DISSENY DE SOFTWARE

Lliurament 2. SeriesTimeUB - Disseny i implementació

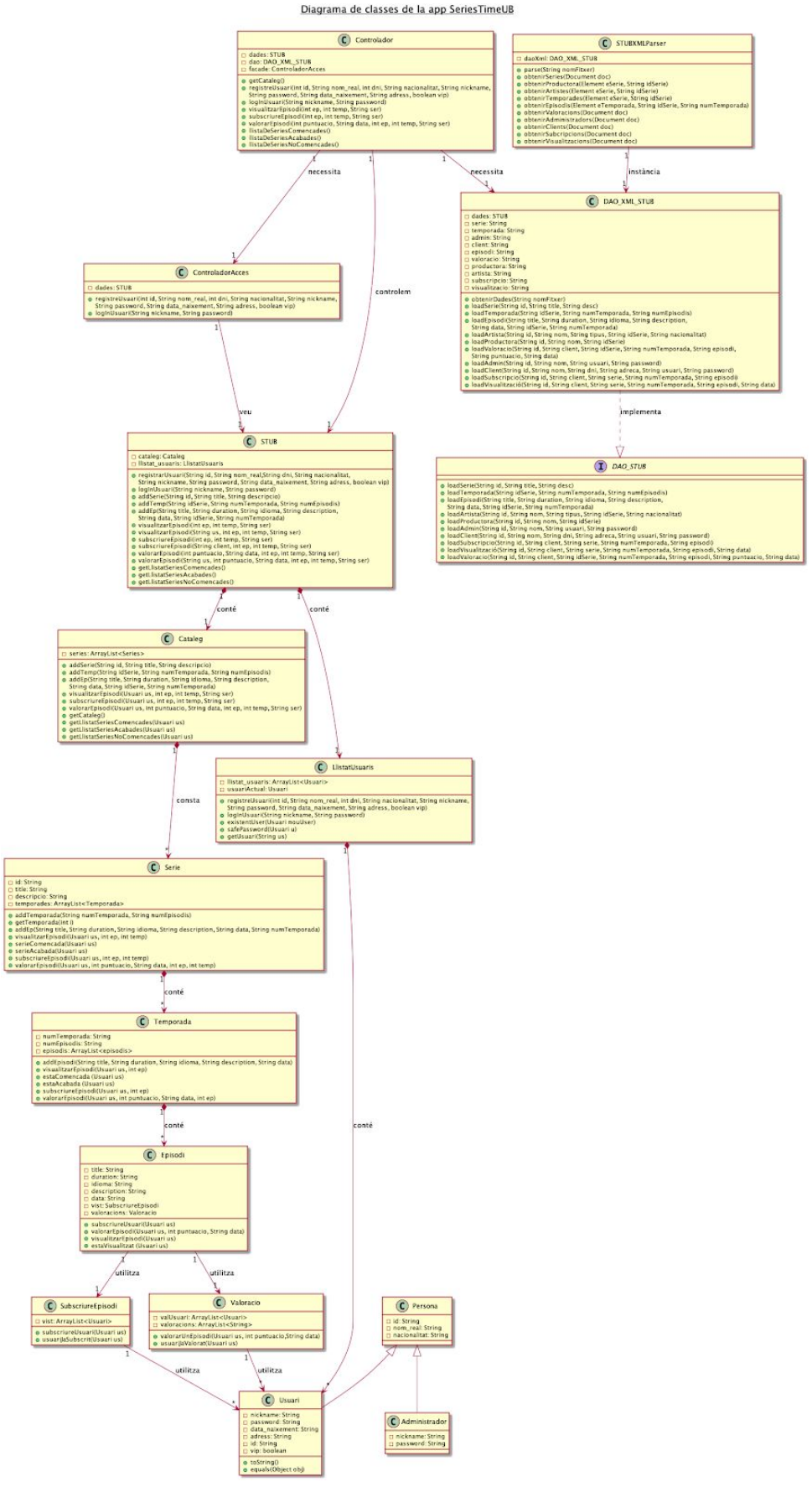
Joaquim Yuste Ramos, NIUB: 20081574
Laurentiu Nedelcu, NIUB: 20081585
Marcos Plaza Gonzàlez, NIUB: 20026915

