Disseny i implementació d'un Videoclub *online* amb *J2EE* i una llibreria de *tags* de suport per la capa de presentació

Armand Berenguer Montaner
Enginyeria en Informàtica

Consultor: Josep Maria Camps Riba
Memòria PFC: 17/01/2011

Nomes volia agrair el suport per part del consultor de l'assignatura Josep Maria Camps Riba, doncs sempre que ha calgut ha donat un cop de mà revisant entregues que realitzava i aportant les seves correccions.

Resum

En aquest projecte "Disseny i implementació d'un Videoclub *online* amb *J2EE* i una llibreria de *tags* de suport per a la capa de presentació" es pretenen assolir diferents objectius que es poden separar en diferents etapes dependents cada una de les altres.

La primera etapa consisteix en obtenir el màxim de coneixements de l'arquitectura *J2EE* per al desenvolupament d'aplicacions i concretament conèixer les diferents capes de l'arquitectura i solucions així com els diferents *frameworks* existents escollint l'arquitectura que millor resolgui la problemàtica del videoclub *online*. Necessitem per tant realitzar una forta investigació sobre els diferents patrons de disseny, arquitectures existents *J2EE*, *frameworks* existents.

Posteriorment investigar sobre l'ús de *Custom Tag Libs* en la capa de presentació i el seu ús en les vistes de l'aplicació.

Un cop pressa la decisió de l'arquitectura realitzar el disseny e implementació de la llibreria de *tags* i la implementació parcial del Videoclub *online*.

Arribats a aquest punt s'haurà aconseguit un gran objectiu aprofundir en l'ús de tecnologies *J2EE* per a desenvolupament d'aplicacions fent ús de diferents *frameworks* i l'ús de llibreries de *tags* en les vistes de la capa de presentació i la seva posterior reutilització.

Paraules Clau

J2EE
Framework
Patró de disseny – Design Patterns
Llibreria de tags - Custom Tag Libs
Capa de presentació
Vistes

Índex

Índex de figures

Capítol 1: Introducció

Descripció del projecte

En aquest projecte es desenvoluparà parcialment la web per a la gestió de un Videoclub *online* aplicada al model de n capes de *J2EE* usant el *framework* més adient per a cada capa. A més en la capa de presentació es generarà una llibreria de *tags* que doni suport i faci més fàcil el desenvolupament de les vistes de la capa de presentació.

Abans de realitzar el projecte s'obtindrà informació sobre la millor arquitectura per a una aplicació *J2EE* d'aquestes característiques i els patrons de disseny aplicats , així com els diferents *frameworks* existents al mercat que les implementen. Tot seguit es justificarà la utilització dels *frameworks* escollits per a cada capa i s'investigarà sobre la realització de *tags* propis en la capa de presentació definint aquells a implementar.

En un videoclub *online* interactuen usuaris que consulten la informació de jocs i pel·lícules existents, socis que són aquells usuaris ja registrats i que poden realitzar préstecs i reserves d'exemplars. Per altra banda tenim els administradors que s'encarreguen de la gestió de la informació com donar d' alta les pel·lícules per a consultar-les i gestionar les seves reserves. En aquest cas utilitzarem *frameworks* per tal de desenvolupar de forma parcial aquest model de negoci. Per tant, es vol aprofitar tots els avantatges ens donen mitjançant el coneixement de la seva arquitectura i serveis que implementen. Tot això, integrant-los en el desenvolupament d'una aplicació e implementar nous *tags* que proporcionin noves funcionalitats en la capa de presentació.

Aquest projecte es pot dividir en 4 grans apartats o fases:

1. Estudi dels *frameworks*, patrons, arquitectures e integració de diferents *frameworks*.
2. Estudi de les llibreries de *tags.*
3. Anàlisi i disseny tant de l'aplicació com de llibreria de *tags.*
4. Implementació tant de la llibreria de *tags* com de i l'aplicació.

En la primera fase s'obtindrà informació i bibliografia dels diferents *frameworks* resumint aquells aspectes i característiques més importants. Per això es seleccionaran aquelles fonts d'informació més rellevants. Tindrem en compte informació com arquitectura usada en aplicacions *J2EE*, les diferents capes existents, els patrons de disseny aplicats, facilitat d'ús dels *frameworks*, acoblament dels *frameworks*, i decidir la utilitat dels *frameworks* o no.

En la segona fase es farà èmfasi en la importància del ús de llibreries de *tags* en la capa de presentació i els avantatges que comporten la seva utilització i com millorar el desenvolupament d'una aplicació generant una llibreria de *tags* pròpia aplicada al videoclub *online*. Així mateix es seleccionaran i justificaran aquells a implementar.

En la tercera fase un cop fet l'estudi de les llibreries de *tags* en la fase anterior, es realitzarà l'anàlisi i disseny del videoclub *online* "parcialment" i de la llibreria de *tags* de forma complerta.

Finalment en la quarta fase es realitzarà la implementació de totes les funcionalitats dissenyades tant de la llibreria de *tags* com de l'aplicació.
S'ha pensat en un model de dades multi-idioma i en la capa de presentació s'ha pensat en usar *Struts* sota el patró *MVC*, per a la capa de Negoci s'ha pensat en *Spring* i finalment per a la capa de dades s'ha pensat en usar *Hibernate*.

En quant al model de dades s'ha pensat en un model *MySQL* i en quant a servidor de l'aplicació s'ha pensat en *Tomcat 6.0* com a servidor web i contenidor de S*ervlets* i *JSP*.
En quant a eina de desenvolupament utilitzarem *Eclipse Galileo* i hem pensat que sigui una aplicació web compatible amb els navegadors *Internet Explorer* i *Mozilla Firefox* tot i que es recomana l'ús d'aquest últim.

Objectius generals i específics

Els objectius generals d'aquest projecte:

- Aprofundir en el coneixement de diferents *frameworks* i la utilització de llibreries de *tags* per a la capa de presentació .
- Conèixer com utilitzar els *frameworks* i les llibreries de *tags.*
- Integrar les diferents arquitectures per tal de desenvolupar una aplicació en aquest cas un Videoclub *online.*
- Conèixer com crear una llibreria de *tags* pròpia.

Els objectius específics:

- Desenvolupar aplicacions *J2EE* fent ús de *frameworks.*
- Integrar diferents *frameworks* en el desenvolupament d'aplicacions J2EE.
- Desenvolupar aplicacions *J2EE* fent ús de llibreries de *tags* en la capa de presentació.
- Comprovar els avantatges i mancances de les llibreries de *tags* i justificar la seva utilització davant de certes necessitats de les aplicacions.
- Desenvolupar llibreries de *tags* personalitzades.
- Desenvolupar nous *tags* que donin resposta a unes necessitats determinades en aquest cas per a un Videoclub *online.*
- Aplicar la implementació de la llibreria de *tags* personalitzada en el desenvolupament d'aplicacions *J2EE* en aquest cas un Videoclub *online*.
- Desenvolupar una metodologia i llibreria de *tags* genèrica que sigui exportable i aplicable a aplicacions que segueixin un model de negoci de Videoclub *online*.
- Descobrir amb el perfil de desenvolupador els avantatges d'utilitzar aquestes llibreries de *tags* personalitzades en les vistes de la capa de presentació.

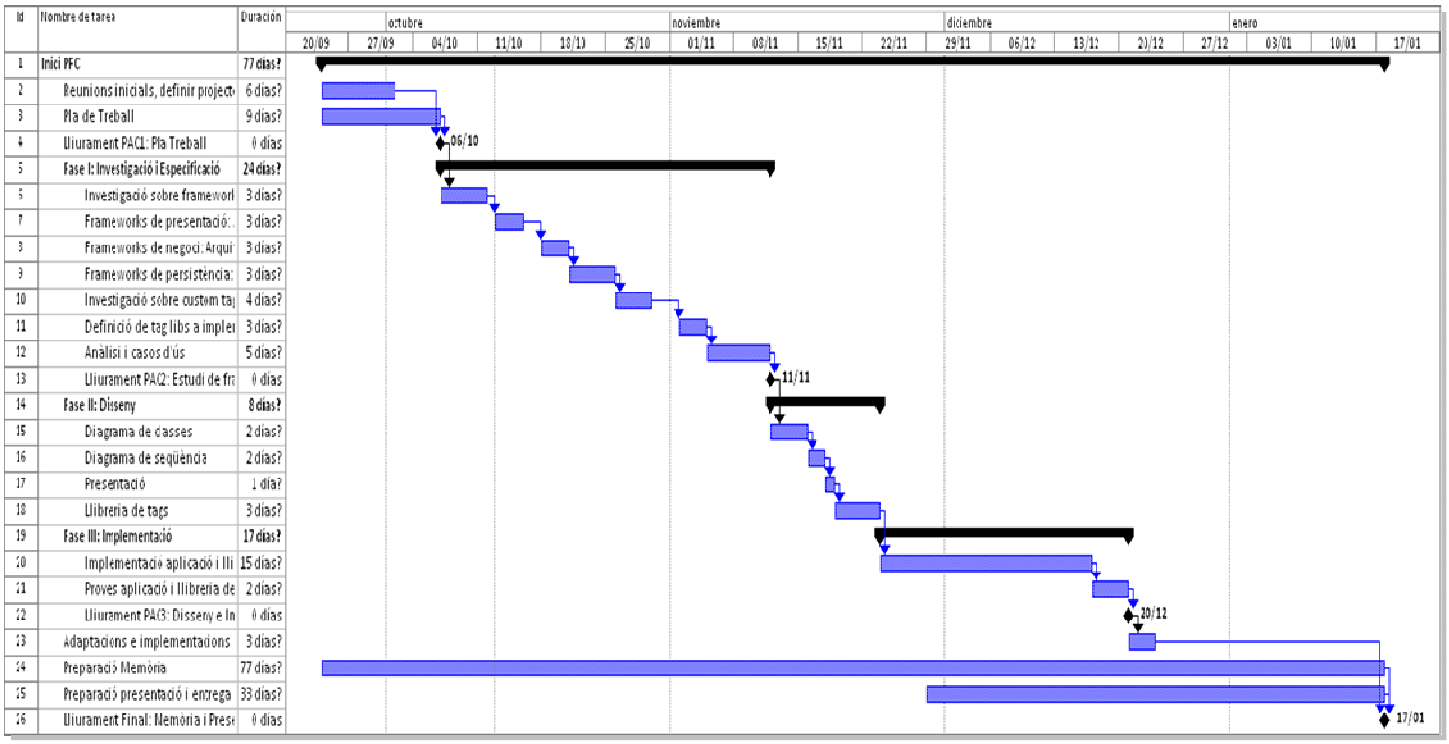
Planificació amb fites i temporització

Les principals fites coincidiran amb les entregues de les diferents PACs:

NOM FITA	DATA D'ENTREGA
Lliurament PAC1: Pla de Treball	06/10/2010
Lliurament PAC2: Estudi de *frameworks*, patrons, arquitectures i anàlisi	11/11/2010
Lliurament PAC3: Disseny e implementació	20/12/2010
Entrega Final: Memòria i presentació	17/01/2011

La planificació que tindrem per a les diferents tasques es dividirà seguint les dates que s'indiquen a continuació:

TASCA	DATA INICI	DATA FI
Pla de Treball	24/09/2010	06/10/2010
Investigació sobre *Frameworks*	07/10/2010	11/10/2010
Frameworks de presentació: Arquitectura i Patrons de disseny	13/10/2010	15/10/2010
Frameworks de negoci: Arquitectura i Patrons de disseny	18/10/2010	20/10/2010
Frameworks de persistència: Arquitectura i Patrons de disseny	21/10/2010	25/10/2010
Investigació sobre *Custom Tag Libs*	26/10/2010	29/10/2010
Definició de *Tag Libs* a implementar	2/11/2010	04/11/2010
Especificació: Anàlisi i casos d'ús	5/11/2010	11/11/2010
Disseny: Diagrama de classes	12/11/2010	15/11/2010
Disseny: Diagrama de seqüència	16/11/2010	17/11/2010
Disseny: Presentació	18/11/2010	18/11/2010
Disseny: Llibreria de *tags*	19/11/2010	23/11/2010
Implementació aplicació i llibreria de *tags*	24/11/2010	16/12/2010
Proves aplicació i llibreria de *tags*	17/12/2010	20/12/2010
Adaptacions implementació pendents	21/12/2010	23/12/2010
Preparació Memòria	24/09/2010	17/01/2011
Preparació entrega i presentació	29/11/2010	17/01/2011


Id
Nombre de tarea
Duración
octubre
noviembre
diciembre
enero
20/09
27/09
04/10
11/10
18/10
25/10
01/11
08/11
15/11
22/11
29/11
06/12
13/12
20/12
27/12
03/01
10/01
17/01
1 Inici PFC 77 días?
2 Reunions inicials, definir projecte 6 días?
3 Pla de Treball 9 días?
4 Lliurament PAC1: Pla Treball 0 días
5 Fase I: Investigació i Especificació 24 días?
6 Investigació sobre framework 3 días?
7 Frameworks de presentació: 3 días?
8 Frameworks de negoci: Arqui 3 días?
9 Frameworks de persistència: 3 días?
10 Investigació sobre custom tag 4 días?
11 Definició de taglibs a imple 3 días?
12 Anàlisi i casos d'ús 5 días?
13 Lliurament PAC2: Estudi de fra 0 días
14 Fase II: Disseny 8 días?
15 Diagrama de classes 2 días?
16 Diagrama de seqüència 2 días?
17 Presentació 1 día?
18 Llibreria de tags 3 días?
19 Fase III: Implementació 17 días?
20 Implementació aplicació i lli 15 días?
21 Proves aplicació i llibreria de 2 días?
22 Lliurament PAC3: Disseny e Im 0 días
23 Adaptacions e implementacions 3 días?
24 Preparació Memòria 77 días?
25 Preparació presentació i entrega 33 días?
26 Lliurament Final: Memòria i Prese 0 días
06/10
11/11
20/12
17/01

Enfocament i mètode seguit

Partint de la investigació de les diferents arquitectures *J2EE, frameworks* i Patrons de disseny es vol prendre la decisió d'escollir la millor arquitectura que s'adapti a les necessitats del Videoclub *online,* per tal de realitzar la implementació parcial.
També es considera l'ús de llibreries de *tags* en les vistes i la realització de una llibreria de *tags* pròpia per a realitzar el desenvolupament del Videoclub *online* que es pugui exportar a altres aplicacions que facin servir un model de negoci similar.

El mètode seguit es pot dividir en aquestes etapes:

Investigació de l'arquitectura *J2EE*, Patrons de disseny i *Frameworks.*

Decisió sobre l'arquitectura *J2EE* (*EJB* no *EJB*).

Selecció de Patrons de Disseny a utilitzar en la implementació.

Investigació de *Frameworks* a la capa de presentació centrant en *Struts.*

Investigació de *Frameworks* en la capa de negoci centrant en *Spring.*

Investigació de *Frameworks* en la capa de integració centrant en *Hibernate.*

Investigació de la Integració dels diferents *Frameworks.*

Investigació sobre els *Custom Tag Libs* i selecció de *tags* a implementar.

Construir parcialment el Videoclub *online* fent ús de *Struts, Spring Hibernate.*

Implementar una llibreria de *tags* per treballar amb les Vistes del Videoclub *online.*

Provar la implementació.

Abstreure la llibreria i model per utilitzar-les amb altres models de negoci similars.

Productes Obtinguts

S'han obtingut els següents productes o resultats com a desenvolupament del PFC:

Memòria (aberenguer78_memoria.pdf): reflecteix el treball desenvolupat al llarg del projecte.
Presnetació (aberenguer78_presentació.ppt): presentació de 20 *slides* que resumeix el projecte i el presenta.
Producte (aberenguer78_producte.zip): Aquet fitxer conté diversos fitxers alhora comprimits que a continuació exposem:

VideoclubOnline.war: Aquest fitxer es el "*Web Application Archiver*" preparat per ser desplegat en un *Tomcat*, només haurem de tenir en compte incloure les dependencies descrites al Annex.
vcotaglib_1_0.jar: Aquest fitxer *jar* conté la llibreria de tags desenvolupada per el projecte.
vcopagertaglib_1_0.jar: Aquest fitxer *jar* conté la llibreria de *tags* adaptada i a ser utilitzada amb la llibreria vcotaglib.
vcotaglib-src.zip: Aquest fitxer conte el codi font de la llibreira vcotaglib_1_0.jar
vcopagertaglib-src.zip: Aquest fitxer conté el codi font de la llibreria vcopagertaglib_1_0.jar
video_club.sql.zip: Aquest fitxer conté la BD amb els exemples e informació mostrada.

NOTA: Posteriorment als annexes és facilita més informació sobre la instal·lació i les llibreries de *tags*. Els 2 *jars* de les llibreries de *tags* anteriors s'han d'incloure directament a la carpeta **WEB-INF/lib** del projecte com la resta de llibreries.

Descripció de la resta de capítols

Capítol 2: Estudi sobre l'arquitectura de l'aplicació i dels *frameworks*.

Consisteix en realitzar un estudi e investigació de l'arquitectura, patrons de disseny existents existents per a cada capa de l'arquitectura *J2EE* i *frameworks* a utilitzar per al desenvolupament parcial del Videoclub *online*. Per tal de realitzar-lo s'ha d'investigar sobre les diferents solucions i alternatives existents escollint aquelles més adients per al Videoclub *online*, de la mateixa manera amb els *frameworks* es revisaran els avantatges e inconvenients i s'escolliran aquells que s'ajustin més a les necessitats del projecte.

Capítol 3: Investigació sobre les *Custom Tag Libs* (llibreries de *tags*) i justificació de *tags* a implementar.

Consisteix en realitzar un estudi e investigació sobre l'ús de les llibreries de *tags* en les vistes de la capa de presentació. Per una part estudiarem l'arquitectura d'aquestes llibreries i per altre estudiarem les llibreries existents *Standard JSTL i Struts Tag libs*, així com la justificació d'aquells *tags* a implementar per el model de Videoclub *online*.

Capítol 4: Anàlisi i disseny parcial del Videoclub online i llibreria de tags.

Consisteix en realitzar el anàlisi i disseny de l'aplicació per tal de poder realitzar la implementació fent ús dels *frameworks* escollits anteriorment, concretament *Struts*, *Spring, Hibernate*. Apart es desenvolupa un anàlisi basat en components dels diferents components a desenvolupar fent us dels Patrons de Disseny seleccionats i també dels components de la llibreria de tags.

Capítol 5: Implementació.

Consisteix en la implementació tant del Videoclub *online* com de la llibreria de *tags* fent èmfasi en el ús dels *frameworks* escollits en cada una de les capes de l'aplicació. És mostra el codi generat i les pantalles resultants.

Capítol 6: Conclusions.

Analitzarem quins coneixements i experiència s'ha assolit en la el·laboració del projecte integrant els diferents *frameworks* i desenvolupant una llibreria de *tags* pròpia per a les vistes de la capa de presentació.

Capítol 2: Estudi sobre l'arquitectura de l'aplicació i dels frameworks

Introducció

Abans de començar el projecte s'ha d'investigar quina és la millor arquitectura a aplicar per aquest projecte. Per tal de fer-ho prèviament s'haurà de revisar que ens proposa l'arquitectura de *J2EE* en el seu estàndard.

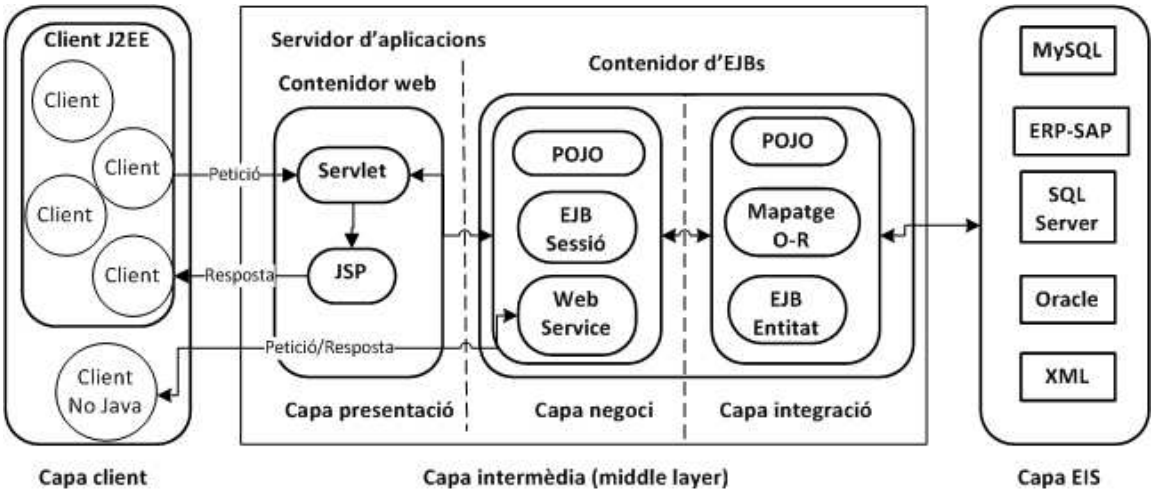


Figura 1. Arquitectura n-capes J2EE

Figura 1.- Arquitectura n-capes J2EE

Com es pot veure en el gràfic, en l'Arquitectura *J2EE* hi ha una clara diferenciació en 3 capes principals:

- **Capa client:** Aquesta capa conté els components o clients que s'executen a les màquines client realitzant les diferents peticions. Aquests poden ser diferents tipus de clients (*Thin Client, PDA*, Mòbils, *MIDlets*, Aplicacions d'escriptori) tots ells clients *J2EE* o clients que siguin aplicacions no Java.
- **Capa intermèdia (middle layer):** Conté la lògica de negoci i separa la capa client de la capa *EIS*. Aquesta es pot subdividir en 3 subcapes formades per diferents components.
    - **Capa de presentació:** Conté la lògica de interacció entre el usuari i l'aplicació i s'executa en el contenidor web.
    - **Capa de negoci:** Conté les regles de negoci de l'aplicació i els components de negoci a utilitzar. Poden ser *EJB*, classes *POJO*, o *Web Services* que interactuen amb clients no java.
    - **Capa d'integració:** Integració de l'aplicació amb els diferents sistemes gestors de la informació a interactuar. Es poden utilitzar *EJB* d'entitat classes *POJO* o Mapatge O-R.

