Manual del Projecte Base CexTrem-DAO

Per a realitzar lla 'entrega de la pràctica 2, us facilitem els següents elements:

- Un projecte base IntelliJ estructurat en una arquitectura de tres capes: (1) la capa de la Vista, (2) la capa de la lògica de negoci (controlador i model) i (3) la capa de persistència o de recursos.
- En el projecte de IntelliJ s'inclou un exemple de com executar els tests a Concordion de les dades carregades des d'un DAO. A l'exemple que es dóna les dades es creen en el mateix programa tot i que està llest per a poder fer la implementació de DAOs que carreguin d'un fitxer o des d'una base de dades.

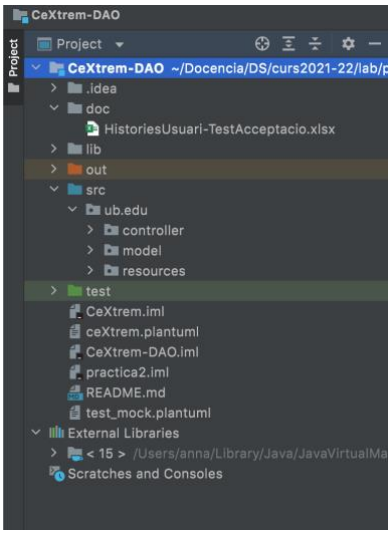
El pots clonar del classroom:

https://classroom.github.com/a/Ys-xd_Cj

Mitjançant aquests elements haureu de carregar un conjunt de dades inicials al vostre model de l'aplicació per a fer els tests seguin els criteris d'acceptació de les històries d'usuari que heu definit en la pràctica 1. A continuació us expliquem l'estructura del projecte proporcionat i el procés que heu de seguir.

Estructura del projecte

Quan obris el projecte CeXtrem-DAO, et trobaràs aquesta disposició de les carpetes:



Si s'analitzen les tres capes:

(1) la capa de la Vista és la part de l'especificació i del test (paquet test)

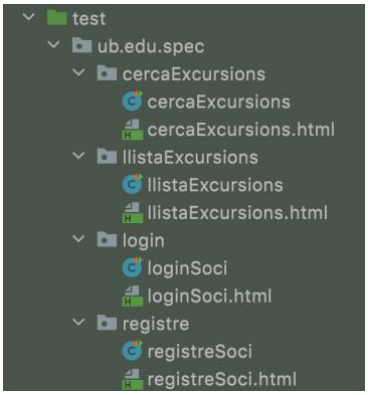
(2) la capa de la lògica de negoci està formada per les carpetes controller i model del paquet src

(3) la capa de persistència o recursos està formada per la carpeta resources del paquet src.

1. Capa de la Vista:

És la capa dels tests, que consta de **carpetes** (o paquets), **una per a cada història d'usuari**. Cada carpeta conté **un fitxer d'especificació html (Spec) i el seu java (fixture)** per a cadascuna de les **històries d'usuari** que serveix als tests de l'especificació en Concordion. Dins de cada html es localitzen els diferents exemples amb dades concretes per a cada història d'usuari.

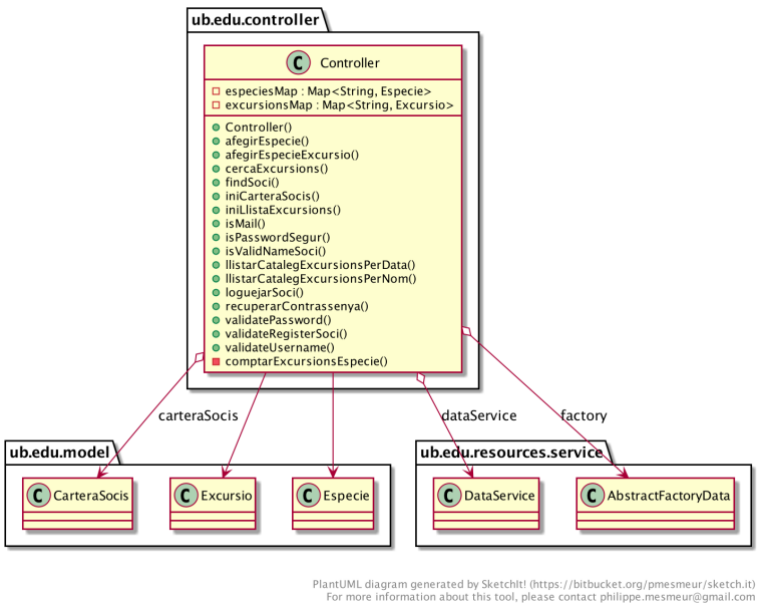
En l'exemple es proporcionen les històries d'usuari de Registre, Login, Llista Excursions i Cerca Excursions. Fixa't que en el fitxer Java tots els tests poden compartir la mateixa inicialització del controlador i de les dades. En els tests de Registre, Login i Llista Excursions, les dades s'inicialitzen des de la capa de persistència (o de recursos). En el test de Cerca Excursions per espècies, les espècies s'afegeixen en el mateix test.