ÍNDEX

Objectius

Els objectius d'aquest lliurament són bàsicament el aprendre a realitzar el disseny d'una aplicació (en el nostre cas, ***SeriesTimeUB***) iniciant desde la seva especificació, fent servir el patró ***TDD*** o ***Test Driven Development***. És a dir partirem dels tests d'acceptació (per tal de complir amb les funcionalitats requerides a l'enunciat) i anirem construint les nostres classes fixant-nos en els principis de disseny ***GRASP***, el patró arquitectònic ***Model-Vista-Controlador*** i altres patrons de disseny. Com a objectiu de disseny és vol aprendre a identificar els patrons més adients i la seva aplicació en un problema concret.

Model de classes de l'aplicació SeriesTimeUB



Comentari del Model de Classes de l'aplicació SeriesTimeUB

En l'anterior diagrama generat a partir d'un codi *UML*, es detalla cada classe, tant de *Model*, com de *Controlador*, com de *Resources*, s'ha especificat cada atribut de la classe i cada mètode a més de la visibilitat de cadascun d'aquests elements.

Es pot apreciar la relació que hi ha entre les diferents classes, és a dir, quines necessiten veure a d'altres per invocar mètodes o bé per processar dades, quines hereten d'altres classes, quines implementen interfícies, etc.

El codi en *plant UML*, és el següent:

```
@startuml
title __Diagrama de classes de la app SeriesTimeUB__\n

class Controlador{
-dades: STUB
-dao: DAO_XML_STUB
-facade: ControladorAcces
+getCataleg()
+registreUsuari(int id, String nom_real, int dni, String nacionalitat, String nickname,
String password, String data_naixement, String adress, boolean vip)
+logInUsuari(String nickname, String password)
+visualitzarEpisodi(int ep, int temp, String ser)
+subscriureEpisodi(int ep, int temp, String ser)
+valorarEpisodi(int puntuacio, String data, int ep, int temp, String ser)
+llistaDeSeriesComencades()
+llistaDeSeriesAcabades()
+llistaDeSeriesNoComencades()
}

class ControladorAcces{
-dades: STUB
+registreUsuari(int id, String nom_real, int dni, String nacionalitat, String nickname,
String password, String data_naixement, String adress, boolean vip)
+logInUsuari(String nickname, String password)
}
class Administrador{
-nickname: String
-password: String
}

class Cataleg{
-series: ArrayList<Series>
+addSerie(String id, String title, String descripcio)
+addTemp(String idSerie, String numTemporada, String numEpisodis)
```

+addEp(String title, String duration, String idioma, String description,
String data, String idSerie, String numTemporada)
+visualitzarEpisodi(Usuari us, int ep, int temp, String ser)
+subscriureEpisodi(Usuari us, int ep, int temp, String ser)
+valorarEpisodi(Usuari us, int puntuacio, String data, int ep, int temp, String ser)
+getCataleg()
+getLlistatSeriesComencades(Usuari us)
+getLlistatSeriesAcabades(Usuari us)
+getLlistatSeriesNoComencades(Usuari us)
}

class Episodi{
-title: String
-duration: String
-idioma: String
-description: String
-data: String
-vist: SubscriureEpisodi
-valoracions: Valoracio
+subscriureUsuari(Usuari us)
+valorarEpisodi(Usuari us, int puntuacio, String data)
+visualitzarEpisodi(Usuari us)
+estaVisualitzat (Usuari us)
}

