DISSENY DE SOFTWARE

Lliurament 3. SeriesTimeUB

Iteració 2 - Disseny i implementació

Realitzat per

Joaquim Yuste Ramos, NIUB: 20081574

Laurentiu Nedelcu, NIUB: 20081585

Marcos Plaza Gonzàlez, NIUB: 20026915

ÍNDEX

Objectius 2

Model de classes de l'aplicació SeriesTimeUB 3

Comentari del Model de Classes de l'aplicació SeriesTimeUB 4

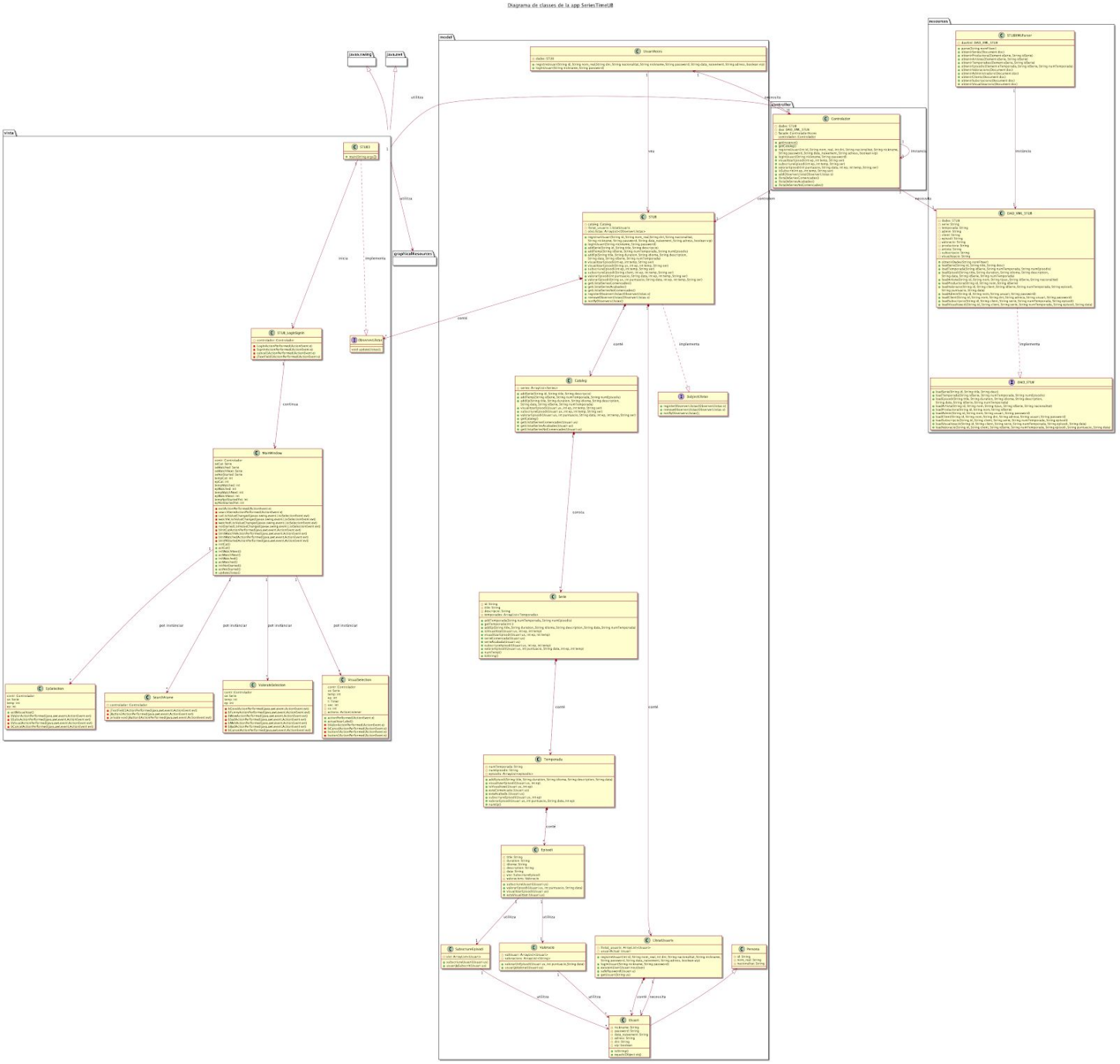
Observacions i decisions de disseny 9

Conclusions 11

Objectius

L'objectiu principal d'aquesta pràctica és aprendre l'aplicació de patrons de disseny en el desenvolupament d'una aplicació basada en interfícies gràfiques. El primer pas, ara que les funcionalitats ja estan testejades (en la pràctica 2), és afegir una GUI implementada amb Swing, SwingX o JavaFX tot seguint el patró Model-Vista-Controlador aplicant els principis de disseny GRASP, el patró arquitectònic Model-Vista-Controlador (aproximació general) i patrons de disseny. Com a objectiu de disseny s'aprendrà a identificar els patrons més adients i la seva aplicació .

