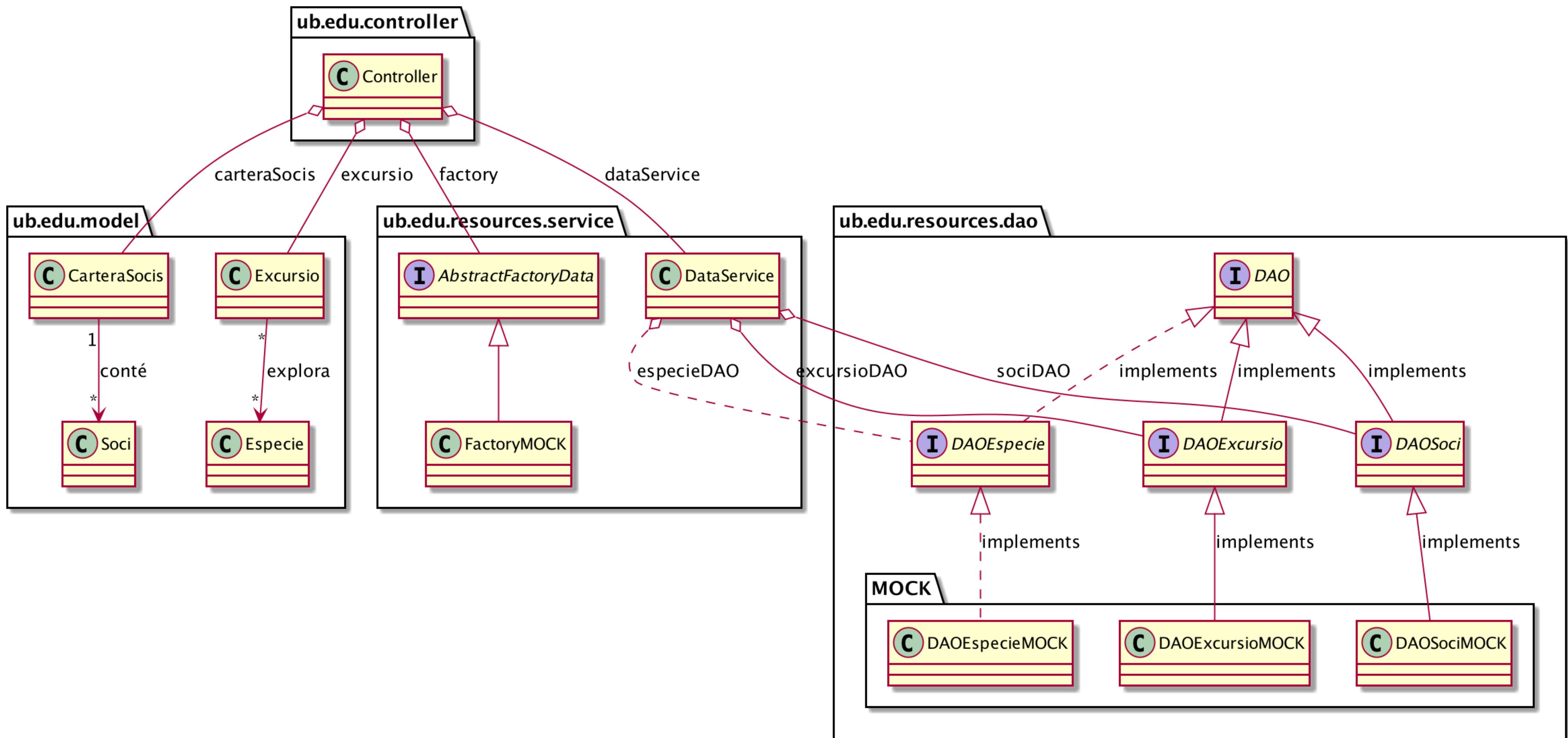
Capa de persistència: DAO



Capa de persistència



Arquitectura general



Pràctica 2

- **Parts:**

1. Refactoritzar seguint els criteris GRASP
 1. Respostes al document de l'enunciat
 2. Projecte
2. Noves prestacions en persistència:
 1. Respostes a l'enunciat
 2. [OPT] Connexió amb la capa de persistència utilitzant el patró DAO

Test d'acceptació	Criteris GRASP	Breu explicació de les classes canviades

Patró GRASP usat a la capa de persistència	Breu Justificació
Per exemple, Cohesió alta a les classes DAO-MOCK	Les classes DAO-MOCK internament només tenen la responsabilitat de gestionar les dades com si fossin una base de dades i cap altra