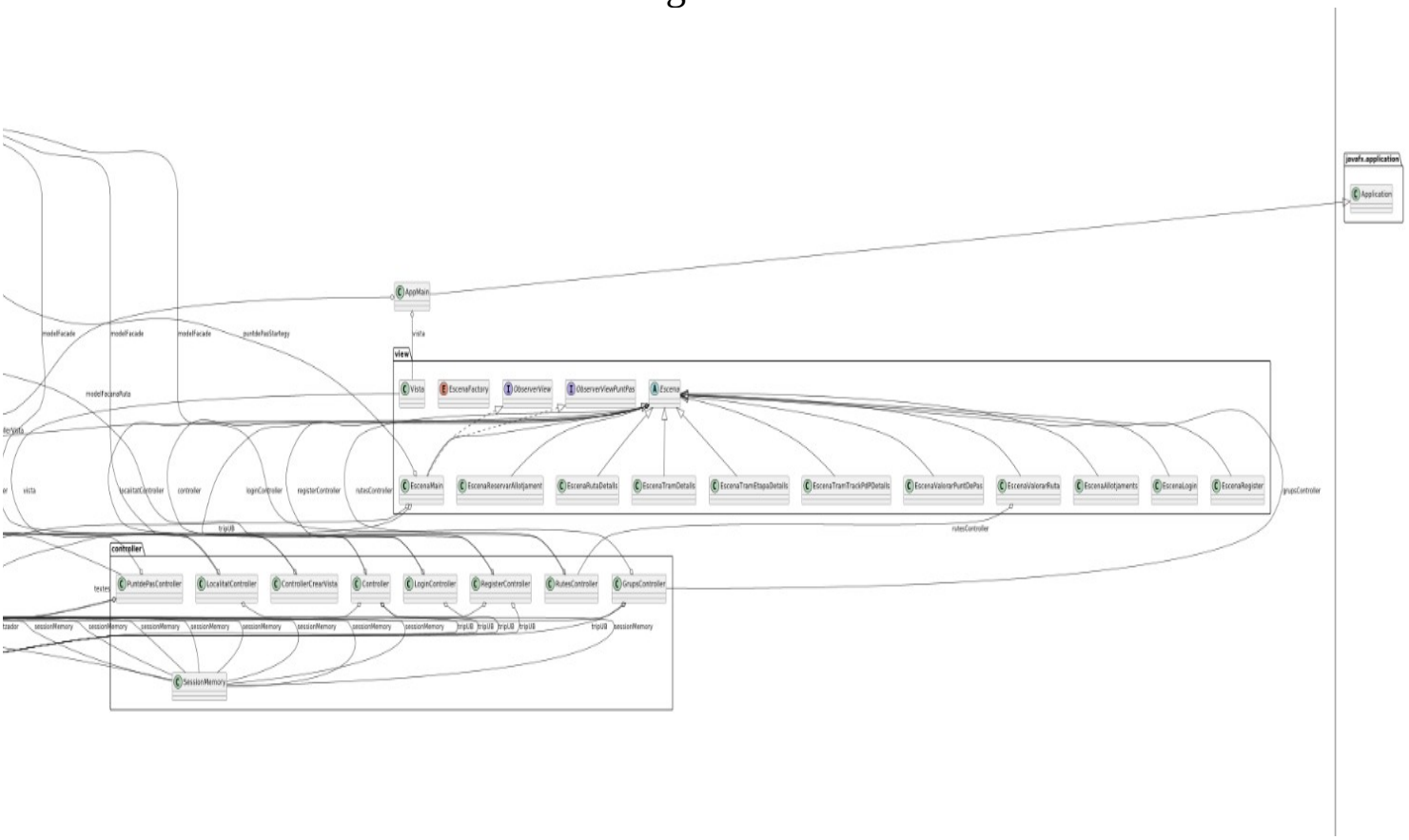
Pràctica 4 – TripUb.DB

Lab A: Grup A03

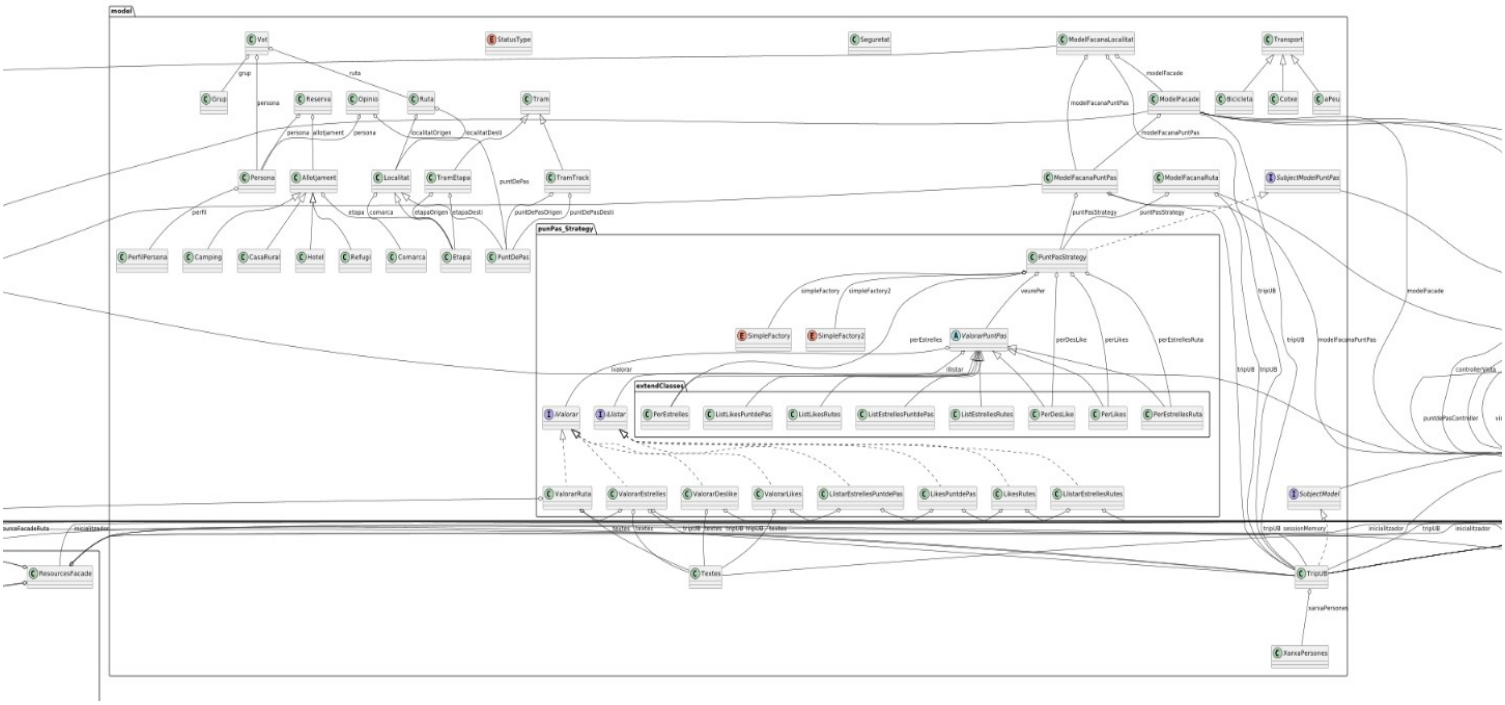