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, aquest cop de manera on es mostra les relacions entre classes seguint el patró de disseny arquitectònic Model-Vista-Controlador, tot i que també entren en joc altres móduls com son el de resources o graphicalResources, necessàris per al nostre projecte.

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. Val a dir, però, que tant els atributs, és a dir, els components de cada frame utilitzat en el nostre projecte, com el mètode per inicialitzar els components (fer el new) no apareix en aquest diagrama de classes ja que hem considerat que ja que és una característica comú en totes les classes de la vista no calia afegir-ho per que a més dificulta l'enteniment del diagrama.

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

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

package "controller" {
  class Controlador{
  -dades: STUB
  -dao: DAO_XML_STUB
  -facade: ControladorAcces
  {static} controlador: Controlador
  +{static} getInstance()
  +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)
  +isSubscrit(int ep, int temp, String ser)
  +addObserverLlista(ObserverLlistas o)
  +llistaDeSeriesComencades()
  +llistaDeSeriesAcabades()
  +llistaDeSeriesNoComencades()
  }
```

```
}

package "model"{
  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)
+isVisualitzat(Usuari us, int ep, int temp)
+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)
+numTemp()
+toString()
}

class STUB{
-cataleg: Cataleg
-llistat_usuaris: LlistatUsuaris
-obsLlistas: ArrayList<ObserverLlistas>
+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()
+registerObserverLlistas(ObserverLlistas o)
+removeObserverLlistas(ObserverLlistas o)
+notifyObserversLlistas()
}

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

interface SubjectLlistas{
+registerObserverLlistas(ObserverLlistas o)
+removeObserverLlistas(ObserverLlistas o)
+notifyObserversLlistas();
}

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)
  +isVisualitzat(Usuari us, int ep)
  +estaComencada (Usuari us)
  +estaAcabada (Usuari us)
  +subscriureEpisodi(Usuari us, int ep)
  +valorarEpisodi(Usuari us, int puntuacio, String data, int ep)
  +numEp()
  }

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

  class UsuariAcces{
  -dades: STUB
   +registreUsuari(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)
  }

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

package "resources"{
  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)
  }
}

package "vista"{
  class EpSelection{
  contr: Controlador
  se: Serie
  temp: int
  ep: int
  +actBVisualitzat()
  -bValorActionPerformed(java.awt.event.ActionEvent evt)
  -bSubsActionPerformed(java.awt.event.ActionEvent evt)
  -bVisualActionPerformed(java.awt.event.ActionEvent evt)
  -bCancelActionPerformed(java.awt.event.ActionEvent evt)
  }
```

class MainWindow{
contr: Controlador
seCat: Serie
seWatched: Serie
seWatchNext: Serie
seNotStarted: Serie
tempCat: int
epCat: int
tempWatched: int
epWatched: int
tempWatchNext: int
epWatchNext: int
tempNotStartedYet: int
epNotStartedYet: int
-exitActionPerformed(ActionEvent e)
-searchSerieActionPerformed(ActionEvent e)
-catListValueChanged(javax.swing.event.ListSelectionEvent evt)
-watchNListValueChanged(javax.swing.event.ListSelectionEvent evt)
-watchedListValueChanged(javax.swing.event.ListSelectionEvent evt)
-notStartedListValueChanged(javax.swing.event.ListSelectionEvent evt)
-bInitCatActionPerformed(java.awt.event.ActionEvent evt)
-bInitWatchNActionPerformed(java.awt.event.ActionEvent evt)
-bInitWatchedActionPerformed(java.awt.event.ActionEvent evt)
-bInitNStartedActionPerformed(java.awt.event.ActionEvent evt)
+initCat()
+actCat()
+initWatchNext()
+actWatchNext()
+initWatched()
+actWatched()
+initNotStarted()
+actNotStarted()
+updateLlistas()
}

interface ObserverLlistas{
void updateLlistas();
}

class SearchFrame{
-controlador: Controlador
-jTextField1ActionPerformed(java.awt.event.ActionEvent evt)
-jButton2ActionPerformed(java.awt.event.ActionEvent evt)
-private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
}

class STUB3{
+{static}main(String args[])
}

class STUB_LogInSignIn{
-controlador: Controlador
-LogInActionPerformed(ActionEvent e)
-SignInActionPerformed(ActionEvent e)