2. Capa de lògica de negoci:

És la capa on es situa el Controlador, responsable de servir a la Vista però també d'inicialitzar el model i la capa de persistència.

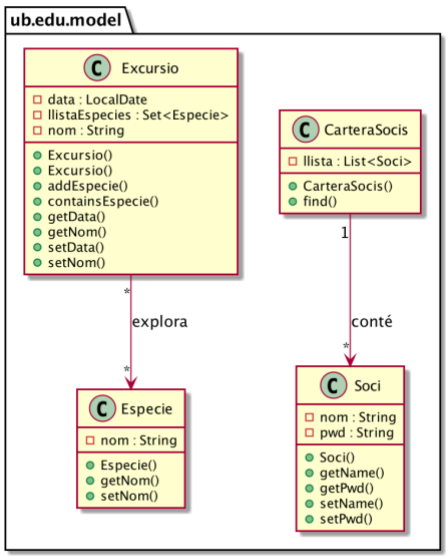
CONTROLLER's Class Diagram



En l'exemple el controlador, inicialitza la capa de persistència via el DataService i després inicialitza una classe CarteraSocis del model (aquesta és una classe exemple que no té per què estar en el teu projecte final).

En la part del model es proporcionen les classes CarteraSocis, Soci, Excursio i Espècie a mode d'exemple, però les hauràs de modificar o afegir les que tens dels projecte de la pràctica 1.

MODEL's Class Diagram

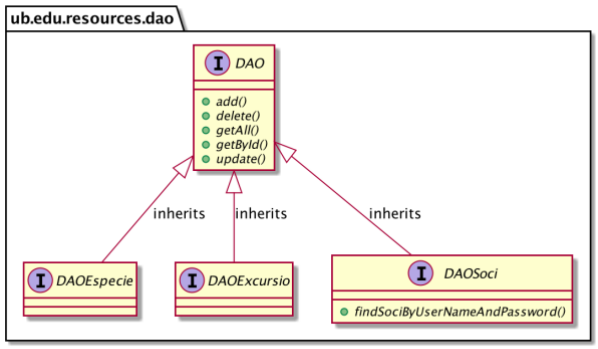


3. Capa de persistència o de recursos:

En aquesta capa es troben tots els serveis per a poder carregar les dades del model de forma independent de la base de dades o de la font d'on es treuen les dades del model. En aquesta capa s'han fet servir diferents patrons de disseny que s'estudiaran a l'assignatura (el patró de Facade, el patró d'Abstract Factory i el patró DAO - Data Access Object).

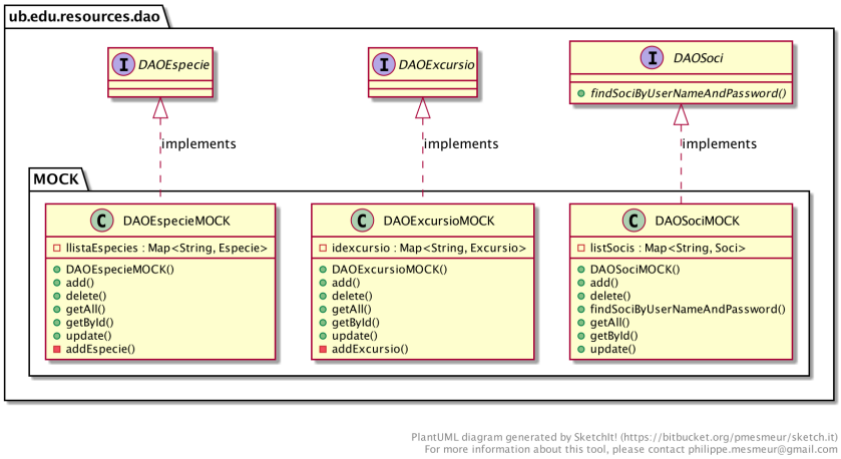
El patró DAO permet accedir a les dades persistents (o de la base de dades, per exemple) i retornar-les ja en forma de classes del model. Per a aconseguir aquesta correspondència s'implementa per a cada classe bàsica (DOJO) una classe DAO que té les funcionalitats concretes CRUD (creació, lectura, modificació i esborrat) i segueixen una interfície ben definida. Per exemple, per a obtenir dades de Soci de la base de dades, es definirà una interfície DAOSoci, que serà implementada per la corresponent connexió de lectura a la Bases dades o al repositori on es tinguin les dades.