- **Capa *EIS*:** Conté els components de la capa de dades que s'emmagatzemen de forma persistent en el sistema gestor de dades (MySQL, Oracle, SQL Server, ERP-SAP, XML...).

Decisió sobre l'arquitectura *J2EE* a implementar

Ara que ja s'ha vist que proposa l'arquitectura *J2EE* s'ha de decidir quina és la millor arquitectura per implementar l'aplicació. Cal recordar que per al nostre sistema la millor arquitectura serà aquella més simple que compleixi amb els requeriments. Per tal de fer-ho, hi ha una sèrie de qüestions importants a resoldre.

1. ¿ Quin tipus de clients interactuaran amb l'aplicació en la capa client ?

L'aplicació és una aplicació web per a la gestió de un Videoclub *online* on els clients que podran ser: (usuaris, socis o administradors de la web) utilitzaran clients web "*thin client*" (concretament serà compatible amb navegadors de última generació *Internet Explorer 8.0 i Mozilla Firefox 3.0*).

2. ¿ La capa de negoci e integració es trobaran separades en diferents màquines virtuals ?

La capa de negoci e integració es trobaran a la mateixa màquina virtual i mateix servidor d'aplicacions doncs l'aplicació residirà en el servidor Web del videoclub doncs mitjançant aquesta és gestionarà i no té una naturalesa distribuïda. L'arquitectura permetria utilitzar un contenidor d'*EJBs* a un altre servidor per a la capa de negoci e integració o utilitzar un servidor d'aplicacions que disposi de un contenidor *d'EJBs,* no obstant això no es considera necessari la seva utilització doncs seria molt costos i l'aplicació no requereix de moltes de les funcionalitats que ofereixen els *EJBs* i que més endavant s'expliquen. Concretament farem servir un *Tomcat 6.0* com a contenidor web de *Servlets i JSPs* i classes *POJO* en la capa de negoci.

3. ¿ Quin tipus d'operacions es realitzaran a l'aplicació ?

A la aplicació hi hauran bastantes operacions *CRUD* (alta,lectura,baixa,modificació) per al manteniment de les taules i dades del videoclub per part dels administradors. No obstant això la majoria d'usuaris que utilitzaran l'aplicació seran usuaris que realitzen operacions

de consulta sobre els exemplars disponibles e informació a la web de novetats.. Per tant la majoria de operacions seran de consulta un 60-70% sobre els usuaris de la web.

4. ¿ L'aplicació requereix de una forta transaccionalitat ?

Les operacions de reserva d'exemplars i sol·licitud de préstec requeriran de transaccionalitat en la seva execució no obstant això la resta d' operacions que s'han de realitzar no requereixen de transaccionalitat.

5. ¿ L'aplicació requereix de serveis de seguretat addicionals per accés als components ?

L'aplicació requerirà dels serveis de seguretat i autenticació propis de una aplicació web, i no es requereixen de serveis addicionals per a la seva gestió. En quant a la gestió del saldo dels socis per a fer préstecs aquest es gestionarà físicament al establiment i l'aplicació únicament comprovarà que disposi el saldo necessari per a realitzar el préstec.

6. ¿ L'aplicació requereix de una forta concurrència i escalabilitat?

L'abast de l'aplicació son els visitants del web del videoclub, els socis, i els administradors que gestionen la informació. No es preveu per tant una forta concurrència doncs les operacions a realitzar seran operacions puntuals a realitzar de consulta, préstec i possibles visitants de un videoclub que gestiona la entrega física i retorn dels exemplars al seu establiment.

Analitzant els punts anteriors podem veure que no es tracta de una aplicació pròpiament de naturalesa distribuïda i que si mirem els punts anteriors tot i que a la capa de Negoci e Integració podríem utilitzar *EJB* el seu ús penalitzaria el rendiment doncs es redimensionarien els requeriments que té l'aplicació i es poden resoldre sense el seu ús.

Finalment l'arquitectura que es proposa per l'aplicació:

- Capa de presentació: Utilització de *Jsp i Servlets*, investigació *famework*, patrons de disseny. Accés local a aquesta capa.
- Capa de negoci: Utilització de classes *POJO* e investigació de *frameworks*. Accés local a aquesta capa.
- Capa d'integració: Utilització de classes *POJO* e investigació de *frameworks* per Mapatge O-R, patrons de disseny.

Investigació sobre Patrons

Tot seguit s'ha d'investigar quins Patrons podem aplicar per al desenvolupament de la nostra aplicació. Entenem per patró como una plantilla que es fa servir com a base per donar solució a un determinat problema.

Aquests patrons es poden classificar en:

- **Patrons d'arquitectura:** Aquests patrons s'apliquen en la definició l'arquitectura i afecten al disseny de tot el sistema. (Aquests s'estudiaran a continuació).
- **Patrons d'assignació de responsabilitats:** Aquests s'apliquen per a repartir responsabilitats entre les diferents classes del diagrama de classes.
- **Patrons de disseny:** Aquests patrons que s'apliquen per resoldre problemes de disseny i que no afecten a l'arquitectura.
- **Patrons d'anàlisi:** patrons que s'apliquen durant el anàlisi per a sol·lucionar un determinat problema.

Si seguim el catàleg de Patrons que ens proposa Sun Microsystems en *J2EE* en el catàleg de patrons tenim la llista següent:

Pattern Name	Description
Business Delegate [ACM01]	Reduce coupling between Web and Enterprise JavaBeans ™ tiers
Composite Entity [ACM01]	Model a network of related business entities
Composite View [ACM01]	Separately manage layout and content of multiple composed views
Data Access Object (DAO) [ACM01]	Abstract and encapsulate data access mechanisms
Fast Lane Reader	Improve read performance of tabular data
Front Controller [ACM01]	Centralize application request processing
Intercepting Filter [ACM01]	Pre- and post-process application requests
Model-View-Controller	Decouple data representation, application behavior, and presentation
Service Locator [ACM01]	Simplify client access to enterprise business services
Session Facade [ACM01]	Coordinate operations between multiple business objects in a workflow
Transfer Object [ACM01]	Transfer business data between tiers
Value List Handler [ACM01]	Efficiently iterate a virtual list
View Helper [ACM01]	Simplify access to model state and data access logic

Figura 2.- Patrons de disseny Sun Microsystems

D'aquesta llista de Patrons ens centrarem en alguns dels més importants que es consideren per els factors que s'exposen a contiuació:

- Aplicació web ben estructurada i definida per a la gestió de un portal d'un videoclub *online*.
- Desacoblar al màxim les diferents capes que formen part de l'aplicació.
- Desenvolupar una aplicació multi-idioma.
- Donar una bona gestió de l'accés a dades de l'aplicació.
- Independitzar al màxim l'aplicació del SGBD que emmagatzema les dades.

A continuació presentem cada patró a utilitzar per les diferents capes:

Patrons per a la capa de presentació

Dels patrons que s'han escollit els que a continuació s'esmenten afecten a la capa de presentació:

Patró MVC (Model-Vista-Controlador):

Aquest patró d'arquitectura utilitza les millors practiques de disseny per tal de desacoblar la presentació de l'aplicació del comportament d'aquesta i la seva presentació.
La solució que ens proposa aquest patró separa clarament les responsabilitats dels diferents components del sistema: **Model**, **Vista** i **Controlador.**
Model: encapsula l'estat del sistema.
Vista: presenta les dades al usuari.
Controlador: estableixen les relacions entre les accions del usuari i esdeveniments al sistema.
A continuació tenim una representació d'aquest patró:

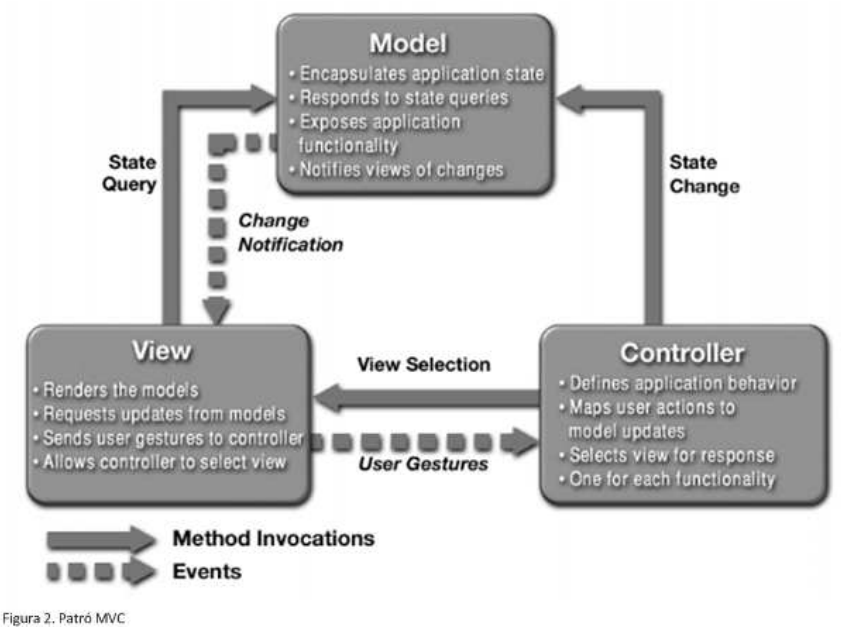


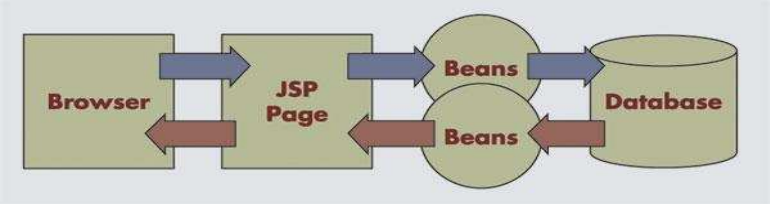
Figura 2. Patró MVC

Figura 3.- Patró MVC

L'aplicació d'aquest patró implica també la utilització del Patró d'assignació de responsabilitats *Front Controller*.

Patró *Front Controller* (Controlador Frontal):

Antigament en el model 1 de les aplicacions web s'aplicava el patró front-page-controller on a les vistes *JSPs* realitzaven tasques tant de presentació com de controlador de l'aplicació.



El problema que tenien aquest sistema era el del manteniment de l'aplicació i la reusabilitat doncs creixia la seva complexitat quan es tractava d'una aplicació més complexa. Amb el Model 2 que proposa l'arquitectura *MVC* el fluxe de l'aplicació i les respostes es centralitzen en un controlador.

La solució proposada per el patró *Front Controller* centralitza totes les peticions enviades per part dels clients, apart es separa les vistes de la lògica. Podem veure un gràfic d'aquesta arquitectura.

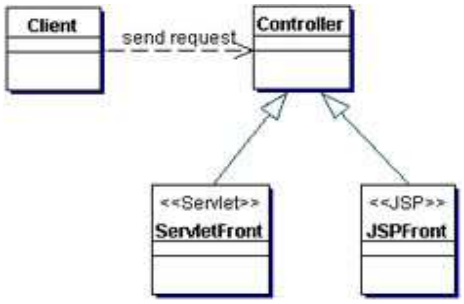


Figura 4.- Disseny Front Controller

Patró *Composite View* (Vista Composta)

En les aplicacions web s'utilitzen vistes que a la vegada estan formades per més subvistes. Aquest patró proposa la utilització de vistes compostes que estiguin formades per vistes atòmiques. En el cas del videoclub sabem que l'aplicació tindrà una estructura determinada amb una capçalera web un peu de pàgina un menú de navegació i el cos central de l'aplicació, per tant seguint aquest patró apliquem aquest esquema a les diferents vistes fent la reutilització del codi i facilitant el manteniment.

Utilitzant aquest patró és mes senzill poder aplicar una determinada plantilla a totes les pàgines.

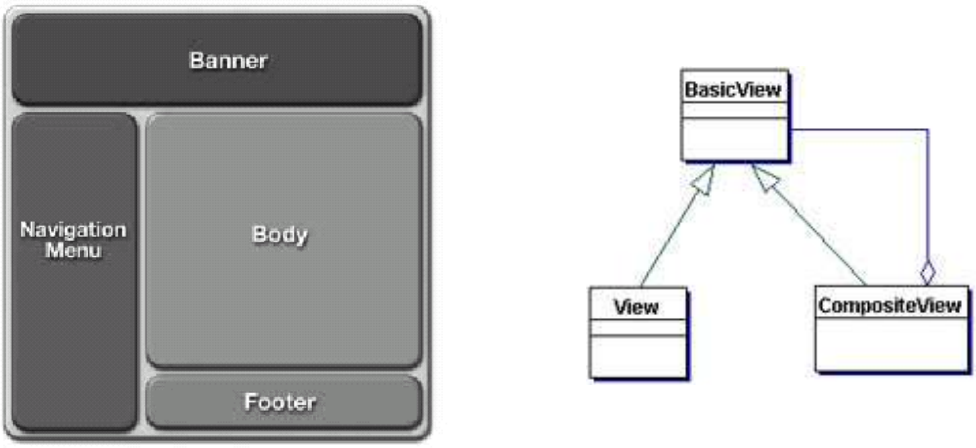


Figura 5.- Patró Composite View

Patrons per a la capa de negoci

A continuació s'exposen els patrons a utilitzar en la capa de negoci.

Patró *Dependency Injection* (Injecció de dependències)

Aquest patró d'arquitectura és molt important i permet separar els serveis que ofereix la aplicació de implementació que es realitza. Es basa en el principi de disseny de inversió de dependències per tal de garantir la reutilització de les diferents classes i limitar el impacte a nivell de domini.

En l'aplicació web del videoclub hem de garantir que la implementació que realitzem pot ser aplicable a diferents SGBD. Pot ser que en un videoclub s'utilitzi un SGBD Postgress

SQL, en un altre *MySQL* i en un altre SQL Server.. Tot això no ha d'influir en la implementació de l'aplicatiu ni les seves funcionalitats.

Podem veure un gràfic exemple d'aplicació d'aquest patró:

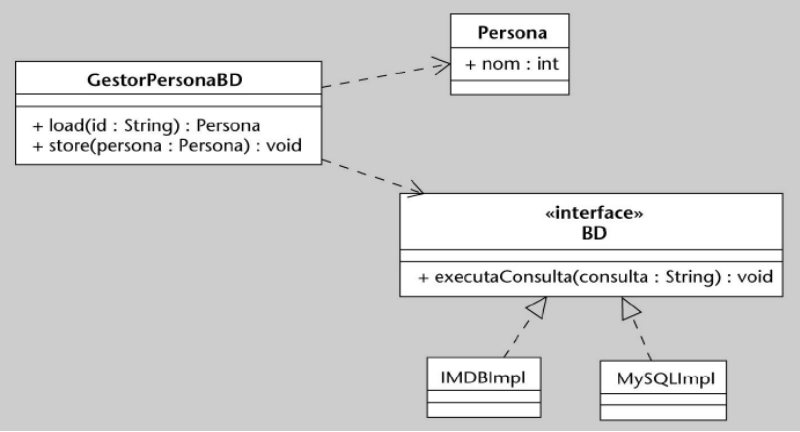


Figura 6.- Patró dependency injection

Patrons per a la capa d'integració

A continuació s'exposen els patrons a utilitzar en la capa d'integració.

Patró *DAO* (Objecte d'accés a dades)

Mitjançant l' utilització d'aquest patró de disseny es pretenen facilitar mètodes d'abstracció i encapsulament d'accés a dades. Aquest patró sorgeix per tal de donar una solució al problema de desenvolupar l'aplicació que sigui el més independent possible al sistema SGBD que hi ha al darrera, de com s'accedeix a les dades o si hi ha o no un SGBD i fer que en el seu cas funcioni amb diferents SGBD.

Aquest patró proposa definir una *<<interface>>* amb la signatura dels mètodes necessaris per a realitzar les diferents operacions i després una classe que implementi aquests mètodes on definirem la implementació específica.

Podem veure un gràfic exemple d'aplicació d'aquest patró:

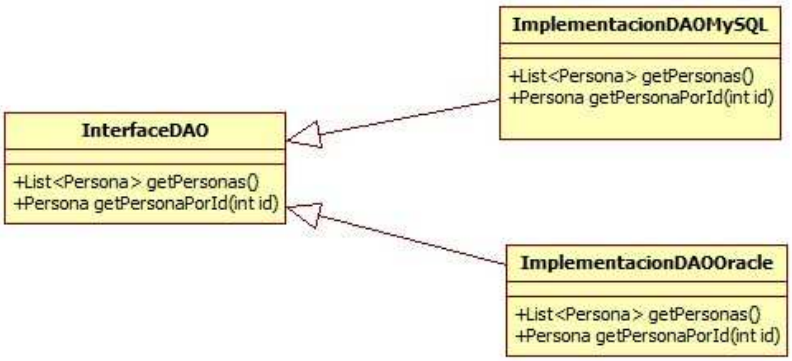


Figura 7.- Patró DAO

Com s'ha comentat anteriorment en el desenvolupament de l'aplicació del videoclub es vol que sigui independent del SGBD i per tant és important utilitzar aquest patró. Apart per tal de abstraure al màxim la seva implementació se sol utilitzar una combinació amb el patró *Factory*.

Patró *Factory* (Factoria)

Aquest patró és considera com una família de patrons i pretén donar una solució a la creació d'objectes de forma genèrica. Tenen la responsabilitat de crear instàncies d'objectes d'altres classes. També la responsabilitat i coneixement necessari per encapsular la forma en la que es creen determinats tipus d'objectes en una aplicació.

Existeixen les següents patrons *Factory*:

- ***Simple Factory* (Factoria simple)**: classe que crea objectes d'altres classes. No delega a subclasses i els mètodes poden ser estàtics.
- ***Factory Method* (Mètode factoria)**: Es defineix una interfície per a crear objectes però es delega a les subclasses la creació en concret.
- **Abstract Factory (Factoria abstracta)**: Ens dona una interfície per a crear objectes de una família sense especificar la implementació en concret.

Podem veure un gràfic exemple d'aplicació de la factoria simple:

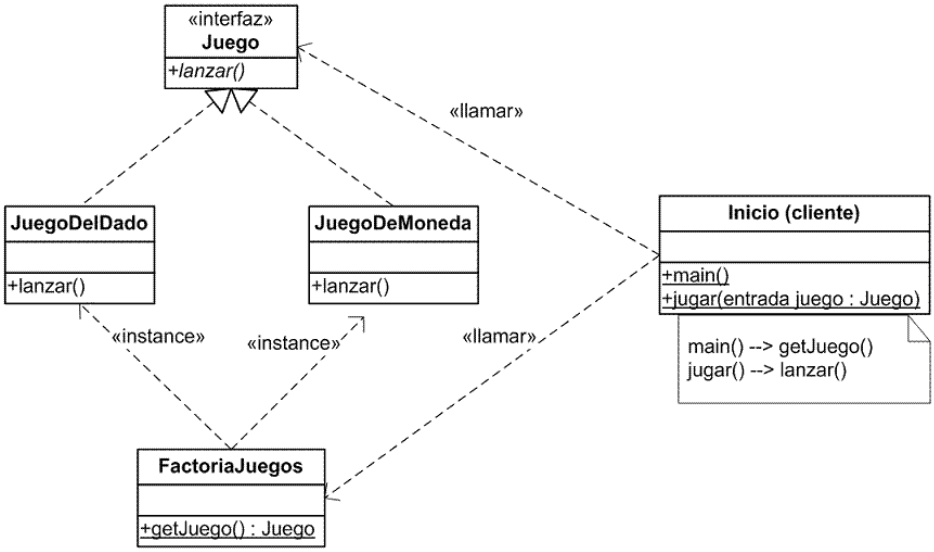


Figura 8.- Patró Factory (Factoria)

Investigació sobre *Frameworks*

Introducció

¿ Que entenem per un *framework*?

Ara que s'han vist diferents patrons desde un punt de vista arquitectònic es pot veure un *framework* com la implementació de diferents patrons de disseny que faciliten la reutilització de disseny i codi així com agilitzar el desenvolupament d'aplicacions.

Per cada una de les subcapes existents a la capa intermèdia de l'arquitectura *J2EE* s'exposaran aquells *frameworks* més rellevants i es justificarà la utilització del *framework* seleccionat.

Frameworks per a la capa de presentació

Existeixen innumerables *frameworks* per a la capa de presentació, no obstant això ens hem centrat en 3 que són aquells que s'han considerat més rellevant desde el punt de vista de la seva utilització i aplicació en el entorn professional i són:

- *Struts.*
- *JSF (Java Server Faces).*
- *Spring MVC.*

Tots ells tenen una sèrie de característiques comunes:

- Donen suport i implementen el patró d'arquitectura MVC donant suport al seu ús.
- Mecanismes per a la validació de les dades d'entrada.
- Control d'errors.
- Suport a la internacionalització de les aplicacions.
- Mecanismes de creació de *Tag Libs* (llibreries d'etiquetes).
- Plantilles per al desenvolupament de l'aplicació.
- Gran reutilització de codi.

A continuació parlem en més detall d'ells.

Struts

Aquest framework realitza la implementació del patró MVC i neix com un projecte de Apache Jakarta de codi obert.