class LlistatUsuaris{
-llistat_usuaris: ArrayList<Usuari>
-usuariActual: Usuari
+registreUsuari(int id, String nom_real, int dni, String nacionalitat, String nickname,
String password, String data_naixement, String adress, boolean vip)
+logInUsuari(String nickname, String password)
+existentUser(Usuari nouUser)
+safePassword(Usuari u)
+getUsuari(String us)
}

class Persona{
-id: String
-nom_real: String
-nacionalitat: String
}

class Serie{
-id: String
-title: String
-descripcio: String
-temporades: ArrayList<Temporada>
+addTemporada(String numTemporada, String numEpisodis)
+getTemporada(int i)
+addEp(String title, String duration, String idioma, String description, String data, String numTemporada)
+visualitzarEpisodi(Usuari us, int ep, int temp)
+serieComencada(Usuari us)
+serieAcabada(Usuari us)
+subscriureEpisodi(Usuari us, int ep, int temp)

+valorarEpisodi(Usuari us, int puntuacio, String data, int ep, int temp)
}

class STUB{
-cataleg: Cataleg
-llistat_usuaris: LlistatUsuaris
+registrarUsuari(String id, String nom_real,String dni, String nacionalitat,
String nickname, String password, String data_naixement, String adress, boolean vip)
+logInUsuari(String nickname, String password)
+addSerie(String id, String title, String descripcio)
+addTemp(String idSerie, String numTemporada, String numEpisodis)
+addEp(String title, String duration, String idioma, String description,
String data, String idSerie, String numTemporada)
+visualitzarEpisodi(int ep, int temp, String ser)
+visualitzarEpisodi(String us, int ep, int temp, String ser)
+subscriureEpisodi(int ep, int temp, String ser)
+subscriureEpisodi(String client, int ep, int temp, String ser)
+valorarEpisodi(int puntuacio, String data, int ep, int temp, String ser)
+valorarEpisodi(String us, int puntuacio, String data, int ep, int temp, String ser)
+getLlistatSeriesComencades()
+getLlistatSeriesAcabades()
+getLlistatSeriesNoComencades()
}

class SubscriureEpisodi{
-vist: ArrayList<Usuari>
+subscriureUsuari(Usuari us)
+usuariJaSubscrit(Usuari us)
}

class Temporada{
-numTemporada: String
-numEpisodis: String
-episodis: ArrayList<episodis>
+addEpisodi(String title, String duration, String idioma, String description, String data)
+visualitzarEpisodi(Usuari us, int ep)
+estaComencada (Usuari us)
+estaAcabada (Usuari us)
+subscriureEpisodi(Usuari us, int ep)
+valorarEpisodi(Usuari us, int puntuacio, String data, int ep)
}

class Usuari{
-nickname: String
-password: String
-data_naixement: String
-adress: String
-id: String
-vip: boolean
+toString()
+equals(Object obj)
}

class Valoracio{

-valUsuari: ArrayList<Usuari>
-valoracions: ArrayList<String>
+valorarUnEpisodi(Usuari us, int puntuacio,String data)
+usuariJaValorat(Usuari us)
}

interface DAO_STUB{
+loadSerie(String id, String title, String desc)
+loadTemporada(String idSerie, String numTemporada, String numEpisodis)
+loadEpisodi(String title, String duration, String idioma, String description,
String data, String idSerie, String numTemporada)
+loadArtista(String id, String nom, String tipus, String idSerie, String nacionalitat)
+loadProductora(String id, String nom, String idSerie)
+loadAdmin(String id, String nom, String usuari, String password)
+loadClient(String id, String nom, String dni, String adreca, String usuari, String password)
+loadSubscripcio(String id, String client, String serie, String numTemporada, String episodi)
+loadVisualització(String id, String client, String serie, String numTemporada, String episodi, String data)
+loadValoracio(String id, String client, String idSerie, String numTemporada, String episodi, String puntuacio, String data)
}