DAO's Class Diagram



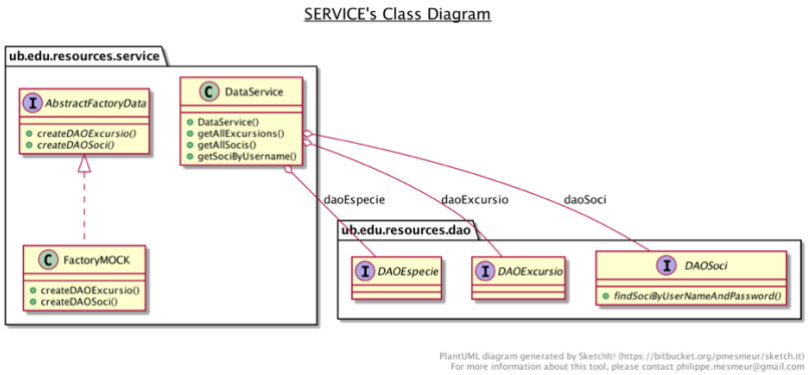
En el cas del codi base, per a poder fer les proves independentment de la base de dades, es proporciona una primera implementació **MOCK** (és a dir una simulació en memòria del que retornaria la base de dades).

MOCK's Class Diagram

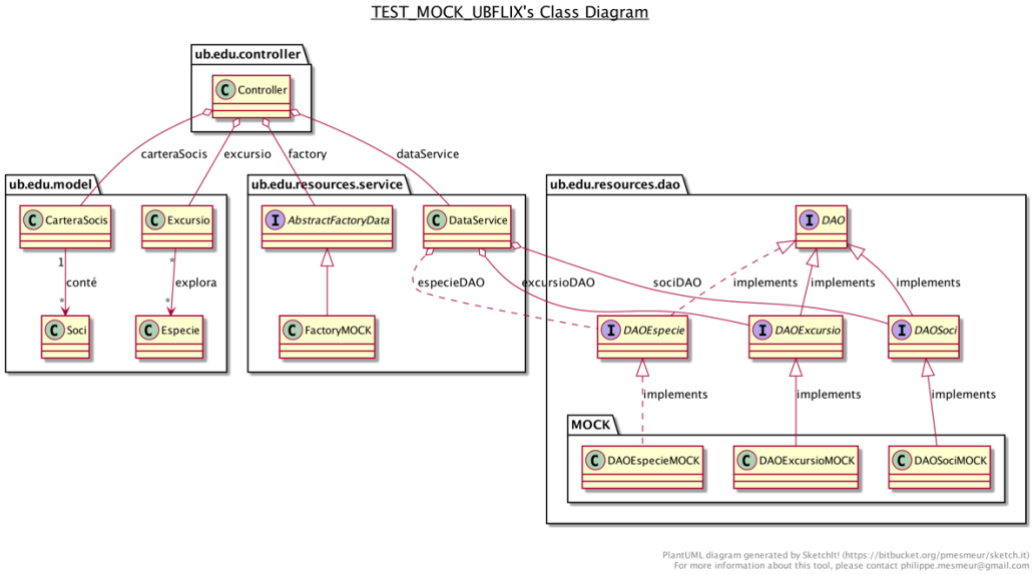


Per accedir a aquest serveis, s'ha dissenyat via el patró Facade, la classe DataService que servirà d'API o d'interfície entre la capa de lògica de negoci i la capa de persistència. Aquesta classe serà l'encarregada de crear la connexió amb la capa de persistència i donar accés a les dades de forma transparent. Per a poder canviar l'origen de les dades (ja sigui des del MOCK o des de la base de dades en un futur), s'injecta a la classe DataService, l'objecte encarregar de construir les connexions amb les dades. Aquest objecte creador es basa en el patró AbstractFactory. Concretament, en el projecte podeu trobar la classe FactoryMOCK, que és l'encarregada de crear els objectes DAO que donaran accés al MOCK.

En la classe DataService ara s'especifiquen alguns mètodes, com getAllSocis() però cal que l'amplieu amb els mètodes necessaris per a aconseguir les dades del vostra model.



Finalment, a mode de resum, en el següent diagrama de classes es reflecteix la part de la carpeta src del projecte (amb les dues capes de lògica de negoci i de persistència). Fixeu-vos que des de fora de la capa de persistència, el model no coneix d'on provenen les dades, i dóna el servei al controlador sobre les peticions concretes de la vista.


TEST_MOCK_UBFLIX's Class Diagram
ub.edu.controller
Controller
carteraSocis
excursio
factory
dataService
ub.edu.model
CarteraSocis
Excursio
1
conté
explora
Soci
Especie
ub.edu.resources.service
AbstractFactoryData
DataService
FactoryMOCK
especieDAO
excursioDAO
sociDAO
ub.edu.resources.dao
DAO
implements
DAOEspecie
DAOExcursio
DAOSoci
MOCK
DAOEspecieMOCK
DAOExcursioMOCK
DAOSociMOCK
PlantUML diagram generated by SketchIt! (https://bitbucket.org/pmesmeur/sketch.it)
For more information about this tool, please contact philippe.mesmeur@gmail.com