Per entendre l'arquitectura de Struts es presenta aquest gràfic:

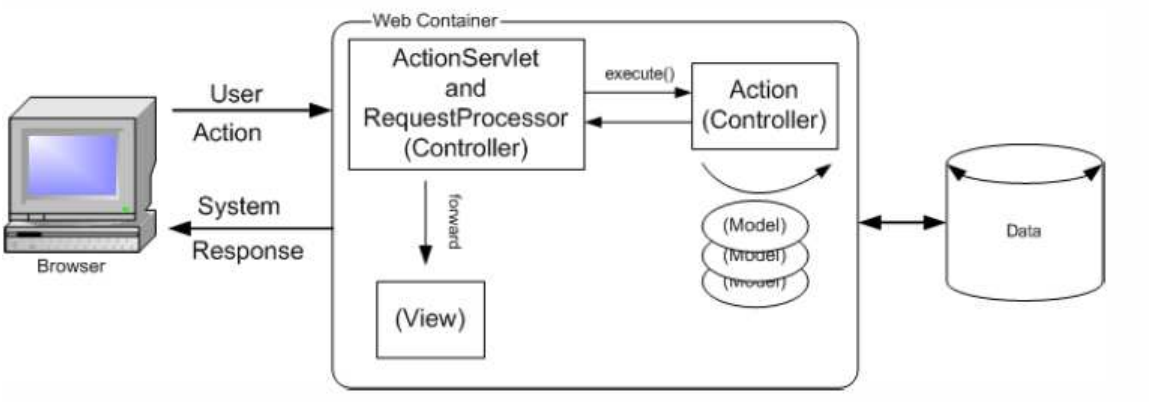


Figura 9.- Arquitectura Struts

Com es pot veure les peticions del usuari es centralitzen cap a un **ActionServlet** *d'Struts* que delega la petició cap a la classe **Action** responsable d'atendre la petició i realitzar les accions necessàries en funció de la configuració feta al descriptor de Struts **struts-config.xml.** Per passar la informació dels formularis d'entrada cap a la action responsable d'atendre la petició Struts defineix els **ActionForm** i per comunicar aquestes dades amb la capa d'integració es fan servir els *Value Objects* (objectes java serialitzables). Per tal de recuperar la informació a les vistes es fan servir els beans definits en aquestes. En el

descriptor d'*Struts* es defineixen també les respostes de les diferents actions i les vistes *jsp* a les que s'ha de redirigir la petició com a resposta.

Punts a destacar del *framework*:

- És un *framework* robust i amb amplia experiència en aplicació de diferents projectes.
- Està orientat al paradigma petició / resposta dins de la capa web.
- La corba d'aprenentatge del *framework* no és excessivament elevada.
- Disposa de una llibreria de *tags* pròpia de suport a la capa de presentació *Struts Tag Libs*.
- És un *framework* ideal per aplicacions web que no tinguin una gran complexitat.
- Ofereix serveis de validació.
- Ofereix serveis d'internacionalització (i18n)
- S'integra perfectament amb el framework *Strut Tiles* que realitza el patró (*Composite View*) i permet el desenvolupament més àgil de la estructura de la web i navegació.
- No és un framework pensat per a extendre'l.
- Hi ha infinitat de documentació a la web i exemples.
- En entorn laboral i empresarial és el *framework* més usat per a la capa de presentació concretament la versió 1 d'aquest.

JSF (Java Server Faces)

Aquest *framework* neix de la ma de Sun Microsystems com una especificació per al desenvolupament de la capa de presentació d'aplicacions empresarials. Aprofita l'experiència que té de *Struts* per tal de resoldre certs apartats no obstant el enfocament d'aquest es completament diferent doncs està orientat a components i manipular els seus esdeveniments.
Es pot veure a continuació un esquema del funcionament de l'arquitectura basada en components:

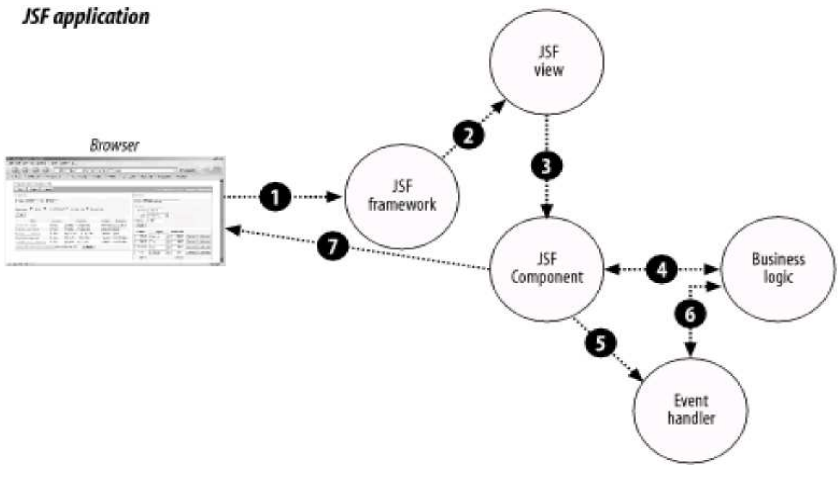


Figura 10.- Arquitectura JSF

A continuació es pot veure el diagrama de classes dels components que formen part de *JSF*:

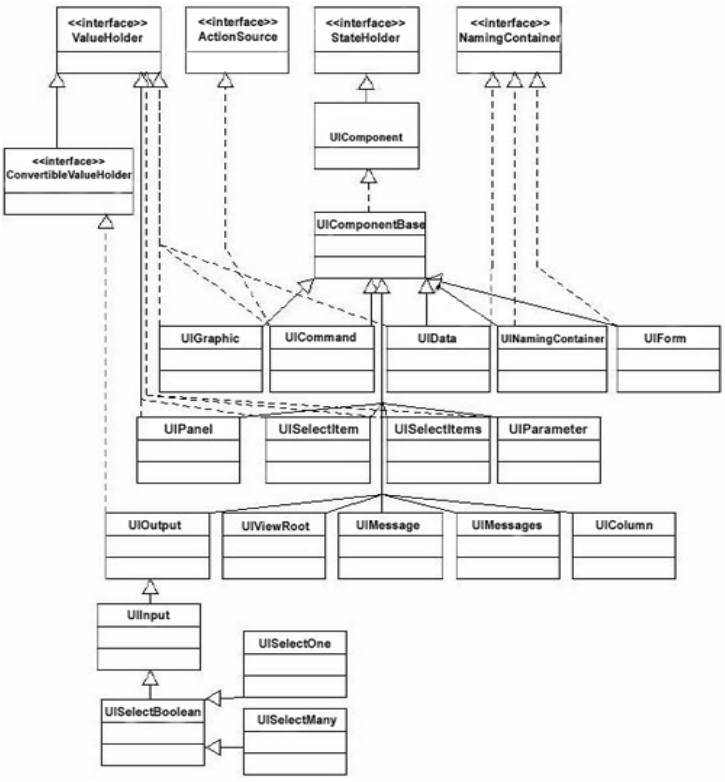


Figura 11.- Diagrama de components JSF

Punts a destacar del framework:

- Pretén ser un estàndard per el desenvolupament d'aplicacions complexes *J2EE*.
- Està orientat al paradigma de gestió de components de la part web i gestió d'events.
- La corba d'aprenentatge del *framework* és una mica més elevada que la d'altres frameworks com per exemple *Struts*.
- És un *framework* ideal per a poder extendre els diferents components i extendre el seu ús.
- No és un *framework* el més idoni per aplicacions que no requereixin de gran complexitat a la capa web.
- Ofereix serveis de validació.
- Ofereix serveis d'internacionalització.

- No té la maduresa que tenen altres *frameworks* com per exemple *Struts*.
- Hi ha bastanta documentació a la web tot i que no tant extensa com altres *frameworks*.

Spring MVC

Aquest *framework* s'engloba dins del *framework* de codi obert *Spring* que pretén donar solucions en el desenvolupament d'aplicacions *J2EE* en tota l'arquitectura. Per tant aporta una visió global per a totes les capes de la aplicació. *Spring MVC* és el encarregat de donar solució a la capa de presentació. Tal i com indica el seu nom compleix el patró d'arquitectura MVC i a continuació s'exposa un esquema de l'arquitectura que proposa:

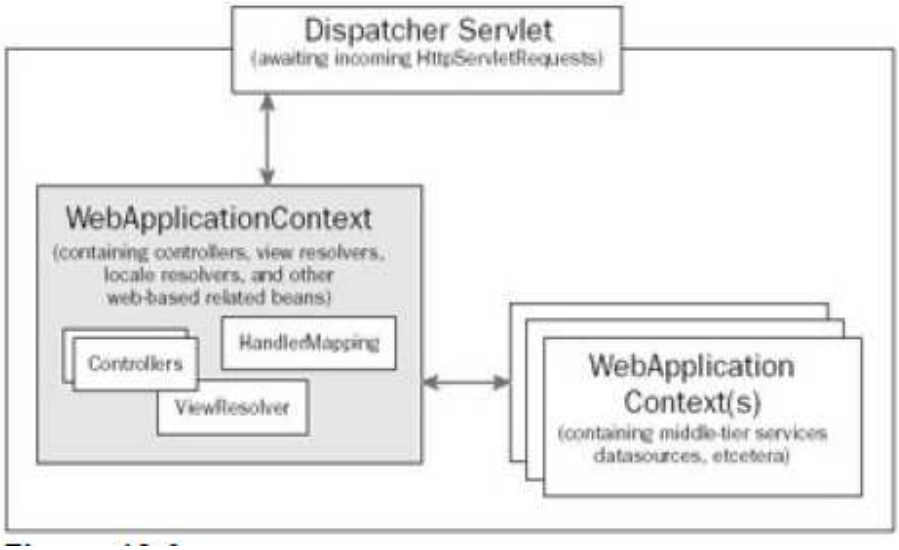


Figura 12.- Arquitectura Spring MVC

L'arquitectura es similar a la de Struts, disposem de un *servlet* **Dispatcher Servlet** encarregat d'atendre les diferents peticions. Aquestes peticions són seleccionades per un **HandlerMapping** i que hi ha de diferent tipus segons el que s'està tractant. Aquesta està associada amb els **Controllers** que la atendran i tractaran i amb els **ViewResolvers** s'indica a quines vistes han de donar resposta.

Punts a destacar del framework:

- Pertany al *framework Spring* i dona solució a la capa web, es pot integrar amb el *framework* en totes les capes o utilitzar de forma independent només en aquesta capa.
- És un *framework* amb un paradigma orientat a petició / resposta similar a *Struts.*

- Aplica el patró *IOC (Dependency Injection)* amb el que es facilita la reusabilitat del codi i integració amb altres components així com el testeig d'aquests.
- Ofereix diferents tipus de Controladors per al desenvolupament i és més parametritzable que altres *frameworks*.
- La corba d'aprenentatge del *framework* no és senzilla doncs incorpora conceptes nous.
- Ofereix serveis d'internacionalització.
- Ofereix serveis de validació.
- No disposa de tanta documentació com altres *frameworks*.
- Tot i que s'està usant en nous desenvolupaments i projectes no te la maduresa d'altres *frameworks.*

Selecció d'arquitectura per a la capa de presentació

Com s'ha vist per a la capa de presentació existeixen diferents *frameworks* i segurament no hi ha una única solució vàlida per a la implementació, també es podria optar per realitzar aquesta fent ús de *JSP* i *Servlets* sense ús de *frameworks*. No obstant analitzant els pros i contres de cada *framework* i diferents possibilitats hem decidit escollir com opció *Struts*.

Motius:

- Es farà servir una aplicació *Thin Client* web orientada a MVC sota petició / resposta.
- Es podria haver optat per *JSF* però el model d'aplicació a desenvolupar no requereix de unes pantalles de gran complexitat ni estem parlant de una web molt complexa per tant es considera que per aquest cas Struts pot donar millor solució.
- Per tal de realitzar certes accions desenvoluparem una llibreria de *tags* pròpia i ens ajudarem de *JSTL* + els *tags* propis de *Struts* i apart integrarem la llibreria *JQuery* per agilitzar certes accions d'scripting al client.
- L'aplicació web a desenvolupar te una estructura clara de les diferents pàgines i funcionament i farem servir el projecte *Struts Tiles* per el seu desenvolupament.
- L'aplicació requerirà dels serveis d'internacionalització doncs serà multi-idioma i utilitzarem els serveis d'*Struts*.
- Utilitzarem els serveis de validació propis de *Struts*.
- Per aquesta capa podríem haver pensat en *Spring MVC* tot i això es considera que *Struts* dona millor resposta per a les necessitats de l'aplicació doncs *Spring* ens

aporta altres conceptes que per aquesta capa no ens són de gran utilitat i no requereix.

- La versió de *Struts* a utilitzar serà la 1, la 2 ha estat un projecte que ha intentat afegir certs aspectes que després ha desbancant el framework *JSF* i que ha esdevingut una especificació per aquesta capa. També hi ha altres projectes que han intentat integrar en Struts els components de *JSF* com *struts faces*, no obstant com ja hem comentat optem per aquesta versió, tot i que antiga però contrastada i amb molts projectes a les esquenes i per les necessitats de l'aplicació es considera que és la opció més senzilla que compleix amb els requeriments.

Frameworks per a la capa de negoci

Com s'ha comentat anteriorment el dilema en aquesta capa era el ús o no d'*EJB* i finalment s'ha explicat que no són necessaris per aquesta. Els motius per els que s'ha descartat el ús d'*EJB* han estat:

- No ens trobem en una aplicació distribuïda la capa de negoci e integració es troben al mateix servidor i mateixa màquina virtual un servidor web on tindrem l'aplicació del videoclub.
- No disposem de diferents tipus de clients que interactuïn amb la capa de negoci seran "*Thin Client*" que accediran via web a l'aplicació.
- El tipus de operacions de l'aplicació seran majoritàriament de consulta. La majoria d'usuaris realitzaran consultes sobre els exemplars disponibles e informació a la web.
- El procés de reserva i sol·licitud d'exemplars requeriran de transaccionalitat, però la resta d'operacions no.
- L'aplicació no requereix una forta concurrència i escalabilitat doncs no es preveu un nombre de peticions concurrents elevades.

En aquesta capa hi ha poques alternatives o no usar *frameworks* o fer ús de *Spring* o algun framework de poc renom com *HiveMind* i *PicoContainer* no obstant s'ha escollit *Spring* per els motius que a continuació s'exposen i s'ha descartat el estudi d'altres alternatives.

Spring

Com ja s'ha comentat ofereix solucions a totes les capes de l'arquitectura *J2EE* i per tal de poder estudiar amb deteniment algunes de les avantatges i característiques que ofereix es mostra a continuació un esquema de l'arquitectura que proposa el *framework*:

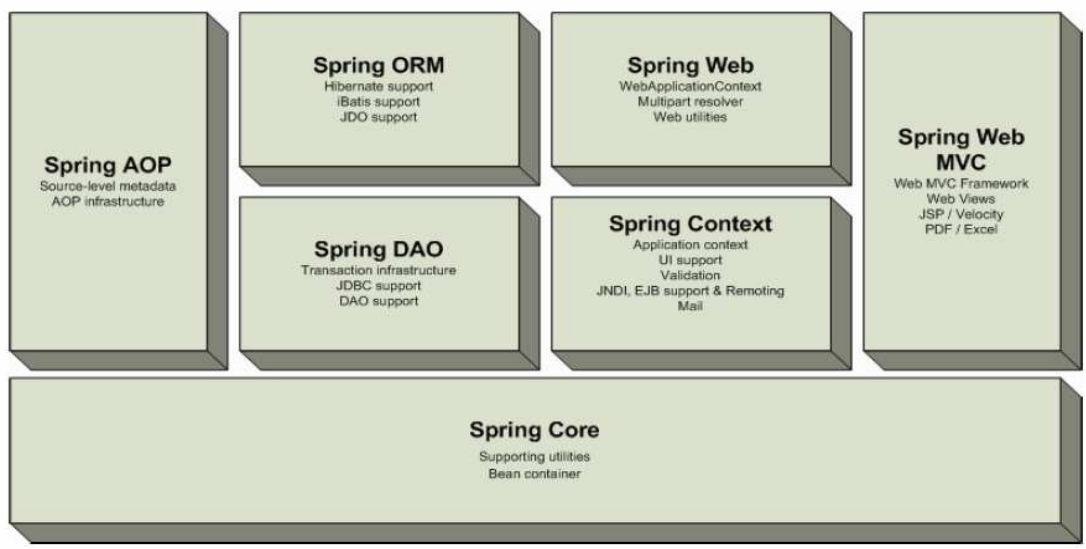


Figura 13.- Arquitectura Framework Spring

Podem veure que es tracta d'una arquitectura multicapa per nivells on cada un dels mòduls implementa diferents serveis o funcionalitats. Aquests alhora son independents i ofereixen una gran flexibilitat, apart s'han desenvolupat seguint les millors pràctiques.

Punts a destacar del framework per la capa de negoci:

- Permet modular la *AOP* (Programació orientada a aspectes). Els aspectes ens permeten modular la gestió de transaccions realitzant una altre manera de desenvolupar el programa en la *OOP* (Programació orientada a objectes).
- Facilita una interfície fent ús del Patró *DAO* per tal de realitzar les transaccions.
- Disposa de un *ORM* (Gestor de relacions d'objectes) propi que facilita la integració amb frameworks de persistència com *Hibernate, Ibatis* o *JDO*.
- Aplica els conceptes de la *IOC (Patró Dependency Injection)* amb el que es facilita la reusabilitat del codi i integració amb altres components així com el testeig d'aquests.

- Diposa de un gestor de *Pool* de Connexions per tal de gestionar la concurrència i les connexions a la aplicació.

Selecció d'arquitectura a la capa de negoci

Analitzant els punts anteriors considerem Spring una bona alternativa i la escollim per:

- És un framework bastant madur i usat com a bona alternativa a la capa de negoci envers al ús d'*EJB*.
- Per a l'aplicació web requerim de certs serveis que ofereix com la gestió de la transaccionalitat.
- Ens facilita una interfície *DAO* per tal de treballar amb les operacions de BD.
- Utilitza aspectes que fan l'aplicació més reusable, que s'integri amb altres components de forma lleugera.
- Ofereix suport per a la integració amb altres *frameworks* de persistència com *Hibernate* o *iBatis*.
- Hi ha bastanta documentació i exemples a la web del seu ús.
- Ofereix serveis per al context de l'aplicació com *JNDI* o la gestió del *Pool* de Connexions.

Frameworks de persistència

Existeixen innumerables *frameworks* per a la capa d'integració, no obstant això ens hem centrat en 3 que són aquells que s'han considerat més rellevant desde el punt de vista de la seva utilització i aplicació en el entorn professional i són:

- *Hibernate.*
- *JPA.*
- *iBatis actualment anomenat MyBatis.*

Tots ells tenen una sèrie de característiques comunes:

- Faciliten el mapatge entre atributs del model de dades i els objectes de l'aplicació.
- Disposen de diferents versions e implementacions.
- Permeten gestionar la seva configuració a través de descriptors *XML*.
- Permeten treballar amb diferents SGBD i de forma àgil gestionar la configuració d'aquests.

A continuació parlem en més detall d'ells.

Hibernate

És el *framework* de persistència més usat i extès actualment. Està basat en software lliure i es distribueix sota una llicència GNU LPGL.

Punts a destacar:

- Permet realitzar el mapatge amb la BD mitjançant fitxers XML o anotacions.
- Es flexible amb el esquema de BD usat i permet adaptar-se a ell o crear la BD a partir de la informació existent.
- Ofereix un llenguatge lleuger de consultes anomenat *HQL* (Hibernate Query Language) i la *API* "criteria" per construir-les programàticament.
- Permet el ús d'anotacions *Hibernate Annotations* que exten de l'estàndard JPA.
- Existeix molta documentació i exemples sobre el seu ús.
- Utilitza el sistema de Cache.
- No ha esdevingut un estàndard.

JPA

És una *API* (conjunt de classes i mètodes) de persistència desenvolupada per a la plataforma *J2EE* i que ha estat adoptada per Sun Microsystems com estàndard per als *EJB* 3.0.

Punts a destacar:

- Existeixen diferents fabricants que la implementen: *Hibernate, Open JPA, TopLink, Eclipselink, OpenJPA, CocoBase, Amber.*
- Unifica els mètodes i utilitats per a realitzar el mapatge objecte relacional.
- Utilitza anotacions.
- Permet el ús de *POJOs* per interactuar amb la Base de Dades.
- No requereix configuració *XML* sols conèixer les anotacions.

iBatis actualment *MyBatis*

El *framework* inicial *iBatis* ha canviat el nom i ara s'anomena *MyBatis*. Aquest *framework* és de codi obert i desenvolupat per Apache Software Foundation.

Punts a destacar:

- Pot ser implementat tant per plataforma Java com .Net
- Associa objectes java amb sentencies SQL i procediments enmagatzemats mitjançant descriptors *XML*.
- Implementa el patró *DAO* per a implementar la capa d'abstracció.
- Implementa un API SQL-Map per a implementar la persistència.
- Defineix una capa *Driver* per a la comunicació amb la Base de dades.
- Permet definir la cache mitjançant *XML* i es suporten diferents implementacions com: EHCache, OSCache i HazelCast.
- Suporta la transacció mitjançant *JDBC* o de forma externa amb *Spring* o *EJB*.

Selecció d'arquitectura per a la capa d'integració

Com s'ha vist per a la capa d'integració existeixen diferents *frameworks* i segurament no hi ha una única solució vàlida per a la implementació, també es podria optar per implementar-la sense el seu ús. No obstant analitzant el pros i contres s'ha decidit escollir *Hibernate* per aquesta capa.

Motius:

- L'aplicació a desenvolupar implementa en el model de dades un sistema multi-idioma que tot i no ser complex requereix de taules relacionades i bastantes taules. *Hibernate* agilitza el mapatge amb un pluguin anomenat *Hibernate Tools* per generar aquesta tasca a partir del model de dades.
- És flexible en el seu ús es pot usar tant fitxers *XML* com anotacions.
- L'aplicació requereix de realitzar bastantes consultes i utilitzar *HQL* per diferents consultes agilitza el seu desenvolupament.
- Permet gestionar el sistema de cache per l'accés a dades i és important per millorar el rendiment de l'aplicació.
- Tot i que altres *frameworks* com *JPA* semblen també interessants, aquest té molta documentació i exemples i és el *framework* més utilitzat actualment i que més flexibilitat dona com a resposta als requeriments de l'aplicació.