Albert Vera Duran
Xavier Vera de la Gala

Index

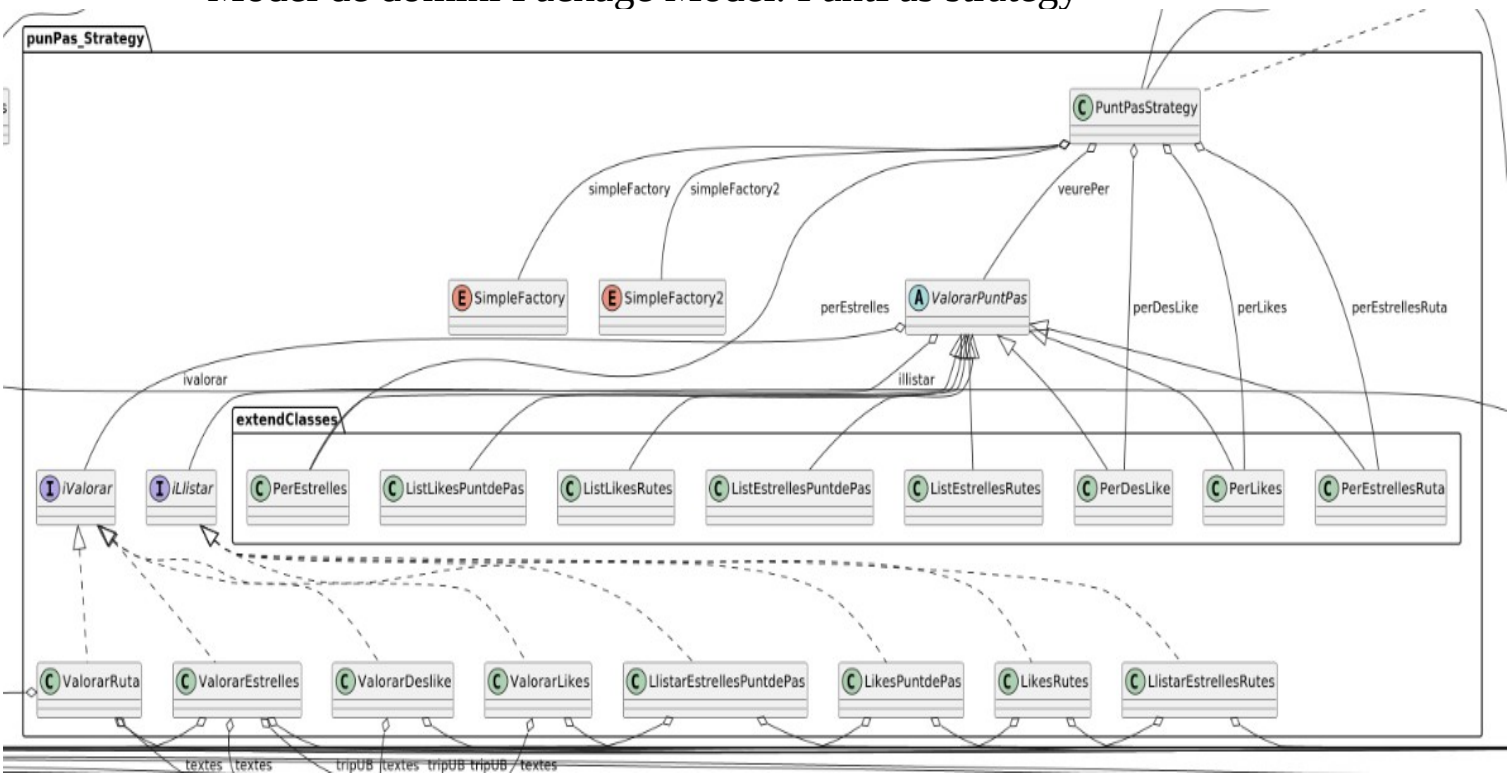
Model de domini Package Controller i View:



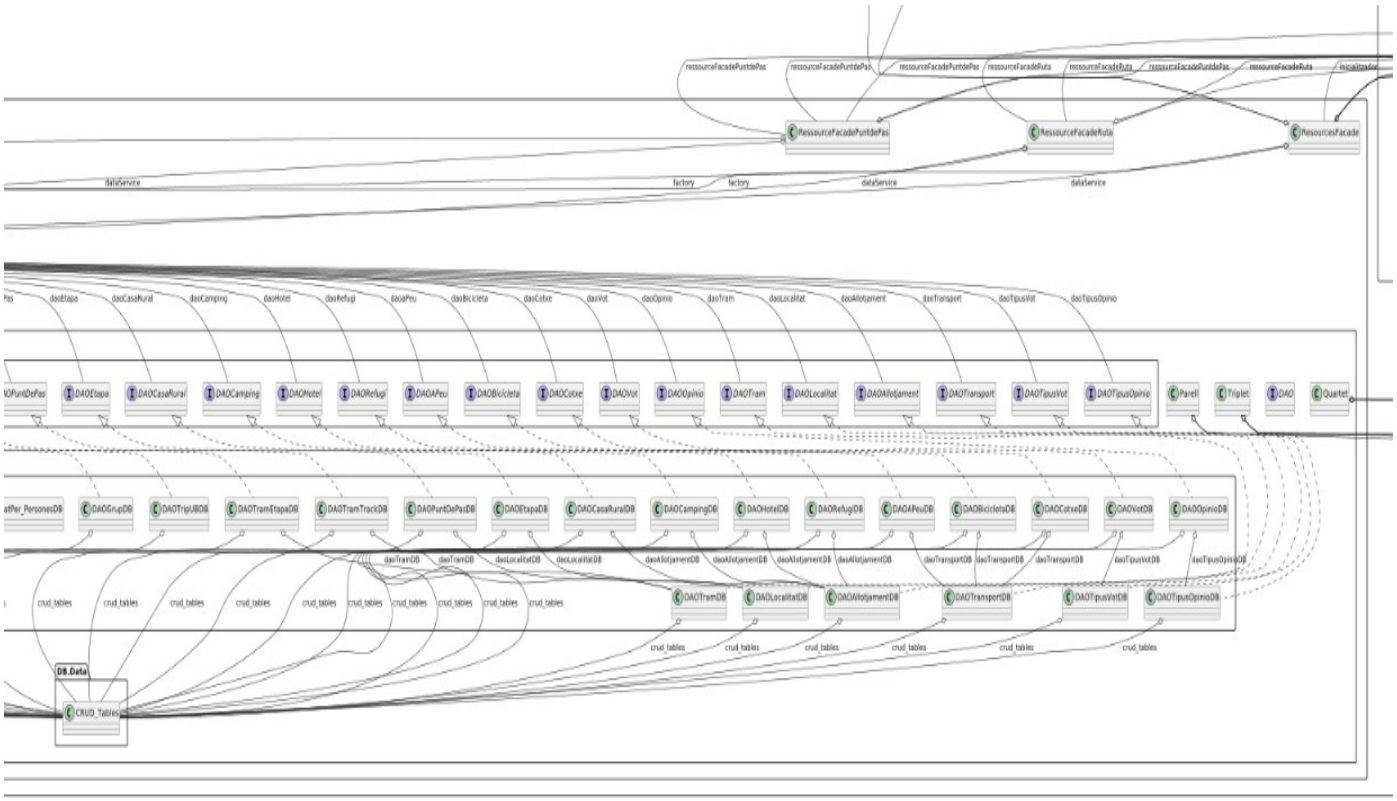
Model de domini Package Model:



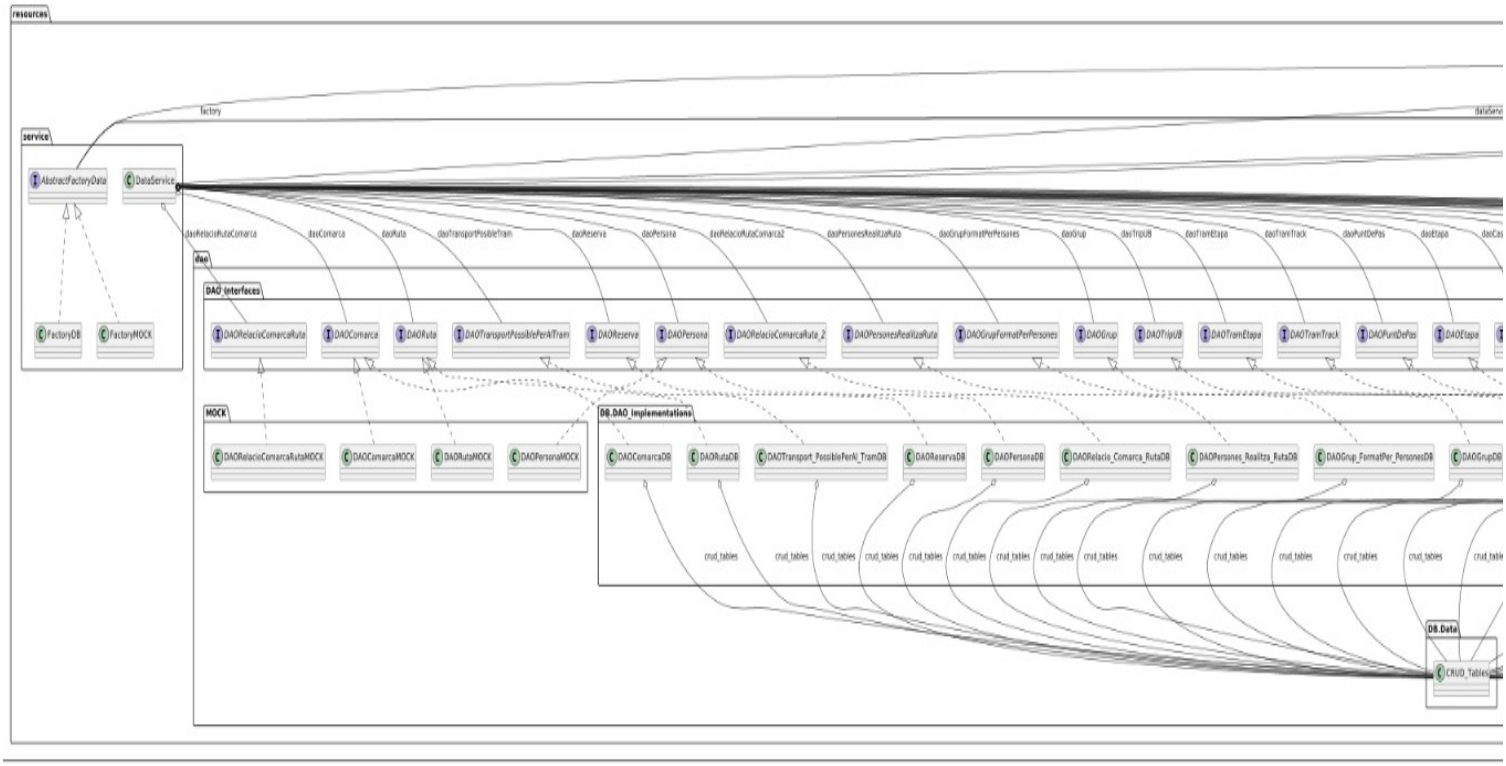
Model de domini Package Model. PuntPas strategy



Model de domini Package Ressoureces: mitat-1.



Model de domini Package Ressources: mitat-2.



Millores respecta pràctica-4.

La millora principal és la utilització d'una base de dades i l'entorn gràfic amb JavaFx, però això ja sabem que no és gràcies a mi.

En aquesta pràctica he utilitzat els patrons que ja vaig fer servir a la pràctica anterior i la millora principal ha sigut la utilització del patró Observer.

Aquest patró l'hi he utilitzat perquè hi hagi una ràpida comunicació entre la Vista i el Model, de manera que, qualsevol canvi en el model de dades sigui notificat instantàniament a la vista.

Patrons emprats.

Singleton: En els Controllers, Façanes i Ressources Façanes.
Per evitar múltiples instàncies d'una mateixa classe i estalviar recursos de memòria.

Façana: Entre Controllers i Model. I entre Model i ressources DB.
Perquè qualsevol canvi en el Model o persistència no afecti els Controllers.

Strategy: Per a gestionar Valoracions d'un Punt de pas o una Ruta.
Evitant d'aquesta manera Open-Close utilitzant múltiples condicionals.

Abstract Factory: A la capa service.
Per instanciar o crear classes de la BD.