class DAO_XML_STUB{
-dades: STUB
-serie: String
-temporada: String
-admin: String
-client: String
-episodi: String
-valoracio: String
-productora: String
-artista: String
-subscripcio: String
-visualitzacio: String
+obtenirDades(String nomFitxer)
+loadSerie(String id, String title, String desc)
+loadTemporada(String idSerie, String numTemporada, String numEpisodis)
+loadEpisodi(String title, String duration, String idioma, String description,
String data, String idSerie, String numTemporada)
+loadArtista(String id, String nom, String tipus, String idSerie, String nacionalitat)
+loadProductora(String id, String nom, String idSerie)
+loadValoracio(String id, String client, String idSerie, String numTemporada, String episodi,
String puntuacio, String data)
+loadAdmin(String id, String nom, String usuari, String password)
+loadClient(String id, String nom, String dni, String adreca, String usuari, String password)
+loadSubscripcio(String id, String client, String serie, String numTemporada, String episodi)
+loadVisualització(String id, String client, String serie, String numTemporada, String episodi, String data)
}

class STUBXMLParser{
-daoXml: DAO_XML_STUB
+parse(String nomFitxer)
+obtenirSeries(Document doc)
+obtenirProductora(Element eSerie, String idSerie)
+obtenirArtistes(Element eSerie, String idSerie)

```
+obtenirTemporades(Element eSerie, String idSerie)
+obtenirEpisodis(Element eTemporada, String idSerie, String numTemporada)
+obtenirValoracions(Document doc)
+obtenirAdministradors(Document doc)
+obtenirClients(Document doc)
+obtenirSubcripcions(Document doc)
+obtenirVisualitzacions(Document doc)
}

STUB "1" *--> "1" Cataleg: conté
STUB "1" *--> "1" LlistatUsuaris: conté
Cataleg "1" *--> "*" Serie: consta
LlistatUsuaris "1" *--> "*" Usuari: conté
Serie "1" *--> "*" Temporada: conté
Temporada "1" *--> "*" Episodi: conté
Controlador “1”-->”1” STUB: controlem
Controlador “1”-->”1” DAO_XML_STUB: necessita
Controlador “1”-->”1” ControladorAcces: necessita
ControladorAcces “1”-->”1” STUB: veu
Persona <|-- Administrador
Persona <|-- Usuari
Episodi “1”-->”1” SubscriureEpisodi: utilitza
Episodi “1”-->”1” Valoracio: utilitza
SubscriureEpisodi “1”-->”*” Usuari: utilitza
Valoracio “1”-->”*” Usuari: utilitza
DAO_XML_STUB ..|> DAO_STUB: implementa
STUBXMLParser “1”-->”1” DAO_XML_STUB: instància
@enduml
```

[Separació Model-Vista-Controlador?/ Cal incloure la classe ArrayList?/ Associacions ben fetes?/ Sentit de les assosiacions?]

Observacions i decisions de disseny

Inicialment, la primera observació a destacar, és la utilització del patró arquitectònic *Model-Vista-Controlador* per tal d'articular tota la nostra aplicació, ja que és així com es demana a l'enunciat d'aquest lliurament. Tot i així cal dir que només s'ha desenvolupat la el *Model* i el *Controlador*, ja que de moment la *Vista* està implementada en els test d'acceptació del directori *test*.

Si parlem en termes de patró per capes hauríem desenvolupat la capa de *lògica de control* (on es veurien inclosos el *Model* i el *Controlador*), i la presentació seria l'equivalent a la nostra vista on trobem els *test d'acceptació*. A la capa de *persistencia* trobarem el fitxer amb informació per a fer funcionar l'aplicació del nostre segon lliurament anomenat *StUB.xml*.

Aprofitant que acabem d'anomenar l'existència del fixer continent de la informació *StUB.xml* proporcionat al *Campus Virtual*, hem fet servir el patró *DAO (Data Access Object)* per tal d'accedir a la *capa de Recursos* desde la *capa de la lógica de control*. El patró *DAO* consta de tres elements bàsics; la interficie on es defineixen les operacions stàndard que s'han de fer en els objectes del *Model* (en el nostre cas la coneixem com *DAO_STUB*), la classe d'implementació on realitzem la implementació concreta (*DAO_XML_STUB*) sobre la base de dades, i per últim l'objecte d'on extreurem les dades (*l'StUB.xml en questió*).