Capítol 3: Investigació sobre les *Custom Tag Libs* (llibreries de *tags*) i justificació de tags a implementar

Introducció

¿Que entenem per una *Custom Tag Library* (*Llibreria de tags*) ?

Entenem con una llibreria que conte una sèrie de accions personalitzades que permeten encapsular lògica en les vistes de la capa de presentació. La especificació de les *JSP* ja prové de una llibreria Standard de *tags* anomenada *JSTL* amb funcions comuns necessàries per als desenvolupadors Java en les *JSP*.

A més baix nivell aquestes accions personalitzades són implementades com classes Java Normals. Per tal de utilitzar aquestes accions en les vistes és necessari la utilització de un fitxer *TLD* (Descriptor de la llibreria de *Tags*), que consisteix en un fitxer *XML* que conté el nom de la acció a utilitzar e informació específica de la etiqueta o acció.

Per tant la llibreria de *tags* personalitzada contindrà el conjunt de fitxers *TLD* i classes Java de les accions personalitzades de les etiquetes encapsulades a un fitxer *JAR* per a realitzar una fàcil distribució e instal·lació. Aquesta llibreria la instal·larem com qualsevol altre llibreria al nostre projecte web a la carpeta *WEB-INF/lib.*

Per tal que el contenidor de *JSP* pugui interpretar les etiquetes cal definir una directiva per a la llibreria de *tags*. Aquesta directiva requereix definir un prefix per el tag a utilitzar i com a segon paràmetre hem d' utilitzar un *URI* (Invocació de recurs universal) que serà el nom de la llibreria que definirem al fitxer *TLD* i que associarà amb la classe Java de la acció associada a la etiqueta.

Es pot veure aquesta la relació entre la directiva i el fitxer *TLD* en la imatge següent:

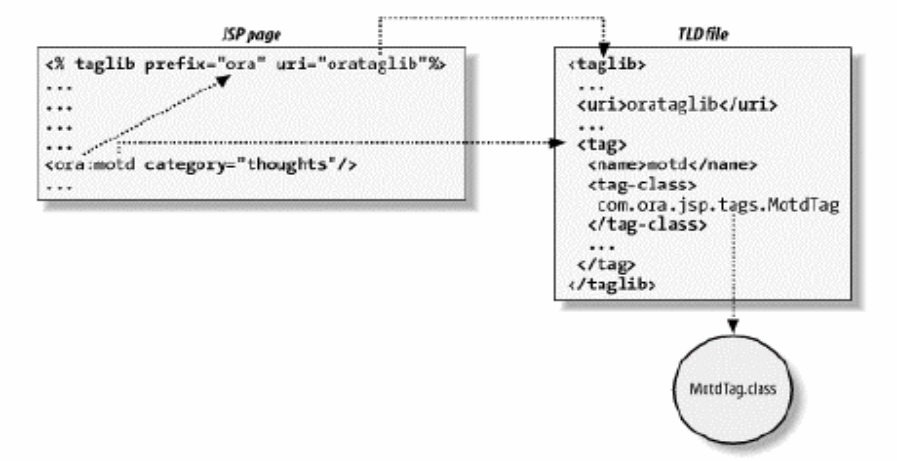


Figura 14.- Fitxers TLD

Per tal de treballar amb les etiquetes s'han de definir les classes capturadores de les etiquetes per a les diferents accions a realitzar. Les tres interfaces primàries per treballar amb aquestes etiquetes són **IterationTag**, **BodyTag** i **Tag**. Per treballar amb aquestes classes s'han desenvolupat noves classes de suport **TagSupportClass** i **BodyTagSupportClass**. Podem veure a continuació un diagrama d'aquestes classes.

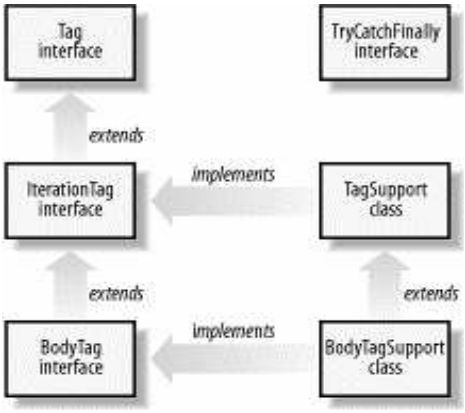


Figura 15.- Diagrama de classes Tag Libs

Aquestes classes defineixen a la seva interface una sèrie de mètodes per a cada una de les etiquetes. Per una part tenim els mètodes accessors 1, 2 del gràfic mostrat a continuació **setNomAtribut()** que s'utilitzen per tal de capturar la informació d'aquells paràmetres que es facin servir amb la etiqueta en cas de que siguin necessaris. El mètode **doStartTag()** s'executa un cop finalitza la etiqueta de inici. Per altre part la etiqueta pot tenir cos amb informació entre la etiqueta d'inici i tancament del tag amb informació a tractar, llavors s'utilitza el mètode **setBodyContent()** per tal de fixar el contingut que hi ha en la etiqueta i posteriorment s'executa el mètode **doInitBody()** un cop s'ha carregat la informació existent al cos de la etiqueta. També existeix el mètode **doAfterBody()** que s'executa després de la inicialització i haver llegit el contingut del cos i finalment tenim la etiqueta de tancament que crida al mètode **doEndTag()**. També existeix la possibilitat de capturar les excepcions produïdes per les accions amb els mètodes **doCatch()** i **doFinally()** de la interface **TryCatchFinally**.

Es pot veure en el següent esquema un diagrama amb aquests mètodes utilitzats:

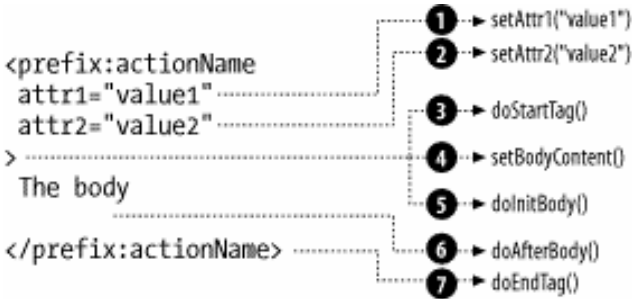


Figura 16.- Funcions dels Tag Libs

A part de les accions comentades anteriorment per les etiquetes es poden crear etiquetes més complexes com etiquetes que contenen etiquetes filles i cooperen amb diferent informació.

Existeixen 2 mètodes **setParent()** i **getParent()** que permeten recuperar els valors de la etiqueta pare i per tant treballar amb aquesta informació. Per altre part existeix el mètode **findAncestorWithClass()** que es recorre la jerarquia d'etiquetes pare fins trobar el que interessa.

Un altre aspecte a tenir en compte quan es desenvolupen accions personalitzades consisteix en no tornar a re inventar la roda. Ens referim a com ja hem dit anteriorment existeix una llibreria Standard de *tags JSTL* per tal d'ajudar als desenvolupadors en el desenvolupament de les vistes i ja hi ha moltes accions desenvolupades, per tant per segons quines accions és preferible adaptar-les a les nostres necessitats. De la mateixa manera utilitzant el *framework Struts* disposem de una llibreria de *tags* Standard. A continuació investigarem més sobre aquestes llibreries i com cooperar amb una llibreria de *tags* pròpia.

Investigació sobre *JSTL* i Standard *tag libs*

Un punt a tenir en compte és la utilització de *JSTL* doncs utilitza accions per les etiquetes que s'han definit per una especificació formal. Això fa que certes accions ja estiguin implementades i definides. Aquesta especificació permet als diferents fabricants desenvolupar les seves implementacions per a les accions de *JSTL*.

La llibreria Standard *JSTL* està formada per 4 diferents llibreries de tags:

Librería	URI	Prefix	Etiquetes
Core (nucli de la llibreria *JSTL*)	http://java.sun.com/jstl/core	c	**<c:catch> ("Captura d'excepcions")** **<c:choose><c:when><c:otherwise>" ("Fluxe de l'aplicació quan compleix condicions")** **<c:forEach> ("Realitza un bucle")** **<c:forTokens> ("Avalua diferents elements d'una cadena de texte")** **<c:if> ("Evaluar una condició")** **<c:import> ("Permet importar recursos externs")** **<c:out> ("Sortida per pantalla")** **<c:param> ("Afegeix paràmetres a la *URL*")**

			<c:redirect> ("Realitza una re direcció *URL* al client" **<c:remove> ("Elimina una variable del àmbit que sigui definida")** **<c:set> ("Assigna el valor de una variable en el àmbit que sigui definida")** **<c:url> ("Aplica per definir *URL* relatives o absolutes")**
Processament d'*XML*	http://java.sun.com/jstl/xml	x	**<x:choose> ("Marca una selecció dins el XML")** **<x:forEach> ("Avalua el cos del *XML* per cada node")** **<x:if> ("Avalua el node si compleix la condició")** **<x:otherwise> ("Avalua si no compleix la condició")** **<x:out> ("Realitza la sortida de un valor")** **<x:param> ("Afegeix un paràmetre al *XSLT*")** **<x:parse> ("Converteix el *XML* a un org.w3c.dom.Document")** **<x:set> ("Afegeix una variable del àmbit escollit una expressió XPath")** **<x:transform> ("Transforma el document utilitzant *XSLT*")** **<x:when> ("Avalua condicions")**
Internacionalització d'aplicacions I18N	http://java.sun.com/jstl/fmt	fmt	**<fmt:bundle> ("Estableix un context per localitzar accions")** **<fmt:formatDate> ("Formateja la data segons el Locale")** **<fmt:formatNumber> ("Formateja un valor numèric segons el Locale")** **<fmt:message> ("Busca el missatge en funció de la clau específica")** **<fmt:parseDate> ("Dona format a la data segons el Locale")** **<fmt:parseNumber> ("Dona format al valor numèric segons el Locale")** **<fmt:requestEncoding> ("Afegeix el character encoding")** **<fmt:setBundle> ("Estableix un context per localitzar accions dins el àmbit que es defineixi")** **<fmt:setLocale> ("Defineix el valor de la variable locale")**

			<fmt:setTimeZone> ("Defineix la configuració de la zona horària") **<fmt:TimeZone> ("Estableix la zona horària")**
Accés a BBDD	http://java.sun.com/jstl/sql	sql	**<sql:dateParam> ("Afegeix una data a una SQL-Query")** **<sql:param> ("Afegeix paràmetres a la SQL-Query")** **<sql:query> ("Defineix una consulta a la BBDD")** **<sql:setDataSource> ("Guarda la informació d'accés a dades en el àmbit que es defineixi")** **<sql:transaction> ("Estableix un context per transaccions")** **<sql:update> ("Estableix una sentencia d'actualització a BBDD")**

Com hem comentat anteriorment també el framework Struts en la versió 1.x incorpora una sèrie de Standard tags a utilitzar a les vistes en 4 llibreries diferents:

Llibreria	URI	Prefix	Etiquetes
Bean (Són útils per treballar amb Java *Beans*)	http://struts.apache.org/tags-bean	bean	**<bean:cookie> ("Defineix una variable en funció d'una cookie")** **<bean:define> ("Defineix una variable en funció d'un bean")** **<bean:header> ("Definex una variable en funció de la capçalera http")** **<bean:include> ("Carrega la resposta de una petició com un *bean*")** **<bean:message> ("Carrega el missatge en funció del idioma")** **<bean:page> ("Mostra un element del contexte de la pagina com un *bean*")** **<bean:parameter> ("Defineix una variable en funció del paràmetre de la petició")** **<bean:resource> ("Carrega un recurs web de l'aplicació i el tracta com un *bean*")** **<bean: size> ("Defineix el nombre d'elements a una Col·lecció o Mapa")** **<bean:struts> ("Mosta la configuració d'*struts* com un bean")** **<bean:write> ("Mostra el valor de una propietat d'un *bean*")**

*Html (*Per generar formularis)	http://struts.apache.org/tags-html	html	**<html:base> (“Genera un element amb href amb una ruta absoluta”)** **<html:button> (“Genera un botó de formulari”)** **<html:cancel> (“Genera un botó de cancel·lació”)** **<html:checkbox> (“Genera un checkbox a un formulari”)** **<html:errors> (“Mostra els missatges d’error”)** **<html:file> (“Genera un camp d’entrada fitxer”)** **<html:form> (“Genera un formulari *HTML*”)** **<html:frame> (“Genera un marc *HTML*”)** **<html:hidden> (“Genera un camp ocult”)** **<html:html> (“Genera la etiqueta <html> amb el llenguatge per defecte”)** **<html:image> (“Genera un camp d’entrada de tipus imatge”)** **<html:img> (“Genera una etiqueta <img> html”)** **<html:javascript> (“Genera una validació de formulari JavaScript”)** **<html:link> (“Genera un enllaç <a>”)** **<html:messages> (“Mostra uns missatges en funció de certes condicions”)** **<html:multibox> (“Genera un checkbox”)** **<html:option> (“Genera una opció de un camp de selecció”)** **<html:options> (“Genera una col·lecció de opcions seleccionades a un camp de selecció”)** **<html:optionsCollection> (“Genera les opcions de una selecció a partir de una col·lecció d’elements”)** **<html:param> (“Permet afegir paràmetres a <html:link><html:rewrite><html:frame>”)** **<html:password> (“Genera un camp de contrasenya”)** **<html:radio> (“Genera un camp radio button”)** **<html:reset> (“Genera un camp per netejar opcions”)** **<html:rewrite> (“Genera un element amb**

			una ruta URL desde el context d'aplicació") **<html:select> ("Genera un camp de selecció HTML")** **<html:submit> ("Genera un camp per enviar les dades del formulari al servidor")** **<html:text> ("Genera un camp *HTML* d'entrada de dades")** **<html:textarea> ("Genera un camp Text Area ")** **<html:xhtml> ("Aquesta etiqueta es molt important per produir planes *XHTML* i al utilitzar tiles que heredin els estils de les pagines que s'inclouen")**
Logic (Per treballar amb la lògica)	http://struts.apache.org/tags-logic	logic	**<logic:empty> ("Avalua la variable si te valor null o es cadena buida">** **<logic:equal> ("Avalua la variable si te el valor igual al indicat")** **<logic:forward> ("Envia a la pàgina amb el ActionForward definit")** **<logic:greaterEqual> ("Avalua si la variable te un valor més gran o igual a un valor")** **<logic:greaterThan> ("Avalua si la variable te un valor més gran a un valor")** **<logic:iterate> ("Recorre una col·lecció d'elements")** **<logic:lessEqual> ("Avalua si la variable te un valor menor o igual a un valor")** **<logic:lessThan> ("Avalua si la variable te un valor menor a un valor")** **<logic:match> ("Avalua si el valor es una subcadena de la variable de la petició")** **<logic:messagesNotPresent> ("Genera el contingut si no hi ha missatge en cap àmbit")** **<logic:messagesPresent> ("Genera el contingut si hi ha missatge en algun àmbit")** **<logic:notEmpty> ("Avalua la variable si no és nul·la o buida per generar el contingut ")** **<logic:notEqual> ("Avalua que la variable no sigui igual al valor indicat")** **<logic:notMatch> ("Avalua que el valor no es una subcadena de la variable de la petició")** **<logic:notPresent> ("Avalua que el valor no**

			sigui present a la petició") **<logic:present> ("Avalua que el valor sigui present a la petició")** **<logic:redirect> ("Realitza una re direcció cap a una resposta")**
Nested (Aquestes són extensions de les llibreries anteriors per generar accions més complexes en etiquetes filles)	http://struts.apache.org/tags-nested	nested	**Tots aquests *tags* son extensions dels anteriorment vistos per utilitzar en situacions on s'hagin d'incloure etiquetes dins d'altres etiquetes per desenvolupar accions més complexes, aquests són:** **<nested:define>** **<nested:empty>** **<nested:equal>** **<nested:errors>** **<nested:file>** **<nested:form>** **<nested:greaterEqual>** **<nested:greaterThan>** **<nested:hidden>** **<nested:image>** **<nested:img>** **<nested:iterate>** **<nested:lessEqual>** **<nested:lessThan>** **<nested:link>** **<nested:match>** **<nested:message>** **<nested:messages>** **<nested:messagesNotPresent>** **<nested:messagesPresent>** **<nested:multibox>** **<nested:nest> ("Permet generar un nou nivell d'anidaments")** **<nested:notEmpty>** **<nested:notEqual>** **<nested:notMatch>** **<nested:notPresent>** **<nested:options>** **<nested:optionsCollection>** **<nested:password>** **<nested:present>** **<nested:radio>** **<nested:root>** **<nested:select>**

			<nested:size> **<nested:submit>** **<nested:text>** **<nested:textarea>** **<nested:write>** **<nested:writeNesting>**

Com hem vist anteriorment existeixen gran quantitats d'etiquetes ja definides tant per utilitzar amb la especificació Standard *JSTL* amb les nostres vistes com fent ús del *framework*. Ara que s'han estudiat es pot tenir en compte alhora de desenvolupar les vistes així com els avantatges que poden aportar.

Justificació utilització Standard *tag libs* i creació de *Custom Tag Libs* (Llibreries d'etiquetes personalitzades)

Arribats a aquest punt cal preguntar-se un punt important. ¿ Es necessària la utilització de Standard *tag libs* ?

Com s'ha vist existeix una especificació estàndard *JSTL* que cada fabricant ha implementat però que defineix un conjunt d'utilitats per al desenvolupament de les vistes en aplicacions *J2EE*. Si fem un resum d'algunes de les avantatges del ús d'aquestes llibreries en les vistes podem resumir:

- Faciliten i agilitzen el desenvolupament de les vistes a la capa de presentació web.
- Aporten un conjunt d'utilitats ja desenvolupades de forma estàndard sense haver de re inventar-se la roda.
- Separen al màxim la lògica de la vista del codi *HTML/XHTML* d'aquesta.
- Permeten generar plantilles específiques a usar a l'aplicació.
- Minimitzen la utilització de codi en vistes complexes.
- Permeten la reutilització de codi en diferents vistes o aplicacions.
- Al haver-les incorporat com estàndard estan lliures d'errors i minimitzen la seva aparició.
- Faciliten la comunicació entre la vista i les diferents capes de l'aplicació

Es pot concloure que la no utilització de llibreries estàndard pot suposar un inconvenient doncs les vistes a desenvolupar poden ser més complexes que fent ús de llibreries *JSTL*. Apart ens podem trobar que cert codi es pugui reutilitzar a diferents vistes, i es necessita

limitar al màxim la seva utilització així com generar nous errors, per tant considerem molt important la seva utilització en les vistes.

Arribats a aquest punt sorgeix una segona pregunta. ¿ Es necessària la creació de *Custom Tag Libs*?

Aquesta és una pregunta complexa i de no fàcil resposta doncs en gran mesura depèn d'un altre factor a tenir en compte, l'aplicació a desenvolupar. És evident si revisem els *tags* estàndard que gran quantitat de funcionalitats i utilitats ja s'han desenvolupat de forma genèrica i aporten una solució generalista al desenvolupament de vistes en aplicacions *J2EE*. No obstant això, aquests no donen resposta a totes les necessitats que ens puguem trobar en el desenvolupament d'una aplicació, i per tant, pot ser necessària la generació de *tags* propis que donin resposta a unes determinades necessitats de desenvolupament. Com ja s'ha comentat en els objectius del projecte un dels punts importants era el desenvolupament d'una llibreria d'etiquetes personalitzades que dones resposta a una sèrie de necessitats funcionals que segueix un model de negoci de un videoclub *online* i per altra part generar una sèrie de *tags* que siguin exportables a altres negocis que segueixin el model de un videoclub. Per tant la resposta es clara, volem agilitzar el desenvolupament de les vistes i per altre part generar etiquetes estàndard per a un videoclub *online*. Ara el que s'haurà de mirar quines son en les vistes d'una aplicació d'aquest tipus de negoci i aquelles necessitats comuns a elles. No obstant ara es considera adient realitzar un resum de l'arquitectura amb la que treballarem per a la nostra aplicació.

Arquitectura resultant per al Videoclub *online*

Ara es important revisar en un esquema per a les diferents capes de l'aplicació *J2EE* els diferents *frameworks* i llibreries de *tags* com utilitats que farem servir per a desenvolupar la nostra solució final:

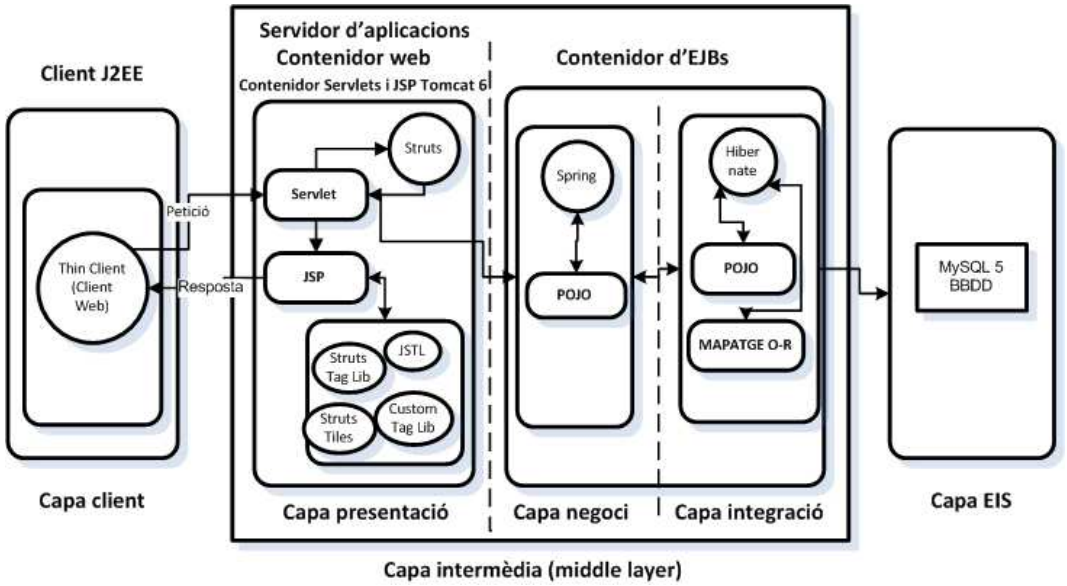


Figura 17.- Arquitectura Resultant Videoclub online

Justificació de *Tag Libs* a implementar

Analitzant el model de Videoclub *online* ens trobem amb un model de web amb una sèrie d'elements comuns a totes les webs que el gestionen. La utilització de tags propis ens pot ajudar en el desenvolupament del model de web i per tal de desenvolupar-los ens hem de centrar en apartats comuns e importants que poden ser útils per a la seva gestió. Per manca de temps no es realitzaran totes les etiquetes que donarien resposta a un model de dades d'aquestes característiques però si aquelles més rellevants.