```
  -cancel1ActionPerformed(ActionEvent e)
  -jTextField1ActionPerformed(ActionEvent e)
  }

  class ValorateSelection{
  contr: Controlador
  se: Serie
  temp: int
  ep: int
  -bGoodActionPerformed(java.awt.event.ActionEvent evt)
  -bFunnyActionPerformed(java.awt.event.ActionEvent evt)
  -bWowActionPerformed(java.awt.event.ActionEvent evt)
  -bSadActionPerformed(java.awt.event.ActionEvent evt)
  -bMehActionPerformed(java.awt.event.ActionEvent evt)
  -bBadActionPerformed(java.awt.event.ActionEvent evt)
  -bCancelActionPerformed(java.awt.event.ActionEvent evt)
  }

  class VisualSelection{
  contr: Controlador
  se: Serie
  temp: int
  ep: int
  t: Timer
  -sec: int
  -cs: int
  -actions: ActionListener
  +actionPerformed(ActionEvent e)
  +actualitzarLabel()
  -bValorActionPerformed(ActionEvent e)
  -bCancelActionPerformed(ActionEvent e)
  -button1ActionPerformed(ActionEvent e)
  -button2ActionPerformed(ActionEvent e)
  }
}

package "javax.swing"{}
package "java.awt"{}
package "graphicalResources"{}

STUB "1" *--> "1" Cataleg: conté
STUB "1" *--> "1" LlistatUsuaris: conté
STUB "1" *--> "*" ObserverLlistas: conté
STUB ..|> SubjectLlistas: implementa
Cataleg "1" *--> "*" Serie: consta
LlistatUsuaris "1" *--> "*" Usuari: conté
LlistatUsuaris "1" --> "1" Usuari: necessita
Serie "1" *--> "*" Temporada: conté
Temporada "1" *--> "*" Episodi: conté
Controlador "1" --> "1" STUB: controlem
Controlador "1" --> "1" DAO_XML_STUB: necessita
Controlador "1" --> "1" UsuariAcces: necessita
Controlador "1"-->"1" Controlador: instancia
UsuariAcces "1" --> "1" STUB: veu
```