Factory Method: A la classe PuntPasStrategy, mètodes llistarLikes i GetTopEstrelles.
Mètodes on amb el paràmetre criteri, diferencia si ha de llistar per Punt de Pas o per Rutes.
Primer, obté la classe que extend de classe abstracta ValorarPuntPas amb SimpleFactory.
Segon, obté la classe que implementa iLlistar, amb SimpleFactory2.
Tot això simplement amb un paràmetre string, aconsegueix situar-se a la implementació del mètode llistar().

Observer: Implementat PuntPasStrategy, TripUB, EscenaMain.
Quan es produeix un canvi les classes del model "PuntPasStrategy" o "TripUB", notifica a "EscenaMain" que alguna cosa a canviat.
El funcionament és que la vista "EscenaMain" es registra a la classe que ell està interessat en conèixer a l'instant, si hi ha algun canvi en les dades.
En aquest cas la vista es registra a "PuntPasStrategy" i "TripUB" i en el moment que ja un canvi a les dades, són elles que notificant a EscenaMain instantàniament i directament la modificació en les dades.

Diagrama de classes: Controller.

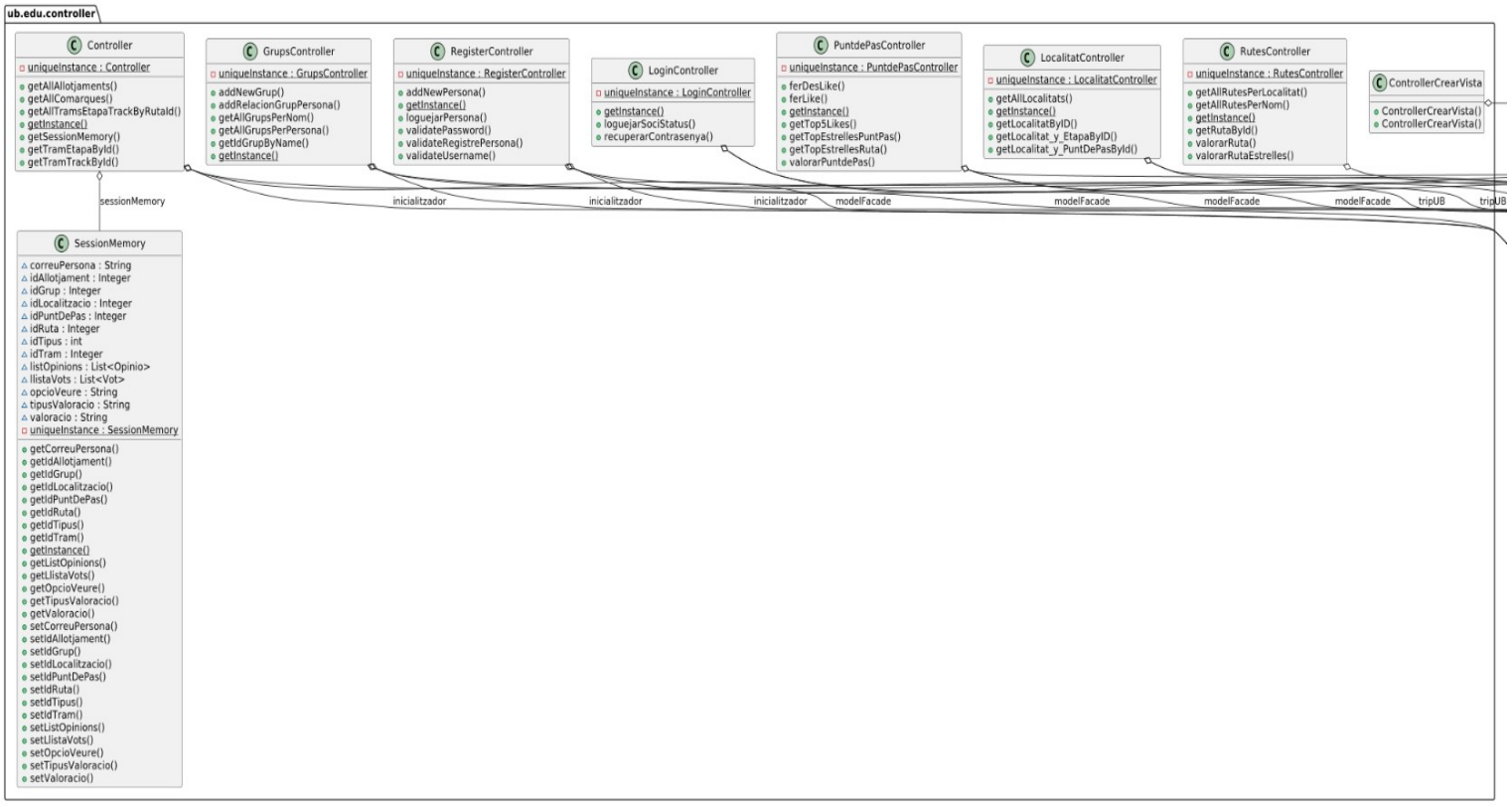


Diagrama de classes: Controller amb Model.