Alguns dels aspectes comuns en videoclubs *online* desde el punt de vista de presentació de la informació son:

Desde el punt de vista del usuari:

- Mostrar un catàleg de generes dels jocs / pel·lícules existents al videoclub.
- Mostrar un apartat per accés al àrea privada per part dels usuaris del videoclub.
- Generar un catàleg dels jocs / pel·lícules existents i navegar per ells mostrant la informació rellevant i una caràtula.
- Mostrar un apartat amb els tràilers i darreres pel·lícules.
- Un apartat amb RSS de noticies del món del cinema.
- Apartats amb comentaris dels usuaris sobre pel·lícules i estrenes.
- Apartats amb els exemplars disponibles al videoclub per a venta.
- Alguns disposen d'estadístiques com per exemple (les mes visitades, les mes llogades, les més valorades).
- Alguns permeten realitzar comandes *online*.
- Alguns permeten la visualització de pel·lícules *online*.
- Un apartat amb novetats existents al videoclub.

Desde el punt de vista del administrador:

- Llistar la informació de usuaris, pel·lícules, exemplars (llistats amb paginació)..
- Alta, modificació i edició d'aquesta informació mitjançant formularis.
- Formulari d'accés a la zona restringida.
- Apartat de configuració amb paràmetres per personalitzar la web.
- Estadístiques dels usuaris connectats i aquelles pel·lícules, jocs més visitades.
- Menú de navegació per els diferents manteniments i llistats de l'aplicació.

Hi haurien més punts apart d'aquests però s'han esmentat els més rellevants i em considerat escollir 3 d'aquests que es consideren importants i existents a tots ells. Els que s'han escollit són:

1. Generar un catàleg de jocs /pel·lícules existents i mostrar la informació rellevant.
2. Generar un apartat amb les novetats existents al videoclub.
3. Realitzar els llistats de manteniment desde el punt de vista del administrador.

Generar un catàleg de jocs / pel·lícules existents i mostrar la informació rellevant

Un dels punts d'entrada més important a una aplicació d'aquestes característiques es la presentació del llistat principal amb la seva informació. Sovint els usuaris quan es connecten busquen una informació determinada i és rellevant el temps que triguin en accedir a aquesta informació i com es presenti aquesta. Per tant una presentació atractiva i àgil atraurà a més visitants que si no ho és. Es fàcil identificar el catàleg i tots ells tenen una sèrie de característiques comunes com nombre d'elements a mostrar per plana, una imatge amb la caràtula de la pel·lícula o joc, un tamany determinat per cada element e imatge, un estil *CSS* a aplicar amb el que es mostra la informació, una sèrie de camps a mostrar que sol ser el títol de la pel·lícula, el director, actors principals i duració. Alguns inclouen la descripció (normalment s'agafen els 50 primers caràcters i es concatena amb ..) Aquí podem mostrar un exemple, en aquest cas es posa la valoració dels usuaris també amb gràfics d'estrella. En aquest exemple la paginació es configurable per el usuari:

Figura 18.- Exemple presentació catàleg

La realització de un tag configurable que ens realitzi aquests llistats seria de gran ajuda doncs sabem que és un requeriment que tots els videoclubs han de complir i apart tots tenen unes característiques similars en la mostra de la informació per això decidim realitzar-lo.

Generar un apartat amb les novetats existents al videoclub

Un dels aspectes importants per tal de consultar i visitar una web és l'accés a la informació així com la facilitat amb la que s'obté aquesta. En aquest punt i desde l'àmbit del videoclub és obvi que els socis i visitants voldran conèixer abans que ningú quines novetats hi ha disponibles al videoclub per tal d'apropar-se a ell o poder reservar algun exemplar que els interessi. Per tant un apartat o secció amb les novetats és important, apart actualment existeixen llibreries com *JQuery* que ofereixen utilitats per mostrar informació de forma dinàmica sense carregar la pàgina mitjançant *Ajax* i així facilitar aquesta interacció amb el usuari. Per desenvolupar aquest apartat podem desenvolupar un tag que utilitzi aquestes tecnologies i ens generi aquesta llista de novetats dinàmicament i ens mostri cada una un cert temps per pantalla. La informació que mostraríem es la mateixa que al apartat anterior, lo únic que podrem definir un temps amb el que es mostri cada novetat, les opcions de avançar i retrocedir, les mesures d'aquesta secció, el estil *CSS* a aplicar. Podem veure un exemple del que estem comentant:

Figura 19.- Exemple de slider

Aquesta secció va desplaçant el seu contingut mostrant les diferents pel·lícules.
Considerem interessant la realitació d'aquest *tag* per utilitzar diferents tecnologies i presentar informació dinàmicament utilitzant *Ajax*.

Llistats de manteniment per els administradors

Un dels punts més repetitius i on guanyen bastanta força la utilització de tags personalitzats és amb la generació dels manteniments del menú privat d'administrador.
Sabem que en un videoclub tindrem llistats de les pel·lícules existents, els jocs existents, els usuaris, els generes, les classificacions, els exemplars.. Per tant disposar d'aquestes etiquetes alliberarà de feina als desenvolupadors i ajudarà a realitzar aquestes tasques més senzilles. Alguns dels punts comuns que tenen aquests llistats són, un estil *CSS* a aplicar per la taula, el estil per fer la zebra de les files alternes, un llistat amb les capçaleres dels camps, unes *URL* per les accions d'edició del registre i esborrat d'aquest. També poden tenir paginació quan es mostren molts registres.
Aquí podem veure un exemple de un llistat de manteniment:

Código	Título	Género	Calificación	Ver	Borrar	Modificar
A3	007 - EL MAÑANA NUNCA MUERE	ACCION	A.P.M. 13			
10	LA AMANTE DEL TENIENTE FRANCES	ACCION	A.P.M. 16			
12	LA DELGADA LINEA ROJA	DRAMA	A.P.M. 16			
18	EL SEÑOR DE LOS ANILLOS-LAS DOS TORRES	AVENTURA	A.P.M. 16			
19	EL PIANISTA	DRAMA	A.P.M. 16			
A2	ASESINOS 2	ACCION	A.P.M. 16			
5	EL FRANCOTIRADOR	ACCION	A.P.M. 18			
7	LA LLAMADA	TERROR	A.P.M. 18			
8	PANDILLAS DE NEW YORK	ACCION	A.P.M. 18			
9	INFIDELIDAD	THRILLER EROTICO	A.P.M. 18			
A0025333	ASESINOS	ACCION	A.P.M. 18			

Figura 20.- Exemple llistat de manteniment

Considerem important desenvolupar aquesta funcionalitat per alliberar de feina en el desenvolupament de manteniments i guanyar en reutilització de codi en les vistes.

Capítol 4: Anàlisi i disseny parcial del Videoclub *online* i llibreria de *tags*

Anàlisis i casos d'us

Requisits funcionals

La web del videoclub *online* està pensada per complir amb una sèrie de requeriments, no obstant es realitzarà una implementació parcial i per tant alguns requeriments es marcaran com opcionals i altres es descartaran doncs no es pretén obtenir una solució completa ni tampoc es el objectiu del projecte.

Les característiques a complir són:

Ha de ser una aplicació multi-idioma.

Ha de permetre la gestió de diferents tipus d'articles (Pel·lícul·les, Jocs, DVD), no obstant ens centrarem en les pel·lícules doncs després es fàcilment exportable la funcionalitat als diferents tipus d'articles.

Ha de mostrar un llistat dels generes en funció de la classificació. (Ens centrarem en pel·lícules) i poder consultar les pel·lícules en funció d'aquest.

Ha de permetre l'autenticació per part dels administradors i accés al menú privat d'administrador. En quant a manteniments ens centrarem en el (alta, modificació, baixa) de pel·lícules i exemplars. La resta de taules de configuració amb informació com (socis, idiomes aplicació, configuració, director, format, intèrprets, classificació, categoria...) Per normalitzar les dades d'alta d'una pel·lícula no es realitzaran els manteniments.

Ha de permetre la consulta de les darreres novetats existents al videoclub.

Ha de permetre el accés a l'àrea privada per part dels socis registrats.

Ha de permetre la cerca per títol de pel·lícules la resta de cerques no es desenvoluparan.

Ha de permetre afegir comentaris a les pel·lícules per part dels socis opcional.

Ha de permetre el alta de socis al videoclub per consultar els exemplars disponibles i realitzar gestions no obstant per actualitzar el saldo s'haurà de realitzar físicament al videoclub.

La part de reserva d'exemplars per part dels socis es posa com un opcional si es disposa de temps. La part de realització de préstecs d'exemplars no es realitzarà tot i que s'hauria de tenir en compte en una fase posterior.

Hauria de disposar de un apartat de notícies per tal de informar als visitants. Aquest requeriment no es tindrà en compte per el moment.

Ha de permetre consultar estadístiques de reserves i visitares. Aquest requeriment no es tindrà en compte per el moment.

Actors

A la web del videoclub *online* ens trobem amb 3 perfils diferenciats d'usuaris. Aquests són:

Usuari: aquest actor representa el usuari visitant del web que vol informar-se del videoclub, consultar les novetats, i veure els exemplars dels que disposa el videoclub i registrar-se com a soci.

Soci: es aquell que podrà realitzar les mateixes tasques però apart com a client del videoclub podrà consultar la disponibilitat d'exemplars físicament al videoclub i fer-ne la seva reserva tranquil·lament desde casa.

Administrador: aquest actor pot realitzar les mateixes tasques que un usuari i apart com administrador disposa accés total a la gestió de tots els continguts existents a la web.

El diagrama resultant el podem veure a continuació.

Diagrama de casos d'ús

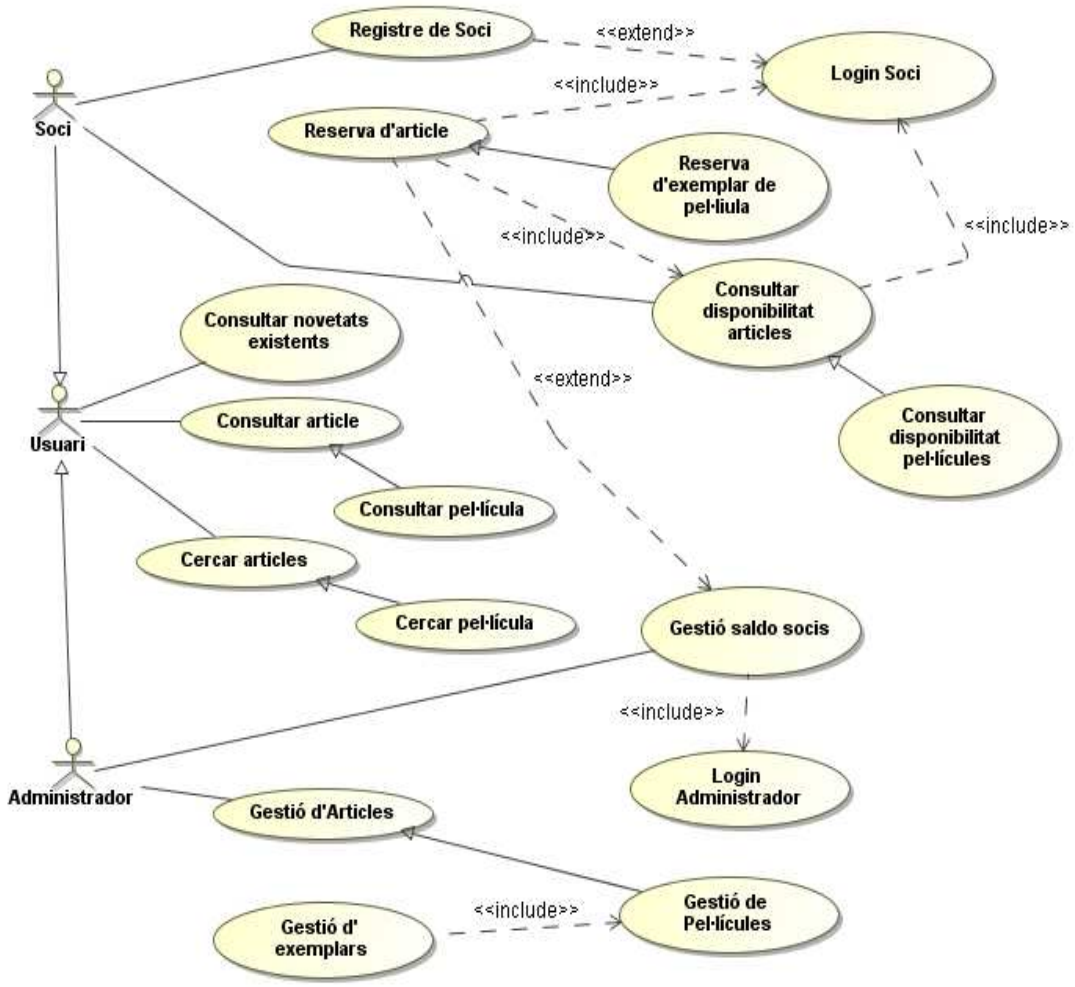


Figura 21.- Diagrama de casos d'ús

Nota: com s'ha esmentat anteriorment alguns dels casos son opcionals i no es realitzaran

Cas d'ús Login de Soci	
Resum de la funcionalitat:	Permet validar que un soci està registrat al videoclub
Flux d' esdeveniments principal:	El usuari introdueix un nom d'usuari i contrasenya i es valida al sistema. Un cop comprovada la validació s'accedeix al menú privat de soci.
Flux d' esdeveniments alternatius:	Si l'usuari no es validat correctament es mostra un

	missatge d'error en la pantalla de Login per tornar a introduir les dades.
Pantalles:	Login de Soci, Menú Privat Soci

Cas d'ús Registre de Soci	
Resum de la funcionalitat:	Permet Registrar un soci al videoclub
Flux d' esdeveniments principal:	El usuari accedeix a un formulari per tal de donar-se d'alta com a soci desde un enllaç a la Pantalla Login Soci. Un cop introduïdes les dades mitjançant el botó de guardar es comprovaran les dades i si són correctes es mostrarà la informació del registre i rebrà un missatge.
Flux d' esdeveniments alternatius:	Si l'usuari introdueix alguna dada no vàlida es mostrarà al costat aquesta informació. Si l'usuari decideix fer ús del botó cancel·lar es tornarà al menú principal.
Pantalles:	Registre de Soci, Login Soci

Consultar disponibilitat d'articles (Implementem disponibilitat pel·lícules)	
Resum de la funcionalitat:	Permet consultar la disponibilitat d'exemplars d'un article determinat al videoclub disponibles per a reserva
Flux d' esdeveniments principal:	Un cop s'accedeix al menú d'usuari per defecte es mostra la disponibilitat de totes les pel·lícules. El usuari pot seleccionar la classificació i el tipus de pel·lícula i un cop es fa ús del boto de cerca s'executa el procés de consulta. Aquest procés busca aquells articles que compleixen amb els criteris de cerca i tenen articles associats disponibles.
Flux d' esdeveniments alternatius:	Si no es troben exemplars disponibles o es produeix algun error al procés es mostrarà un missatge
Pantalles:	Consulta Disponibilitat, Menú privat soci

Reserva d' Article (Opcional per a pel·lícules)	
Resum de la funcionalitat:	Permet fer la reserva d'un exemplar d'una determinada pel·lícula
Flux d' esdeveniments principal:	Un cop s'han consultat els articles sobre aquell que es desitgi es podrà realitzar la reserva d'aquest mitjançant

	l'ús del botó o enllaç de reserva. Primerament es comprova si el soci disposa de saldo a la seva compta per realitzar la reserva. Si es així, apareixerà una finestra emergent amb un llistat dels exemplars disponibles per a reserva d'aquell article i es podrà seleccionar el que es desitgi, seleccionant la opció i acceptant la reserva, a continuació apareixerà un missatge amb la data de reserva i la seva validesa i codi d'exemplar i codi de soci i es rebrà un mail fent el canvi d'estat del exemplar a reservat
Flux d' esdeveniments alternatius:	Si no disposa de saldo es mostra un missatge conforme no pot realitzar-la en una finestra emergent. Si hi ha dos usuaris realitzant al mateix temps la reserva d'un mateix al darrer del error i es refresca la pantalla amb el llistat d'exemplars.
Pantalles:	Reserva Article

Consultar Novetats Existents (Pel·lícules)	
Resum de la funcionalitat:	Permet per als usuaris i visitants al web consultar les novetats de pel·lícules al videoclub en un apartat de la web
Flux d' esdeveniments principal:	Aquest espai web disposarà de unes dimensions determinades parametritzables. El usuari anirà veient cada cert temps les pel·lícules una darrera l'altre en intervals de 7 segons
Flux d' esdeveniments alternatius:	El usuari podrà clickar a veure el tràiler i llavors s'obrirà una finestra per veure el tràiler amb un objecte de control per visualitzar-la. El usuari podrà donar a veure el detall de la pel·lícula també en una nova finestra amb la informació detallada d'aquesta
Pantalles:	Detall Pel·lícula

Consultar d'articles (Cerca de Pel·lícula per títol)	
Resum de la funcionalitat:	Permet realitzar la cerca de una pel·lícula per el títol
Flux d' esdeveniments principal:	El usuari introduirà la paraula o paraules a cercar en el camp de texte de cerca i utilitzarà el botó de cercar. Un cop usat es mostrarà un llistat d'aquelles pel·lícules que

	tinguin al títol les paraules introduïdes. El format a presentar serà el mateix que amb la pantalla principal
Flux d' esdeveniments alternatius:	Si el procés de cerca no troba cap resultat es mostrarà un missatge informant
Pantalles:	Llistat Pel·lícules

Cas d'ús Login d'Administrador	
Resum de la funcionalitat:	Permet validar que el accés es restringeix a un administrador
Flux d' esdeveniments principal:	El administrador introdueix un nom d'usuari i contrasenya i es valida al sistema. Un cop comprovada la validació s'accedeix al menú privat d'administració.
Flux d' esdeveniments alternatius:	Si l'usuari no pot validar l'accés es mostra un missatge d'error per tornar a validar
Pantalles:	Login d'Administrador, Menú Privat Administrador

Gestió d'Articles (Gestió de Pel·lícules, la Gestió d'Exemplars Opcional)	
Resum de la funcionalitat:	Menú d'administració que permet fer l'alta baixa, modificació i consulta de pel·lícules.
Flux d' esdeveniments principal:	El administrador després d'haver realitzat el login consulta el llistat de pel·lícules i selecciona aquella que vol esborrar, o modificar amb la icona d' accions associada. Per a donar d'alta una pel·lícula es mostra un nou formulari amb tots els camps de la pel·lícula associats per donar d'alta la informació. A continuació es guarda i si tota la validació es correcta es torna al menú privat visualitzant la llista de pel·lícules actualitzada.
Flux d' esdeveniments alternatius:	Si es desitja modificar una pel·lícula s'obre un formulari amb les dades carregades per poder modificar-les. Si es desitja esborrar una pel·lícula prèviament es confirma la eliminació i a continuació es mostra el llistat de pel·lícules un cop esborrada.
Pantalles:	Login d'Administrador, Menú Privat Administrador

Disseny de l'aplicació: Videoclub *online*

Diagrama de classes

El diagrama de classes resultant el podem veure a continuació:

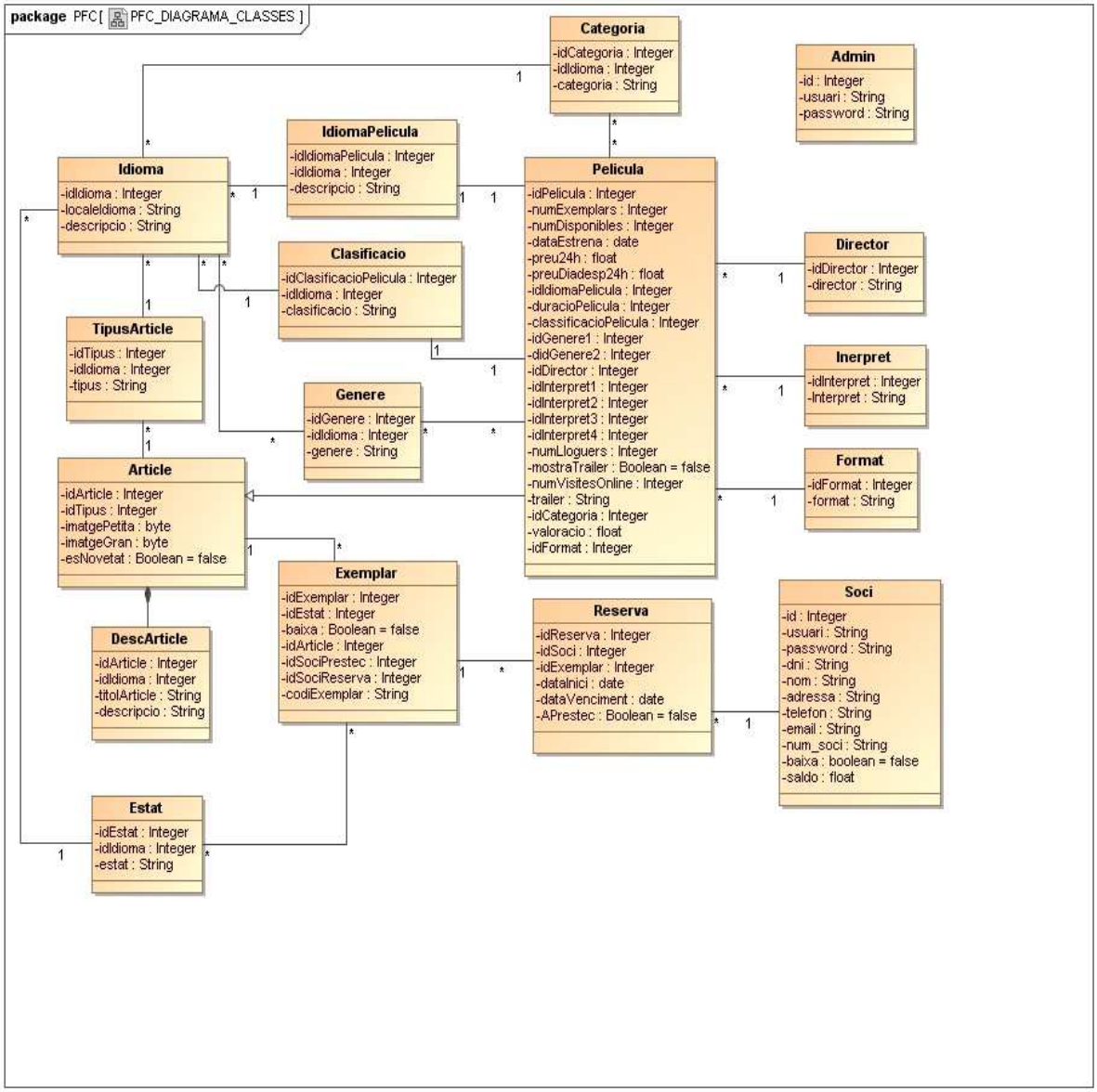


Figura 22.- Diagrama de classes

Disseny basat en components

Si realitzem el disseny de l'aplicació per cada una de les capes de l'arquitectura que tenim en components en una primera fase obtenim:

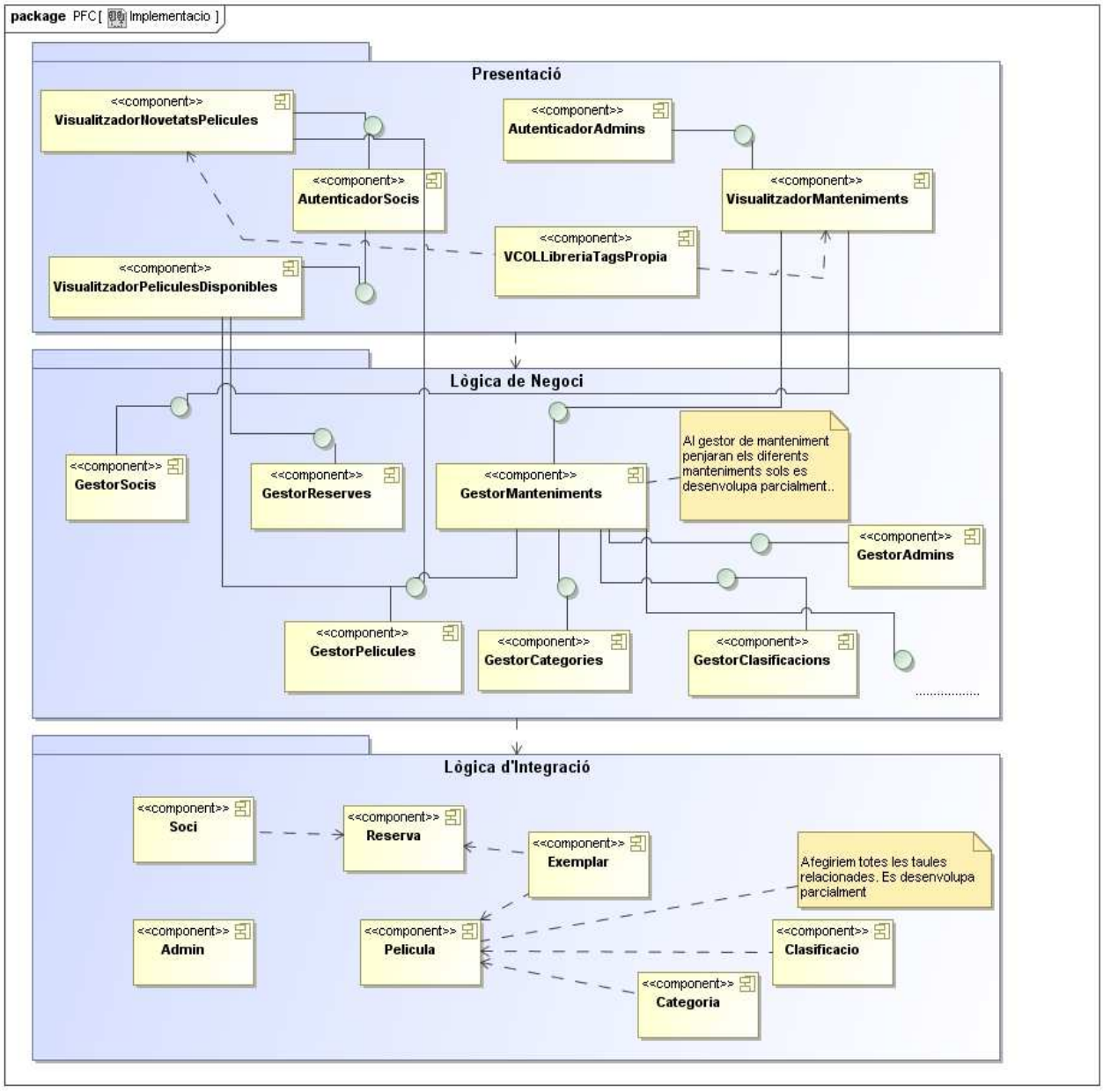


Figura 23.- Diagrama components 3 capes

Posteriorment al aplicar una primera fase de refiinament en cada una de les capes obtenim:

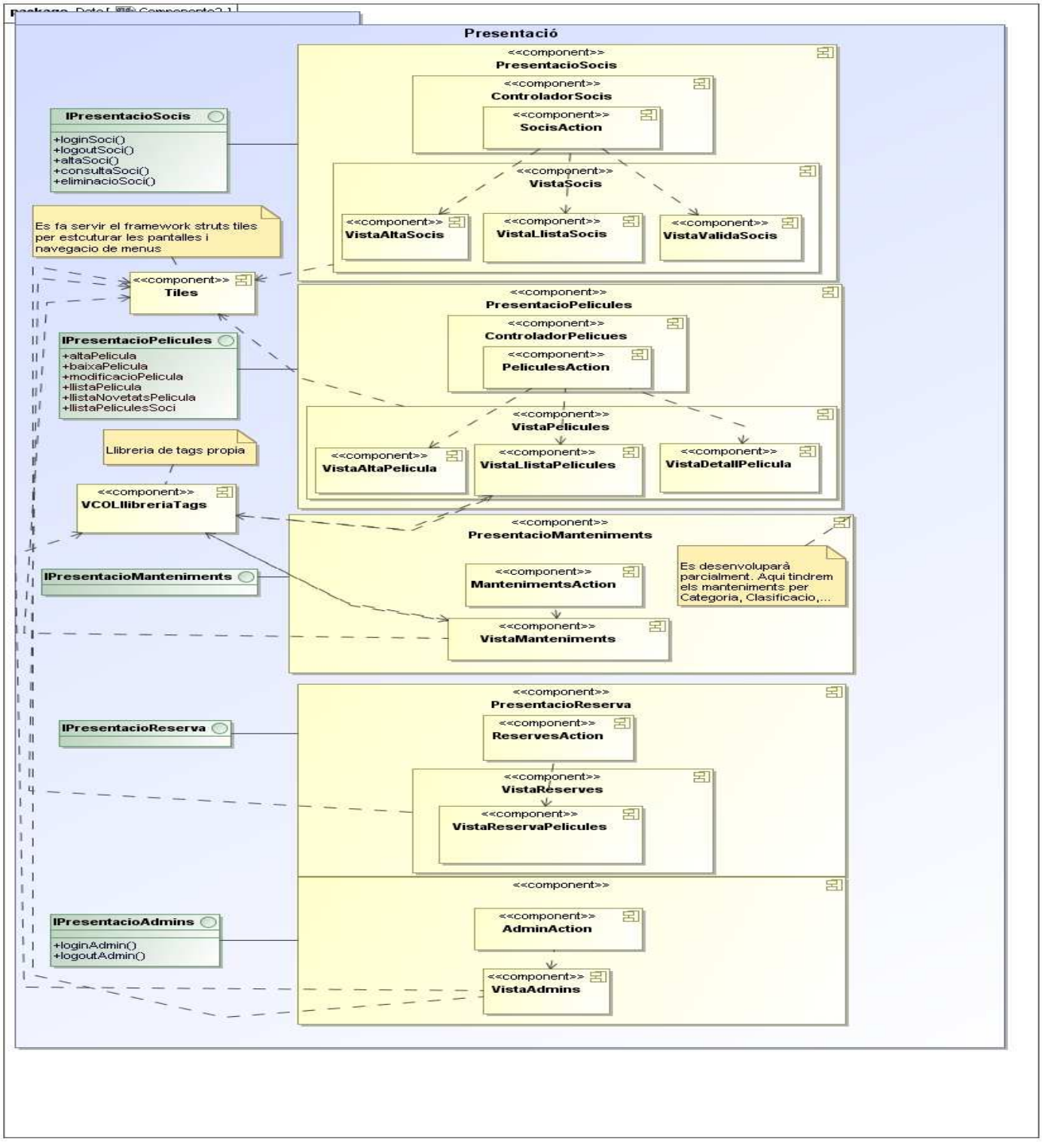


Figura 24.- Diagrama capa presentació

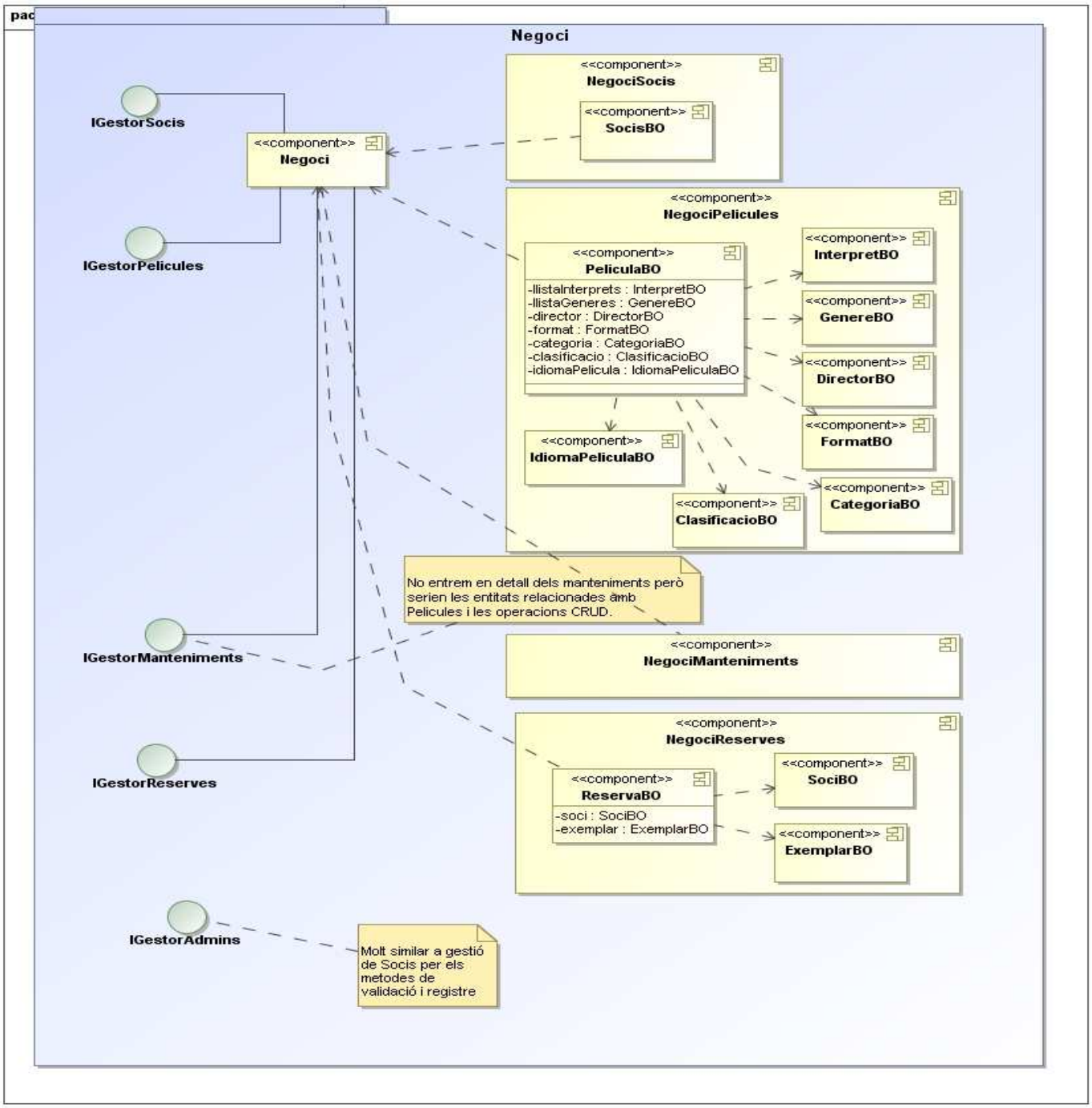


Figura 25.- Diagrama capa de negoci

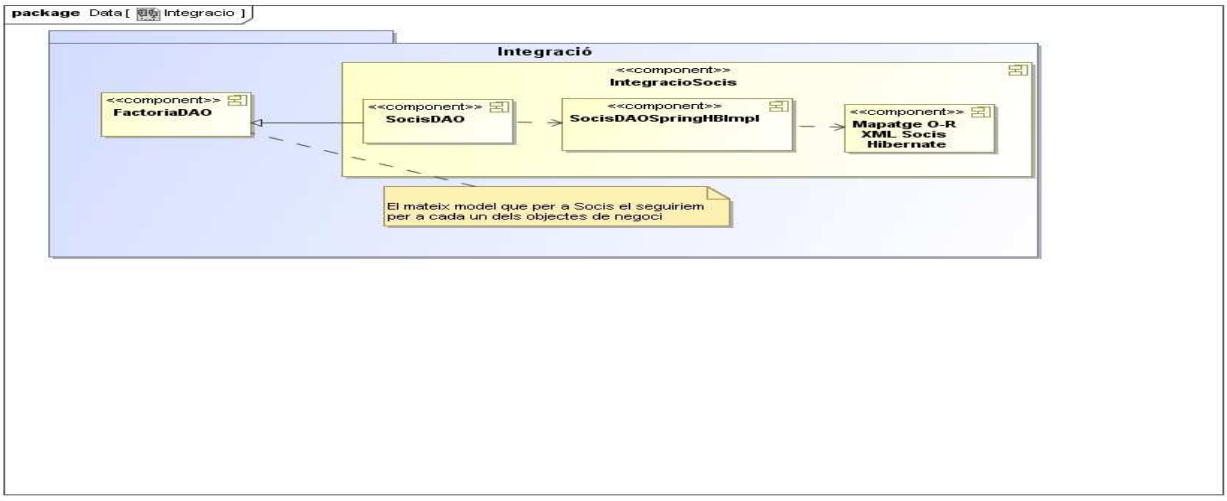


Figura 26.- Diagrama capa d'integració

Si realitzem el disseny de la llibreria de tags de forma més detallada obtenim:

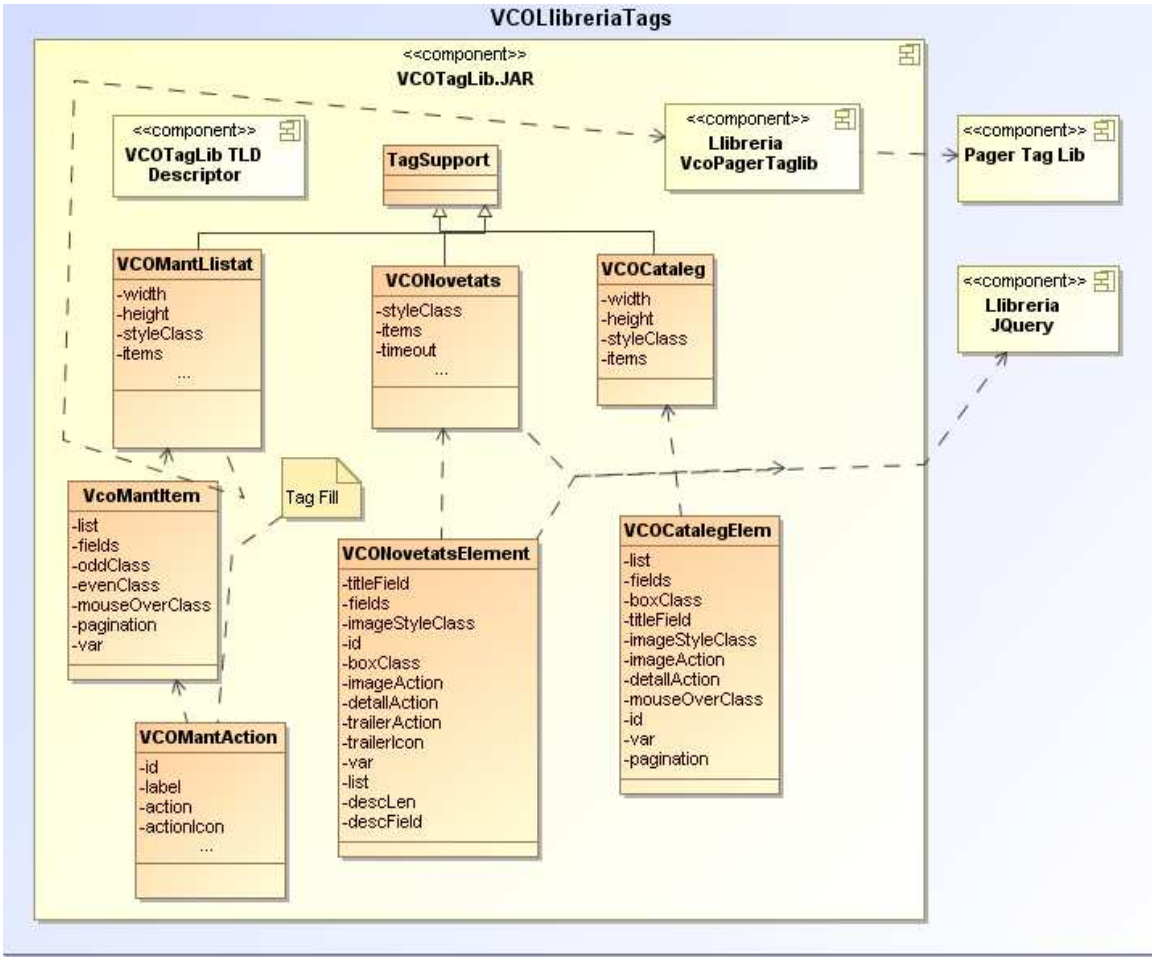


Figura 27.- Diagrama llibreria de Tags

Diagrames de col·laboració i seqüència

Hem optat per documentar el comportament i col·laboració en cada un dels casos d'ús i fer el diagrama de seqüència del cas d'ús més important.