```
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
vista "1" --> "1" Controlador: utilitza
vista --> graphicalResources: utilitza
javax.swing <|-- vista
java.awt <|-- vista
STUB3 "1" --> "1" STUB_LogInSignIn: inicia
STUB3 ··|> ObserverLlistas: implementa
STUB_LogInSignIn "1" --> "1" MainWindow: continua
MainWindow "1" --> "*" EpSelection: pot instànciar
MainWindow "1" --> "*" SearchFrame: pot instànciar
MainWindow "1" --> "*" ValorateSelection: pot instànciar
MainWindow "1" --> "*" VisualSelection: pot instànciar
@enduml
```

Observacions i decisions de disseny

Inicialment, la primera observació a destacar, és que hem desenvolupat la part de la Vista, que és l'única part del patró arquitectònic *Model-Vista-Controlador* que ens quedava per fer i la que es demana per a aquest lliurament.

En quant a la interfície gràfica, hem creat una finestra per tal de registrar i *loguejar* usuaris. Encara que la part de registrar usuaris no està implementada. Com que no era una part obligatòria del projecte, hem decidit no crear una altra finestra per dur a terme aquesta acció. En canvi, per a dur a terme la part de loguejar un usuari, hem creat un per defecte el qual es podrà saber el seu nom i *password* quan es vagi a inicialitzar l'aplicació.

Una vegada l'usuari s'hagi loguejat, s'obrirà una altra finestra on es podran observar les sèries del catàleg, *watchedList*, *watchNext* i *notStartedYet.* Quan l'usuari hagi seleccionat una sèrie, podrà veure les temporades i seguidament els episodis. Per a cada episodi l'usuari podrà visualitzar-lo i llavors es mostrarà en una altra finestra un *timer*. Una altra opció per als episodis es valorar-lo, on també es mostrarà una diferent finestra amb opcions per valorar. I una altra per subscriure's a ell. En cada cas se li oferirà de poder tornar cap atràs.

En quant als patrons de disseny que hem usat en aquest projecte hem de recalcar que hem fet servir el patró *Observer* i el patró *Singleton*. El patró *Singleton* l'hem usat en la classe Controlador ja que si creem diversos objectes d'aquesta classe es podria omplir el heap del sistema amb objectes innecessaris que es creen cada vegada que es crea una nova instància de la classe. Per això si restringim la creació a un sol objecte es podria evitar problemes de rendiment i s'estalvia un gran treball per al *garbage collector*.
En un principi el patró *Singleton* no compleix el principi *Single Responsibility,* ja que aquest patró requereix d'una alta cohesió entre les classes del sistema, i per aquesta raó un simple canvi en el sistema requereix un canvi major en les classes

d'aquest. No obstant això, el problema es pot resoldre si es passa l'objecte per paràmetres. D'aquesta manera es redueix l'acoplament entre les classes del sistema.

El principi *Open Closed* es vulnera. Per una part compleix que no es pugui modificar la classe. Això és dóna perquè en el *Singleton,* es necessita crear constructors privats per a que cap altre programa o funció sigui capaç de generar objectes d'aquesta classe, que fan servir variables privades abans definides per a poder emmagatzemar referències a l'objecte que es crea a través del constructor. Però, per l'altra part no ho fa ja que aquest principi controla el punt d'accés i tan sols es retornarà a n'ell mateix, no una extensió.

El principi de la Segregació d'Interfície és l'única que no trenca directament ja que el *Singleton* no permet començar una interfície.

No vulnerem el principi de substitució de Liskov ja que no fem herències amb aquesta classe. No obstant això, si s'hagués produït alguna herencia no hagués complert aquest principi, ja que el simple fet de tindre diferents versions de l'objecte fa que no sigui un *Singleton* més.

I per acabar, el *Singleton* vulnera el principi d'inversió de dependència, ja que només pot dependre de la seva instància molt concreta.

Per aconseguir tenir la vista sempre actualitzada hem decidit utilitzar el patró observer, el que aconseguim fent servir aquest patró es cridar un mètode que actualitza la vista cada cop que les dades canvien d'una manera ràpida i amb un acoplamente entre l'observer i l'observat molt reduït.

Per implementar-ho, vam convertir la finestra principal encarregada de imprimir el llistat de series en Observer, i la clase més general del model la vem convertir en la Subject, per tant, cadascuna d'aquestes clases implementen els mètodes corresponents de les interfícies ObserverLlistas i SubjectLlistas respectivament.

Quan s'inicialitza el programa, aquest envia al controlador l'objecte de la vista, convertit per mitjà del polimorfisme en un objecte de classe ObserverLlista, al model, i aquest el desa en un llistat d'observers, cada cop que es crida a qualsevol metode que pot implicar un possible canvi en les dades que s'estan imprimint a la vista,

aquest crida al metode *notify()*, que a l'hora va cridant el metode *update()* de cadascun dels observers.

Conclusions

Durant el desenvolupament d'aquesta tercera i última pràctica hem après com aplicar patrons de disseny en el desenvolupament d'una aplicació basada en interfícies gràfiques.

Respecte al darrer lliurament, hem hagut de corregir alguns aspectes de la nostra pràctica que ens ha marcat la professora. Per exemple hem eliminat una de les classes del paquet Controlador i la hem combinat amb l'altra classe (*ControladorAcces).* En lloc d'usar aquesta classe hem aplicat el patró de disseny *Facade* en una altra classe anomenada *UsuariAcces* situada en el paquet model, que s'encarrega bàsicament de registrar i loguejar usuaris.

Després d'haver fet les correccions, hem esquematitzat com quedaria la nostra interfície gràfica. Que està explicada breument en l'apartat d'Observacions i decisions de disseny.

Més tard hem pensat com hauriem de desenvolupar la GUI a la pràctica. No obstant, gràcies a les propostes que ens ha donat la professora, hem arribat a la conclusió de que instalar el plugin *JFormDesigner* a Intellij era la millor opció.

En quant al github, hem hagut de ramificar el nostre projecte ja que així teniem l'opció de fer varies versions del nostre projecte sense perdre l'original. Encara que al final tan sols hem creat una sola versió d'interfície de la pràctica, creiem que això és molt útil en cas que haguéssim volgut crear-ne moltes més, podent partir del projecte que havíem entregat en el lliurament anterior. Ja que al crear varies versions tant nosaltres com el client podiem decidir quina de totes les vistes ens/li pareix que és la més apropiada respecte als nostres/seus gustos i desitjos.