Hem aprofitat la pràctica per a veure com és el construïr una aplicació mitjançant el mètode *TDD* o *Test Driven Development*, el que vol dir que per a cada funcionalitat demanada pel client, abans d'implementar-la havíem de crear els *criteris d'acceptació* per a cadascuna. Plantejàvem una *història d'usuari* i a continuació ens feiem preguntes de com havia de respondre el nostre sistema. Va ser així com van anar creant els *criteris d'acceptació*. A continuació implementàvem el nostre codi per tal de satisfer amb éxit els nostres tests. Ens hem fixat en el nostre *model de domini* (entregat a l'anterior pràctica) i hem anat construïnt a partir d'aquí, tot procurant no

violar cap principi *S.O.L.I.D* i mirant de respectar i seguir els diferents patrons de disseny, com el *GRASP* (per a l'assignació de responsabilitats) o arquitectònics. Hem intentat que al nostre projecte, cada classe tingui una responsabilitat/propòsit específic i pensem que ho hem complert amb éxit. Feiem refactor del codi, i miràvem que les funcionalitats estiguessin ben implementades per tal de passar tots els test d'acceptació. Aquest procés el repetim fins a acabar totes les funcionalitats requerides.

En el moment que vam haver de fer el *Controlador*, vam "splitar" aquesta classe en un altre, ja que aquesta tenia delegada diverses responsabilitats i s'estava fent una classe bastant gran. Per això vam haver d'aplicar el patró de disseny *FAÇANA* tot creant una nova classe *ControladorAcces*, que s'encarregaría de registrar i loguejar als clients dins l'aplicació *STUB*. A Controlador ens va caldre instanciar un objecte *ControladorAcces* per tal d'accedir als seus mètodes mitjançant la sobrecàrrega d'aquests. Tot i així cal apreciar que d'aquesta manera estem violant un dels principis *S.O.L.I.D* (*Single responsability*) ja que *Controlador* es qui s'encarregarà en realitat de la crida dels mètodes per a registrar i loguejar.

Ens agradaría remarcar, per acabar, que tot i que pensem que el nostre codi té un baix acoblament, ja que podem entendre fàcilment el propòsit de cada classe, pot tendir a que cadascuna d'aquestes classes puguin no tenir una alta cohesió, ja que hi ha un fort grau de dependècia entre els mètodes de les classes. De cara a la pròxima entrega ens agradaría millorar tant aquests aspectes, com els que la professora ens exposi de manera que puguem entendre els nostres errors.

Conclusions

Durant el desenvolupament d'aquesta segona pràctica hem après a realitzar el disseny d'una aplicació a partir de la seva especificació utilitzant la metodologia *TDD (Test Driven Development)*.

Hem hagut de corregir el nostre *Model de Domini* ja que en la pràctica anterior teníem algunes idees errònies. I per a poder implementar aquesta pràctica ha sigut necessari modificar-lo, ja que ens facilita comprendre la relació que hi ha entre les diferents classes i així hem pogut identificar les classes que calien tindre en compte per a fer cada test d'acceptació.

També hem posat en pràctica els principis de disseny GRASP, el patró arquitectònic Model-Vista-Controlador i patrons de disseny que havíem vist en classe de teoria com hem pogut veure amb anterioritat. D'aquesta manera podem comprovar si realment hem entès aquests conceptes. A més, hem intentat seguir els principis S.O.L.I.D.

A més, ens estem acostumant a treballar amb el github, i trobem que això és un gran benefici per a nosaltres ja que ens facilita el treball en equip alhora de treballar conjuntament mitjançant la utilització d'un repositori privat per als membres de l'equip.