Figura 28.- Diagrama de col·laboració Login de Soci

Figura 29.- Diagrama de col·laboració Registre de Soci

Figura 30.- Diagrama de col·laboració consulta disp. pel·lícula

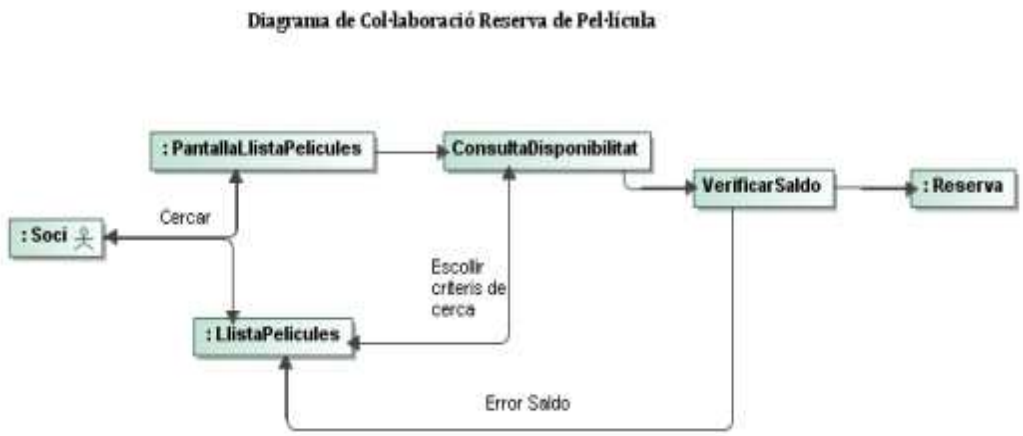


Figura 31.- Diagrama de col·laboració reserva pel·lícula

Figura 32.- Diagrama col·laboració consultar novetats existents

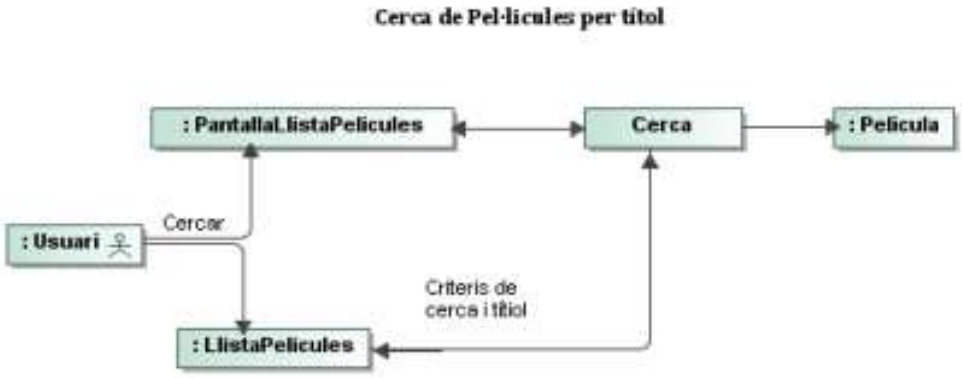


Figura 33.- Diagrama de col·laboració cerca de pel·lícules per títol

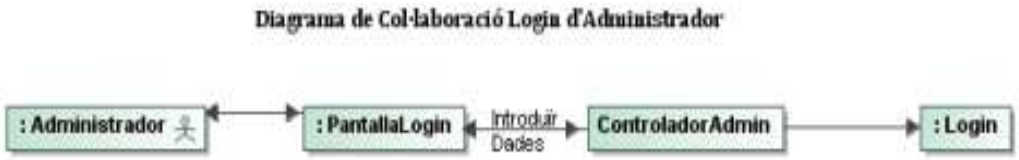


Figura 34.- Diagrama de col·laboració Login d'Administrador

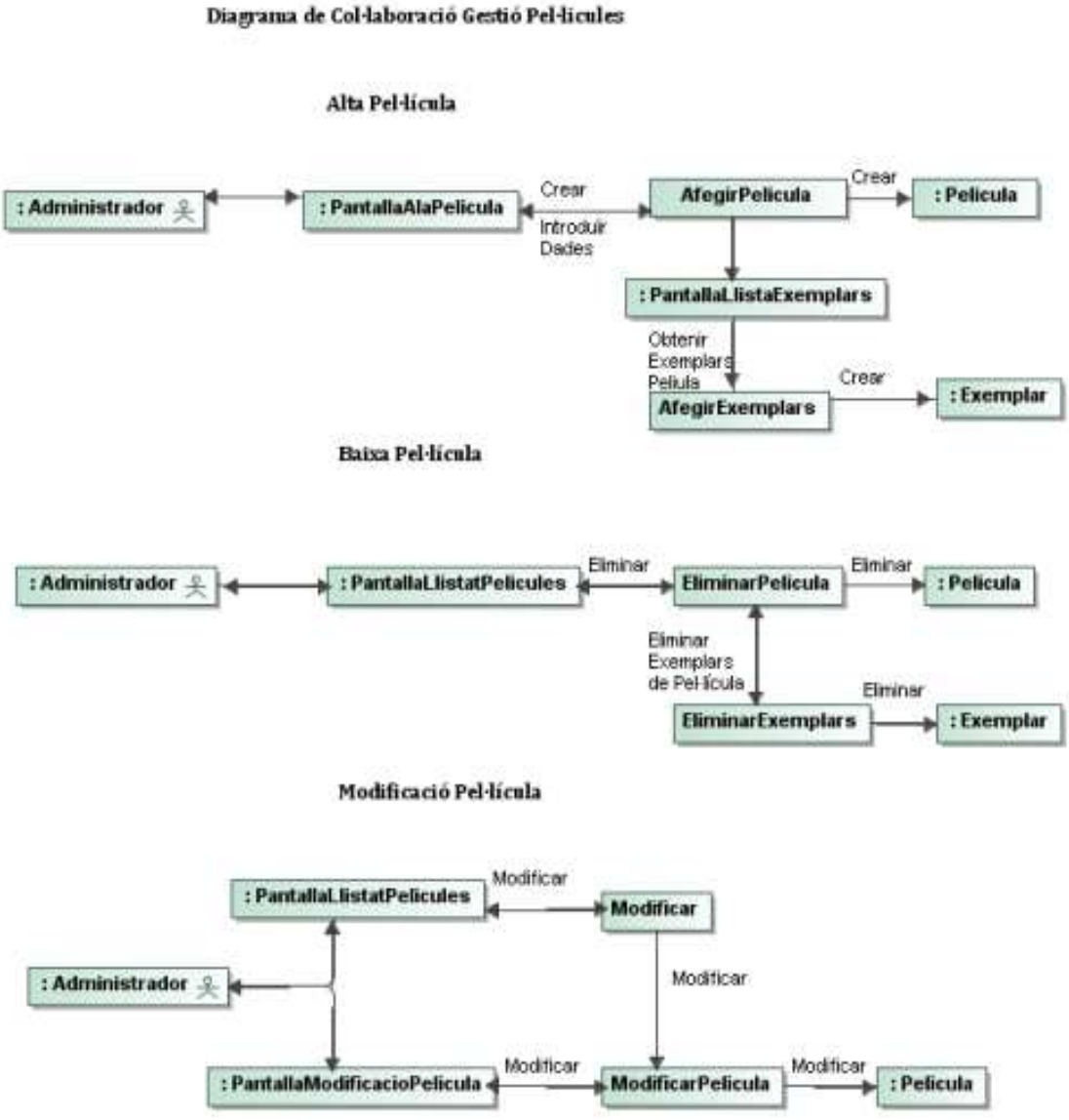


Figura 35.- Diagrames col·laboració gestió pel·lícules

Model de dades resultant

El model de dades resultant per tal de donar resposta a les necessitats definides és el següent:

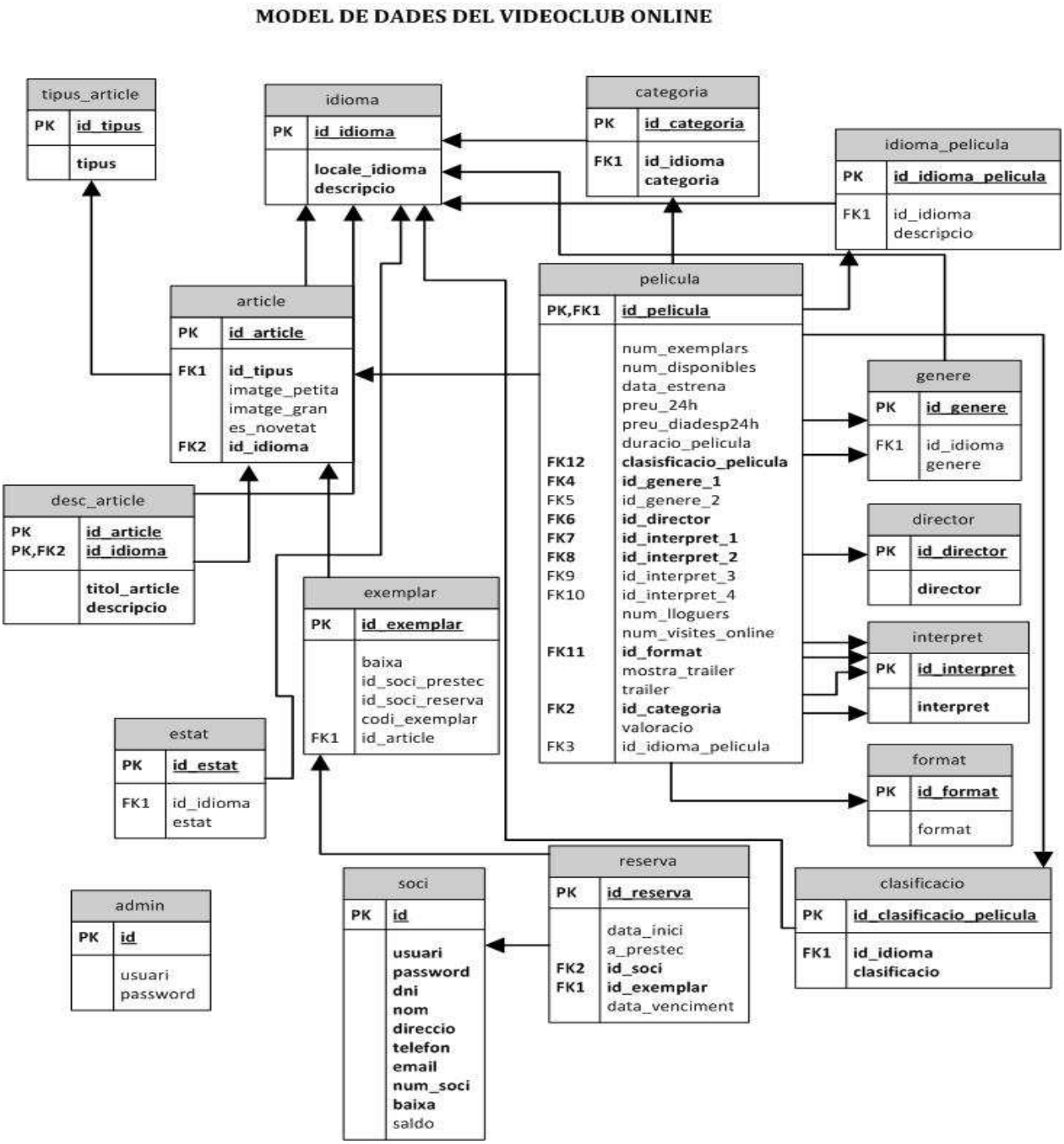


Figura 36.- Model de dades

Capítol 5: Implementació del Videoclub online i llibreria de tags

Implementació de la capa de presentació del Videoclub *online*

La realització de la implementació de la capa de presentació fent ús del *Framework Struts, Struts Tiles* i els serveis i18n de Struts implica:

1.- Creació de les plantilles de *Struts Tiles* a fer servir per a la implementació de les *JSP* del Videoclub *online.*

2.- Definir la navegació de *Struts* en el fitxer struts-config.xml.

3.- Definir les accions de *Struts,* Struts Action*.*

4.- Crear els *beans* de *Stuts* ActionForm*.*

5.- Crear les validacions dels formularis.

6.- Fer ús dels serveis i18n de internacionalització per al desenvolupament.

7.- Crear els objectes seleccionables per els combos de l'aplicació.

1.- Com ja s'ha justificat anteriorment s'ha escollit *Struts Tiles* per a definir les diferents plantilles i homogeneïtzar el desenvolupament de les pantalles de l'aplicació. En el nostre cas hem definit 3 *layouts* o regions:

- **tiles/videoclub-layout.jsp:** Representa el menú principal de l'aplicació i està formada per les regions:
  header: capçalera de la web on situarem el logo de la empresa.
  header_registro: regió on els socis podran accedir al seu menú privat.
  header_busqueda: regió per tal de realitzar cerques sobre el catàleg.
  menu_header: menú de l'aplicació amb els diferents apartats del lloc web.
  menu_left: menú esquerra per tal de fer consultes o mostrar informació per diferents criteris.
  body: contingut principal de la plana on es mostrarà la informació.
  footer: peu de pàgina amb les dades legals i de contacte de la empresa.

El esquema resultant del *layout* és el que mostrem a continuació:

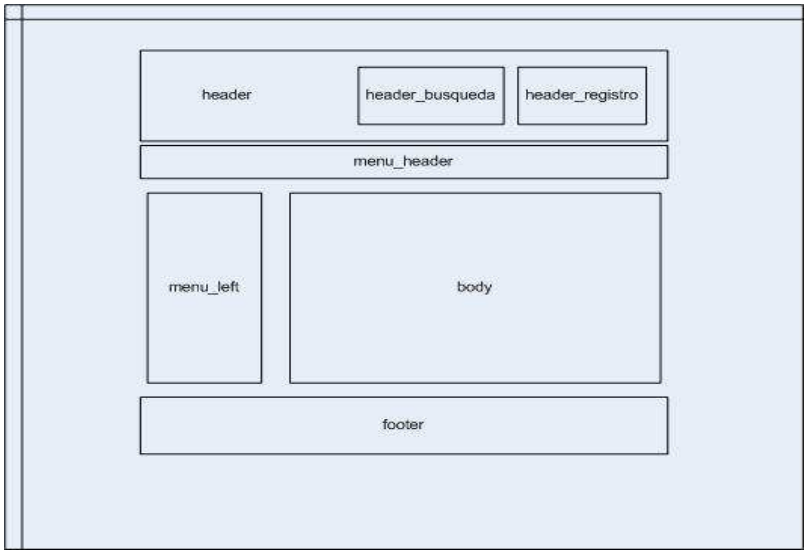


Figura 37.- videoclub-layout.jsp (Portada)

- **tiles/videoclub-layout-admin.jsp:** Representa el menú del administrador amb les regions que mostrem a continuació:

  header: capçelera de la web on situarem el logo de la empresa.

  menu_header: menú de l'aplicació amb els diferents apartats del lloc web. En aquests cas seran els menús de manteniment de administrador.

  menu_left: menú esquerra per tal de fer consultes o mostrar informació per diferents criteris.

  body: contingut principal de la plana on es mostra la informació.

  footer: peu de pàgina amb les dades legals i de contacte de la empresa.

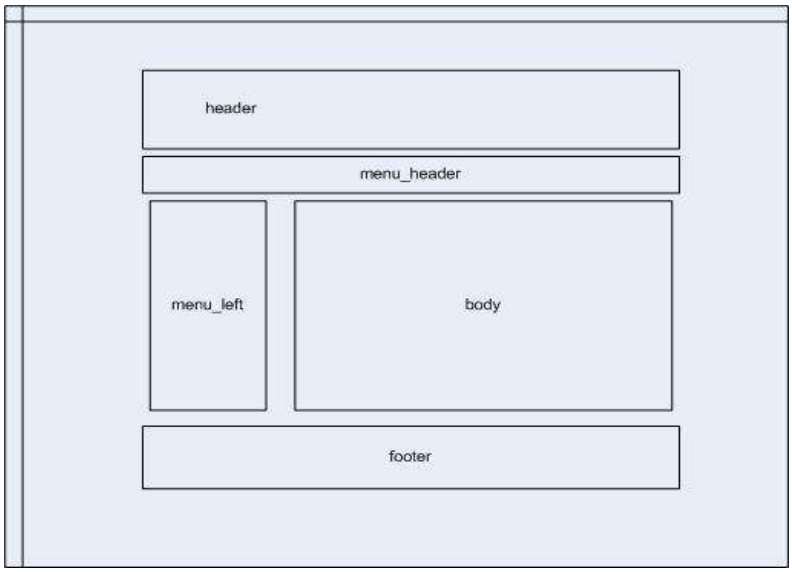


Figura 38.- videoclub-layout-admin.jsp (Menú Administrador)

- **tiles/videoclub-layout-login-admin.jsp:** Representa el menú del login del administrador:
  header: capçelera de la web on situarem el logo de la empresa.
  body: contingut principal on tindrem el formulari de login.
  footer: peu de pàgina amb les dades legals i de contacte de la empresa.

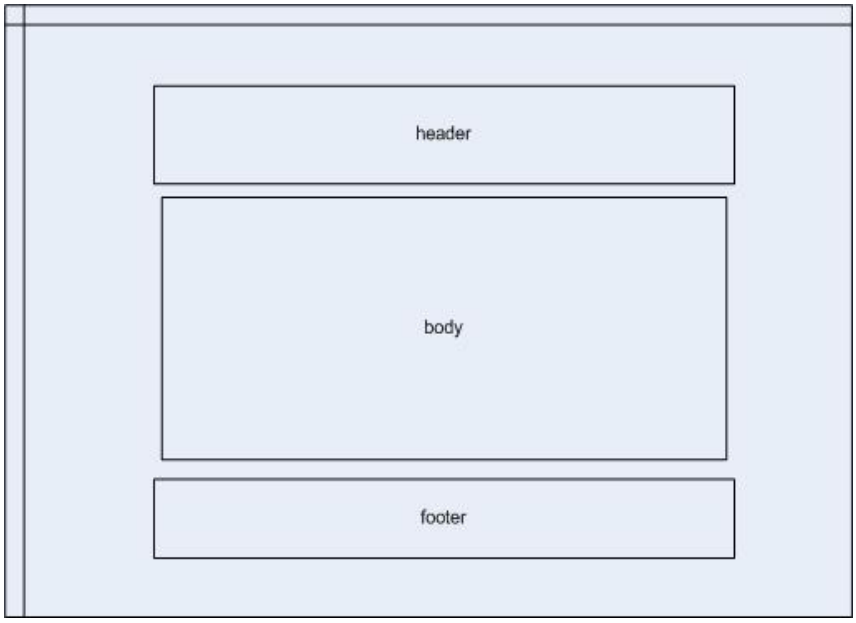


Figura 39.- videoclub-layout-login-admin.jsp (Login d'Admnistrador)

Per utiitzar *Struts Tiles* cal afegir al descriptor a struts-config.xml aquestes línees i crear el fitxer tiles-defs.xml on indiquem els *layouts* a usar.

```
...
<plug-in className="org.apache.struts.tiles.TilesPlugin" >
                <set-property property="definitions-config" value="/WEB-INF/tiles-
defs.xml"/>
...
```

2-3. S'ha escollit *Struts* com a *framework* de la capa de presentació que implementa el patró MVC. Per usar-lo s'ha de crear un fitxer struts-config.xml on es descriuen les diferents accions i navegació de les pantalles i navegació. En aquest cas podem veure que les peticions de Index criden a la action LllistarPeliculesAction.

```
...
<action-mappings>
                <action path="/Index" type="com.actions.LlistarPeliculesAction">
                        <forward name="pelicules_trobades" path="/index.jsp" />
```

```
                    <forward name="sense_pelicules" path="/error.jsp" />
                </action>
...
</action-mappings>
```

4. A les diferents accions és poden usar ActionForms que son un tipus d'accions de *Struts* usades per recuperar i validar dades dels formularis usats i es defineixen també al *struts config*. Aqui podem veure la definició del *bean* formBusqueda que representa el formulari de cerca de pel·lícules.

```
...
<struts-config>
        <form-beans>
                <form-bean name="formBusqueda" type="com.beans.FormBusqueda"/>
    ...
</form-beans>
...
```

5. Per a realitzar les validacions de les dades introduïdes als formularis s'ha de usar els serveis de validació de *struts*. Per fer-ho s'ha de incloure al fitxer struts-config.xml les línees següents i apart afegir els fitxers validator-rules.xml i validation.xml per definir les regles de validació i els formularis que rebran validacions respectivament.

```
...
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,
/WEB-INF/validation.xml"/>
        </plug-in>
...
```

6. Fer ús de les llibreries i18n per a la internacionalització de les aplicacions es senzill. Requereix crear els fitxers ApplicationResources_xx.properties necessaris on xx es el locale del idioma (ca: català, es: espanyol, en: anglès) i per defecte el ApplicationResources.properties que serà l'idioma per defecte. En aquests fitxers tindrem les claus dels idiomes i valor:

```
...
# -- Pantalla Listado Películas -- #
body.pelicula.label.director=Director:
body.pelicula.label.protagonistas=Protagonistas:
...
```

Per altre part cal afegir al fitxer struts-config.xml la entrada **<message-resources parameter="ApplicationResources" />** per a indicar que s'estan fent servir recursos de missatges.

Finalment a les vistes *JSP* farem ús de la llibreria:

```
..
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
..
```

I posteriorment afegirem els diferents missatges a la *JSP* amb la directiva:

```
..
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
..
```

7. Per tal de tractar amb els diferents combos de l'aplicació s'han creat classes seleccionables formats per value y label que representen la etiqueta value y el texte que es mostra en els combos. Podem veure a continuació el de categories:

```
..

public class Categories extends ArrayList implements Serializable {
      public Categories(){}

      public ArrayList llistaCategories(String id_idioma){
            ArrayList llistaCategories = new ArrayList();

            CategoriaDAO categ =
DAOFactory.getCategoriaDAO(TipoDAO.Default);
            ArrayList<CategoriaVO> cat = (ArrayList<CategoriaVO>)
categ.load(id_idioma);

            Categoria categoria = null;

            for (int i=0; i<cat.size(); i++){
                  categoria = new
Categoria(cat.get(i).getIdCategoria().toString(),cat.get(i).getCategoria(
).toString());
                  llistaCategories.add(categoria);
            }

          return llistaCategories;

      }
}
```

```
. .
```

Implementació de la capa de Negoci del Videoclub *online*

En aquesta capa hem usat *Spring* com a *framework* de negoci i per tal de usar-lo hem realitzat el següent:

1.- Configuració de *Spring*.
2.- Comunicació entre la capa de Negoci e integració.
3.- Utilització del Patró Factoria per separar la implementació del tipus de SGBD.

1.- Com ja s'ha comentat hem escollit *Spring* per a gestionar la capa de negoci i la gestió de transaccions. Per tal de configuar-lo s'ha de crear el fitxer descriptor de *Spring* spring.xml. Primerament indicarem el driver a fer servir i la connexió a la BD, substituir les xxx per el usuari i password corresponent:

```
        <?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

        <bean id="datasource"
 class="org.springframework.jdbc.datasource.DriverManagerDataSource">
                <property name="driverClassName" value="com.mysql.jdbc.Driver" />
                <property name="url" value="jdbc:mysql://localhost:3306/video_club" />
                <property name="username">
                  <value>xxx</value>
    </property>
    <property name="password">
      <value>xxx</value>
    </property>
```

A continuació hem de indicar els diferents mapatges de *Hibernate* per comunicar les capes de negoci e integració.

```
        <bean id="sessionFactory"
 class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
```

```
            <property name="dataSource" ref="datasource" />
            <property name="mappingResources">
                  <list>
                        <value>com/videoclub/Admin.hbm.xml</value>
                        ....
.....
```

2. A part de la configuració pròpia de gestió de transaccions i el tema de l'ús de *Hibernate* amb *Spring* s'han de configurar els *beans* que s'utilitzaran per comunicar la capa de negoci amb integració fent ús del patró *DAO*. Per això s'afegeixen les classes que realitzen la implementació per a cada *Bean*:

```
...
<bean id="adminDAO" class="com.model.AdminDAOSpringHBMySQLImpl">
            <property name="sessionFactory" ref="sessionFactory" />
          </bean>.
.....
```

3.- S'ha decidit usar el patró factoria per tal de separar la implementació del SGBD utilitzat. S'ha creat la classe DAOFactory que recull els beans de l'*application context* recollits al fitxer Spring.xml. El mètode accessor obté el *DAO* amb la implementació complerta per el tipus de SGBD.

```
...
public class DAOFactory {

      private static ApplicationContext context;

      static {
            context =
                  new ClassPathXmlApplicationContext("spring.xml");
          }
....
public static AdminDAO getAdminDAO(TipoDAO tipo) {

            switch(tipo)
            {
                  case MySQL:
                        return getMySQLDAO(defaultAdminDAO);
                  case Oracle:
                        return getOracleDAO(defaultAdminDAO);
                  case SqlServer:
                        return getSQLDAO(defaultAdminDAO);

                  default:
                        return getMySQLSpringDAO(defaultAdminDAO);
```

```
            }
          }
.....
```

Implementació de la capa d'Integració del Videoclub *online*

En aquesta capa hem usat Hibernate com a *framework* d'integració i també hem utilitzat el patró de disseny *DAO* per a encapsular els mètodes d'accés a dades.