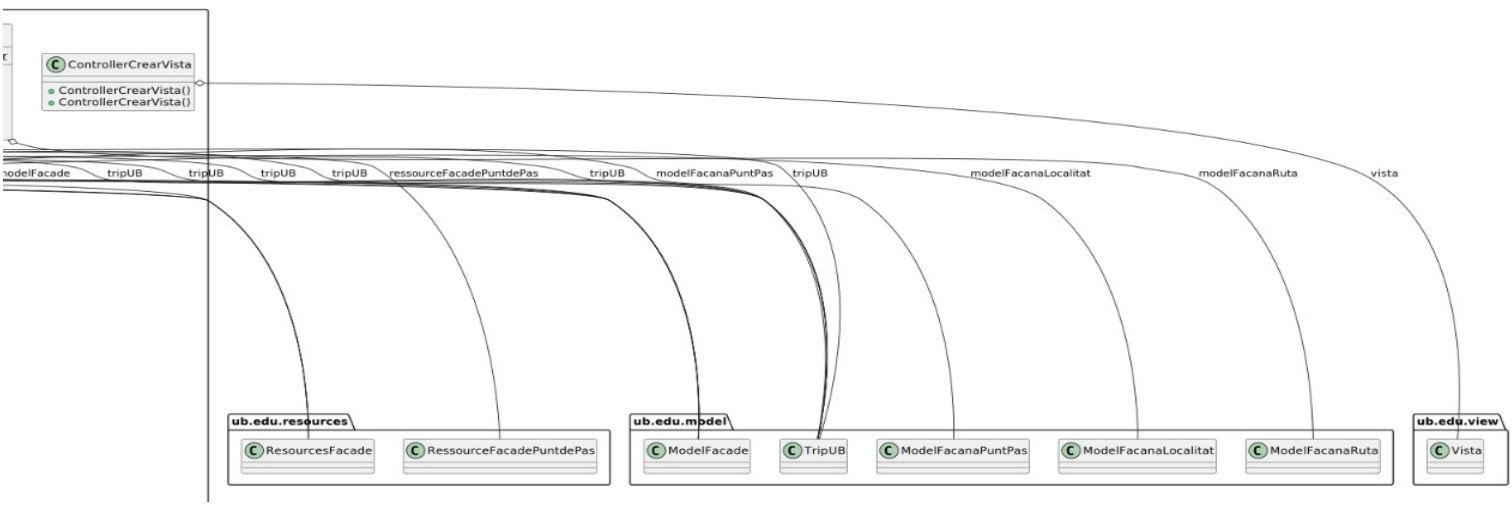


Diagrama de classes: Model.

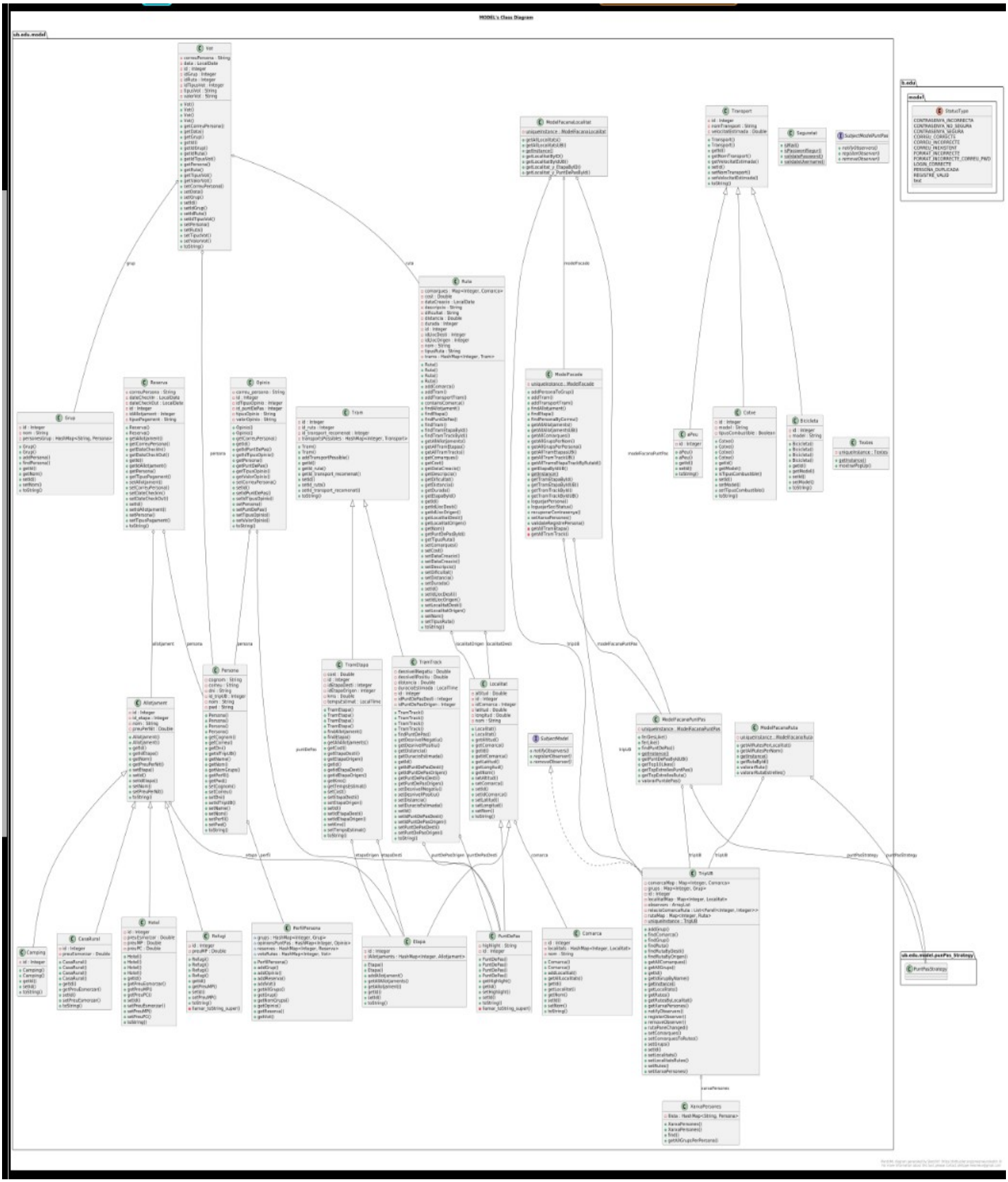


Diagrama de classes: Model. PuntPasStrategy.

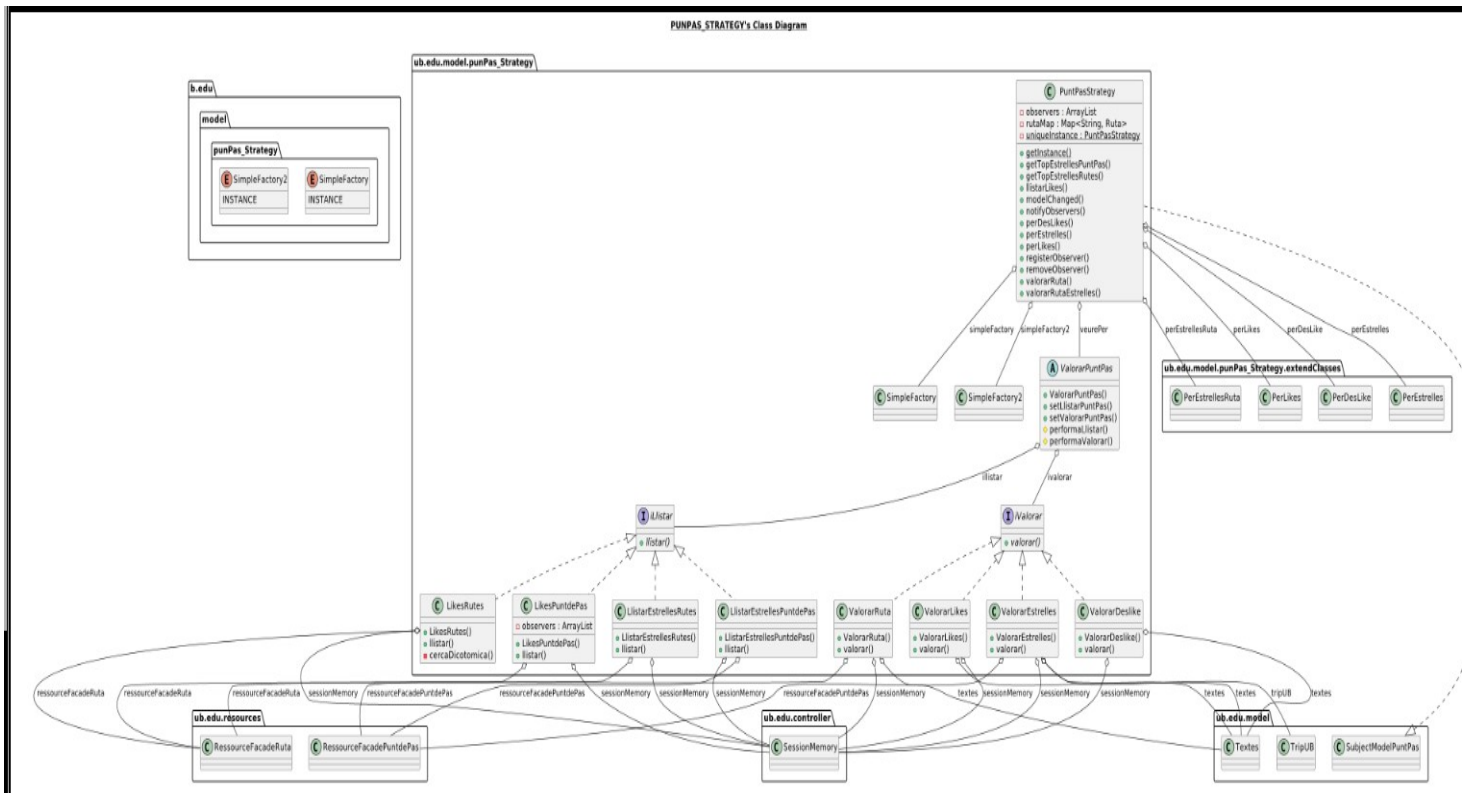
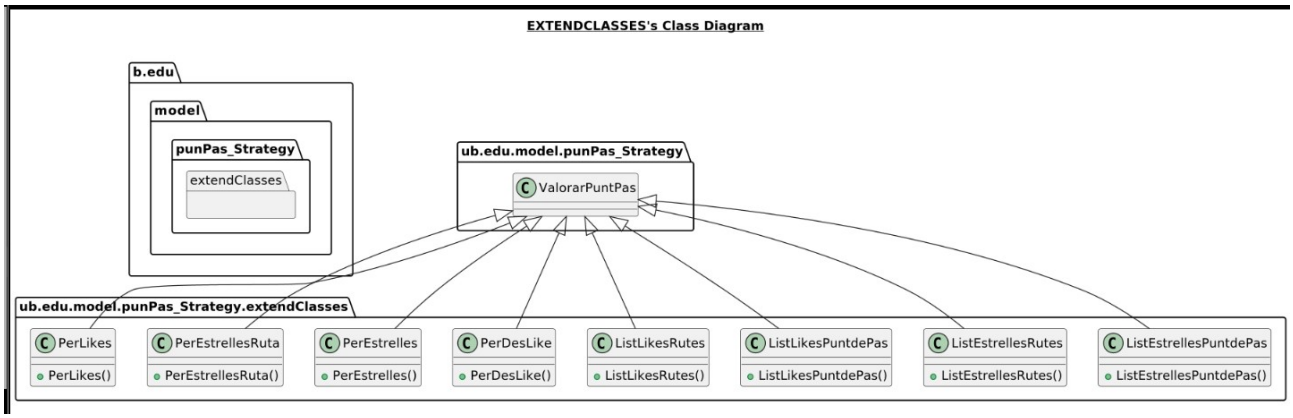


Diagrama de classes: PuntPasStrategy. extendClasses



Conclusions:

Qui crea qué ?

El projecte comença amb el dilema de qui és responsable de crear la vista i qui crear el controlador.

El controlador és qui es comunica amb la vista i qui es comunica amb el model. Per tant, quan inicia l'aplicació per AppMain, és el Main on es crea el controlador i és des del controlador des d'on es distribuirà totes les tasques de l'aplicació.

Llavors el Main crea el controlador i el controlador és on es crea el Model, perquè és el controlador qui es comunica únicament amb el model. Té lògica que el controlador creï model.

Mostrar valoració estrelles.

M'hi trobat amb un dilema i de moment ho he deixat així, perquè el temps s'ha acabat.
Es tracta de valorar per estrelles.

Quan valores per estrelles, com que quan valores per Like, Unlike o Estrelles tot acaba notificant els canvis a la vista amb el mateix mètode update().
Doncs em trobo que retorna un Map<String, Integer>.

Llavors, per la valoració per estrelles, hauria de ser que retornes un Double, no un Integer. Perquè se'n un Integer de vegades el valor no eś veu alterat després d'una valoració, ja qué, a les dades si que ha canviat, però sí el valor resultant és "2,2", llavors mostra "2".

Però en un inici no ho hi fet, per no duplicar mètodes innecesariament. Hauria de sobrecarga el mètode update() i crea un altre mètode a l'intefície per llistar.

Trobava que embrutava més el codi i l'exercici tampoc especifica que estrelles hagi de ser Double. Les estrelles canviaran de valor només amb valors enters d'1 a 5.

Pestanya Punt de Pas & Rutes.

Hi he vist la necessitat d'afegir una pestanya, per diferenciar quan es mostren dades de les valoracions per Punt de Pas o per Rutes.

Feina feta.

- Buscar una ruta per localitat, utilitzant el filtre per localitat, li dius una ruta i et diu quina ruta passa per aquesta localitat.

- Mostrar valoracions per criteri diferent. Per Like, Unlike, Estrelles.
  Fent servir la pestanya “Valoracions per: “, et mostra valoracions per **Punt de Pas** o **per Rutes**.
  Quan es tracta de Rutes, tens opció de fer servir la pestanya de Grups. I llistar els resultats pel grup seleccionat.

- Valorar un Punt de Pas, per Like, Unlike i Estrelles.
- Valorar una Ruta per Like, Unlike i Estrelles.

Repartiment de la feina.

Ha sigut una feina intensa, perquè no coneixiem l’ús del JavaFx.
Hem treballat plegats per entendre el codi, planificar la feina, fer servir els patrons, detectar malas olors.
Dividir exactament, tu fes això i tu fes això altre no. Perquè per aconseguir resultats tant per fer llistar valoracions com fer valoracions, implicava de les mateixes eines i llavors ho hem fet conjuntament.

Problemes amb tingut avui per pujar el projecte a GitHub, és per això que hi ha una mica de merder a els commits.