1.- Mapatge *Hibernate XML*.
2.- Utilització del Patró *DAO.*
3.- Utilització de classe DBManager realització consultes complexes.

1. Disposem del model de dades del Videoclub online *MySQL* i hem optat per utilitzar un mapatge O-R mitjançant fitxers *XML* per les taules i classes *VO* per tal de tractar els valors de forma persistent en classes java. Hem utilitzat el pluguin de *Hibernate Tools* per realitzar el mapatge de les taules. Podem veure un exemple de la taula admin:

Admin.hbm.xml

```
...
<hibernate-mapping>
  <class name="com.videoclub.AdminVO" table="admin" catalog="video_club">
    <id name="id" type="java.lang.Integer">
      <column name="id" />
      <generator class="identity" />
    </id>
    <property name="usuari" type="string">
      <column name="usuari" length="100" not-null="true" unique="true" />
    </property>
    <property name="password" type="string">
      <column name="password" length="100" not-null="true" />
    </property>
  </class>
</hibernate-mapping>
.....
```

AdminVO.java

```
...
public class AdminVO implements java.io.Serializable {
        private static final long serialVersionUID = 1L;
        private Integer id;
        private String usuari;
        private String password;

        public AdminVO() {
```

```
        }

        public AdminVO(String usuari, String password) {
                this.usuari = usuari;
                this.password = password;
        }

        public Integer getId() {
                return this.id;
        }

        public void setId(Integer id) {
                this.id = id;
        }

        public String getUsuari() {
                return this.usuari;
        }

        public void setUsuari(String usuari) {
                this.usuari = usuari;
        }

        public String getPassword() {
                return this.password;
        }

        public void setPassword(String password) {
                this.password = password;
        }

}
.....
```

2. S'ha decidit fer ús del Patró *DAO* per a la capa de integració i això requereix crear per les diferents taules els objectes *DAO* que encapsulin els diferents mètodes i definir les classes que realitzen la implementació. Exemple:
   AdminDAO.java

```
...
public interface AdminDAO {

        public List<AdminVO> load();
        public void save(AdminVO a);
        public void delete(AdminVO a);
```

```
        public void update(AdminVO a);
        public AdminVO load(String usuario, String clave);
}
.....
```

AdminDAOSpringHBMySQLImpl.java

```
...
public class AdminDAOSpringHBMySQLImpl extends HibernateDaoSupport implements
AdminDAO{
                private SessionFactory sessionFactory;
....
@Override
        public AdminVO load(String usuario, String clave) {
                Session session = getSessionFactory().openSession();
                Criteria criteria = session.createCriteria(AdminVO.class);
                criteria.add(Restrictions.eq("usuari", usuario));
                criteria.add(Restrictions.eq("password", clave));
                if(!criteria.list().isEmpty()){
                        AdminVO admin =  (AdminVO) criteria.list().get(0);
                        session.close();
                        return admin;
                }else{
                        session.close();
                        return null;
                }
              }

.....
```

Certes consultes que s'havien de realitzar requerien de consultes complexes en diferents taules degut al model de dades utilitzat i per això s'ha optat per recuperar un llistat mitjançant *JDBC* realitzant les consultes necessaries doncs és més senzill de utilitzar que *Hibernate*. Exemple de cerca de pel·lícules:

PeliculaDAOSpringHBMySQLImpl.java

```
...
public class PeliculaDAOSpringHBMySQLImpl extends HibernateDaoSupport implements
                PeliculaDAO {
…

public List<Pelicula> load(String camp, String idGenere, String idIdioma, String cerca)
throws IOException {
                String sql_query = "SELECT article.id_article, article.imatge_petita,
desc_article.titol_article, "
                                + "  desc_article.descripcio, pelicula.duracio_pelicula,
```

```
pelicula.trailer, pelicula.mostra_trailer, director.director, "
                                + "interpret.interpret, interpret_2.interpret,
interpret_3.interpret, interpret_4.interpret, genere.genere as genere1, "
                                + "genere_2.genere as genere2, genere.id_idioma "
                                + "FROM ( ( ( ( ( ( ( video_club.pelicula pelicula"
                                + " LEFT JOIN"
                                + " video_club.interpret interpret"
                                + " ON (pelicula.id_interpret_1 = interpret.id_interpret))"
                                + " LEFT JOIN"
                                + " video_club.interpret interpret_2"
                                + " ON (pelicula.id_interpret_2 = interpret_2.id_interpret))"
                                + " LEFT JOIN"
                                + " video_club.interpret interpret_3"
                                + " ON (pelicula.id_interpret_3 = interpret_3.id_interpret))"
                                + " LEFT JOIN"
                                + " video_club.interpret interpret_4"
                                + " ON (pelicula.id_interpret_4 = interpret_4.id_interpret))"
                                + " LEFT JOIN"
                                + " video_club.article article"
                                + " ON (pelicula.id_pelicula = article.id_article))"
                                + " LEFT JOIN"
                                + " video_club.desc_article desc_article"
                                + " ON (desc_article.id_article = article.id_article))"
                                + " LEFT JOIN"
                                + " video_club.genere genere"
                                + " ON (pelicula.id_genere_1 = genere.id_genere))"
                                + " LEFT JOIN"
                                + " video_club.genere genere_2"
                                + " ON (pelicula.id_genere_2 = genere_2.id_genere))"
                                + " LEFT JOIN"
                                + " video_club.director director"
                                + " ON (pelicula.id_director = director.id_director)"
                                + " WHERE (desc_article.id_idioma = "+idIdioma+""
                                + ") AND ((pelicula.id_genere_1 = "+idGenere+""
                                + ")"
                                + " OR (pelicula.id_genere_2 = "+idGenere+""
                                + ")) AND ("+camp+""
                                + " LIKE '%"+cerca+"%')";
                ArrayList<Pelicula> pelis = (ArrayList<Pelicula>)
DBManager.cercaPelicules(sql_query);
                return pelis;
        }

.....
```

DBManager.java (Implementa *Singleton* per recuperar instancia única DB).

```
...
public class DBManager {

	private static Connection conn = null;
...

	public Connection getConexion() {
		if (conn != null)
			return conn;
		else {
			try {
				init();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			return conn;
		}
	}
.....
public static Collection cercaPelicules(String SQL) throws IOException {
		Statement stmt = null;
		ResultSet rs = null;

		List<Pelicula> pelicules = new ArrayList<Pelicula>();

		if (conn == null) {
			init();
		}
		try {
			stmt = conn.createStatement();

			rs = stmt.executeQuery(SQL);

			while (rs.next()) {
				Pelicula peli = new Pelicula();
				peli.setId_article(rs.getString("article.id_article"));

	....
....
```

Implementació de la Llibreria de *tags*

S'ha decidit realitzar una sèrie de accions o *tags* per tal de realitzar funcionalitats que ajudin en el desenvolupament de les vistes del Videoclub *online* i concretament s'ha decidit realitzar les accions descrites a continuació:

1.- Realitzar un llistat de manteniment sobre les taules de la B.D.
2.- Realitzar la presentació de un catlàleg d'elements tipus (Pel·lícula, Joc..) de un videoclub *online*.
3.- Realitzar un *slider* per mostrar novetats, ofertes...

Per totes elles s'han realitzat una serie de accions. Existeixen 2 llibreries una anomenada vcopagertaglib que és la llibreria *Pager Tag Lib* adaptada a les necessitats del projecte i per altre part la llibreria vcotaglib la llibreria pròpia del Videoclub *online*.

Les 2 llibreries tenen els seus fitxers *TLD* associats i que son **pager-taglib.tld** i **vco-taglib.tld** inclosos a la carpeta WEB-INF/tld.

Apart s'ha d'indicar al fitxer web.xml (descriptor web) la utilització d'aquestes llibreries:

```
...
<jsp-config>
		<taglib>
			<taglib-uri>vcopagertaglib</taglib-uri>
			<taglib-location>/WEB-INF/tld/pager-taglib.tld</taglib-location>
		</taglib>
		<taglib>
			<taglib-uri>vcotaglib</taglib-uri>
			<taglib-location>/WEB-INF/tld/vco-taglib.tld</taglib-location>
		</taglib>
	</jsp-config>

.....
```

Per a realitzar el seu ús en les *JSP* haurem d'incloure-les amb les directives:

```
<%@ taglib uri="WEB-INF/tld/pager-taglib.tld" prefix="pg"%>
<%@ taglib uri="WEB-INF/tld/vco-taglib.tld" prefix="vco"%>
```

1.- Per realitzar el llistat de manteniment s'han realitzat 3 accions mantLlistat, mantItem i mantAction.

mantLlistat: és la acció que representa el llistat de resultats i concretament les capçaleres i realita les iteracions sobre la taula.
mantItem: representa la fila actual amb les dades de cada columna de la taula de manteniment.

mantAction: representa la acció que es desitja realitzar de edició o esborrat o la acció que es desitgi sobre el element de la fila.

Les accions mantLlistat i mantItem son obligatòries i mantItem és filla de mantLlistat i per tant ha d'estar inclosa dins el seu cos. Per altre part mantAction és una acció filla de mantItem.

Podem veure un exemple de la seva utilització:

```
...
<vco:mantLlistat
                items="${sessionScope['manteniment_titols']}" width="8%"
                styleClass="headerTable">
                <vco:mantItem list="${sessionScope['manteniment_pelicules']}"
                        fields="${sessionScope['camps']}" oddClass="oddTable"
                        evenClass="evenTable" mouseOverClass="mouseOverTable"
pagination="true"
                        var="item">
                                <vco:mantAction action="editPelicula.do" id="id_pelicula"
                                        label="Editar" actionIcon="img/edit.png" />
                                <vco:mantAction action="deletePelicula.do" id="id_pelicula"
                                        label="Borrar" actionIcon="img/delete.png" />
                </vco:mantItem>
</vco:mantLlistat>
.....
```

2.- Per realitzar la presentació del catàleg d'elements s'han desenvolupat 2 accions catàleg i catalegElem. Les 2 accions son obligatòries i catalegElem és una acció filla de cataleg.

cataleg: representa el catàleg amb el llistat i estil a mostrar.

catalegElem: representa un element del catàleg amb el seu estil format i camps a mostrar.

Podem veure un exemple de la seva utilització:

```
...
<vco:cataleg
                items="${sessionScope['llistat_titols']}" styleClass="ficha_fichas">
                <vco:catalegElem list="${sessionScope['llistat_pelicules']}"
```

```
                        fields="${sessionScope['tagFields']}" titleField="titol"
                        imageAction="ServletImages"
detallAction="LlistarDetallPelicula.do"
                        pagination="true" id="id_pelicula" imageStyleClass="ficha_imagen"
var="item">
                    </vco:catalegElem>
                </vco:cataleg>
.....
```

3.- Per realitzar el *slider* per mostrar novetats, noticies... s'han desenvolupat 2 accions novetats i novetatsElem .

novetats: representa l'element principal que aplica el estil del *slider* i iterar sobre el llistat

novetatsElem: representa el element a mostrar dins del *slider* amb els seus camps i format.

Podem veure la seva utilització en aquest exemple:

```
...
<vco:novetats
                    items="${sessionScope['llistat_titols_novetats']}"
                    styleClass="panelContainer" timeout="6000">
                    <vco:novetatsElem list="${sessionScope['llistat_novetats']}"
                            fields="${sessionScope['tagFieldsNov']}" titleField="titol"
                            descField="descripcio" imageAction="ServletImages"
                            detallAction="LlistarDetallPelicula.do" id="id_pelicula"
                            imageStyleClass="imagen" trailerAction="#"
                            trailerIcon="img/icono-triler.gif" var="item">
                    </vco:novetatsElem>
                </vco:novetats>
.....
```

Nota: En el annex final podrem obtenir detall de les diferents accions i atributs així com de les llibreries i dependències necessàries per al seu ús.

Resultats obtinguts

A continuació si revisem l'aplicació del disseny parcial obtingut i l'aplicació de les llibreries de tags per al videoclub online hem obtingut les funcionalitats que a continuació es mostren.

Pantalla Inicial – VideoClub *online.*

Podem observar la pantalla inicial on tenim els menús laterals per realitzar diferents cerques per diferents criteris i amb diferents apartats de navegació estructurats segons el disseny realitzat:

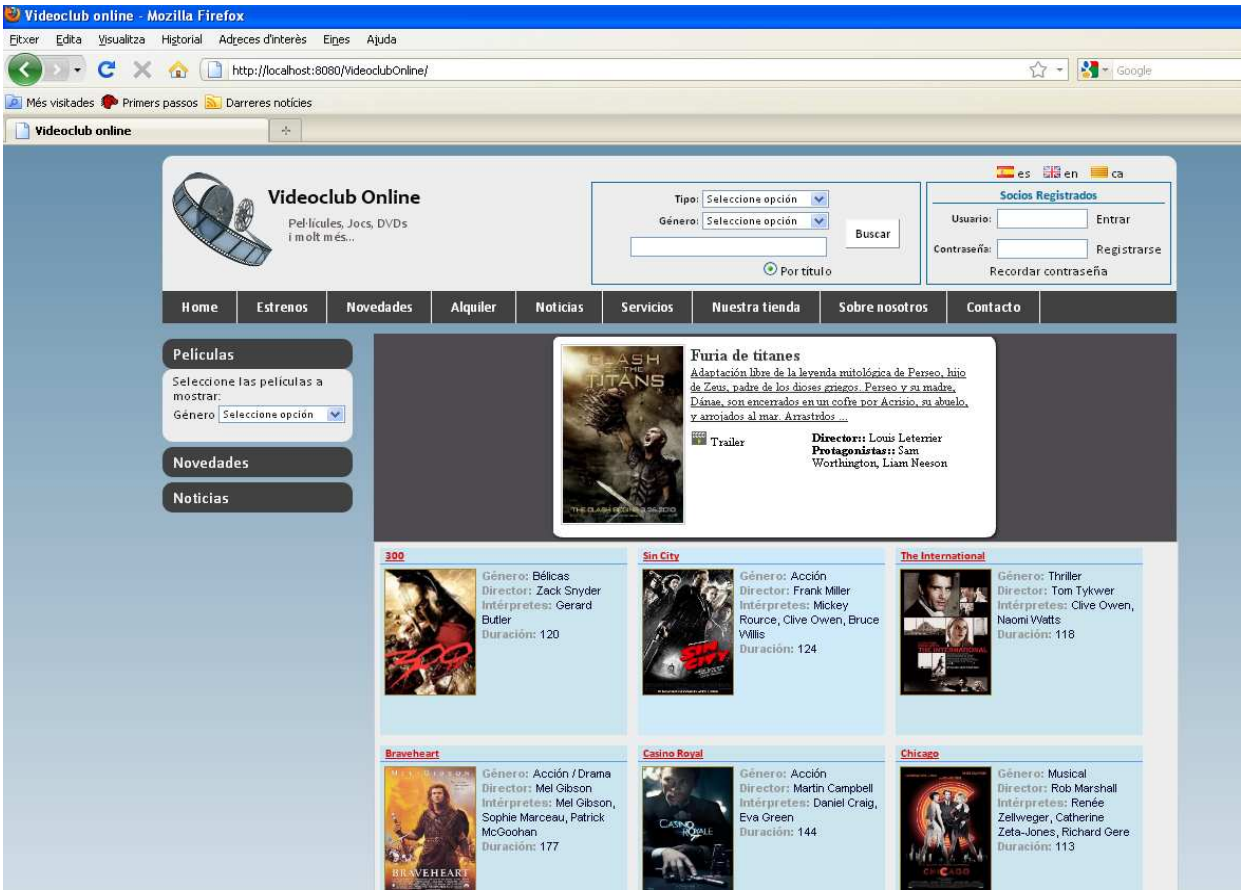


Figura 40.- Pantalla Inicial Videoclub online

En la part inferior de la pantalla disposarem del menú de navegació sobre els resultats:

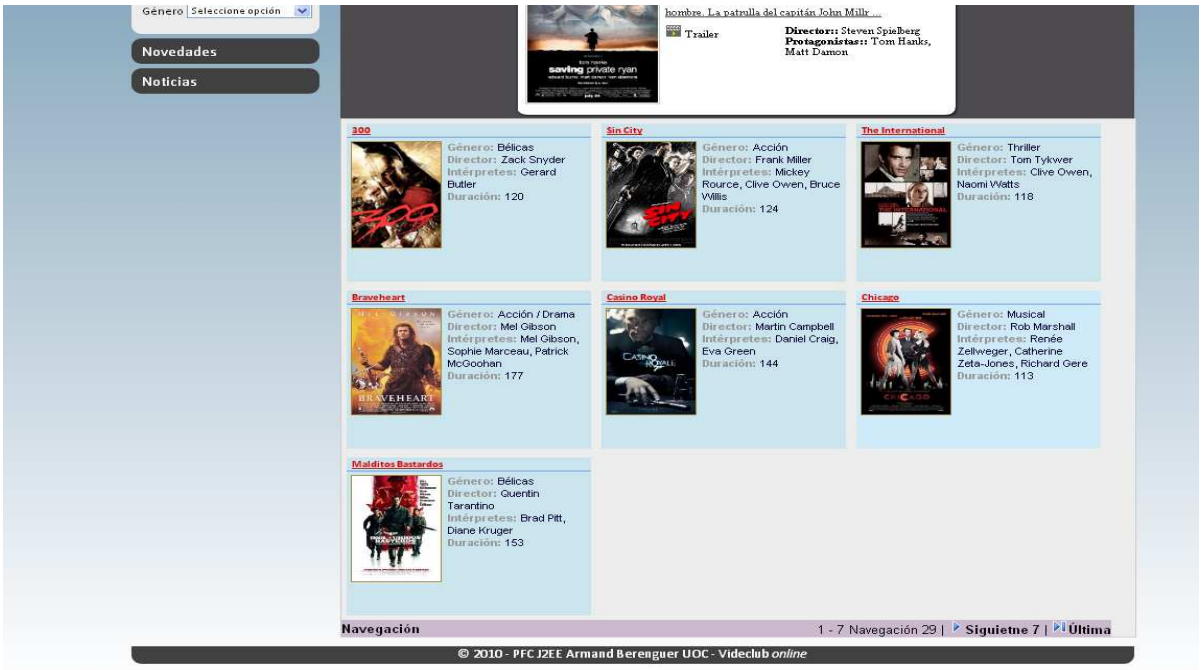


Figura 41.- Part inferior amb menú de navegació

Cerca per diferents criteris en aquest cas per Gènere:

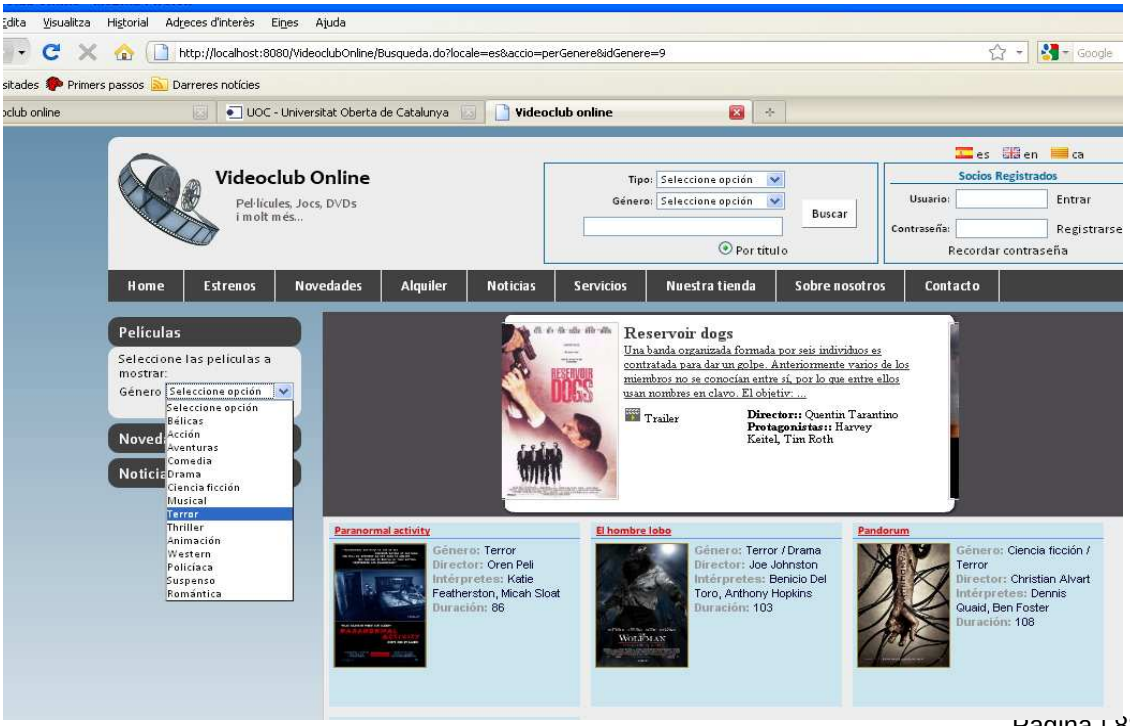


Figura 42.- Cerca per género

Detall dels exemplars:

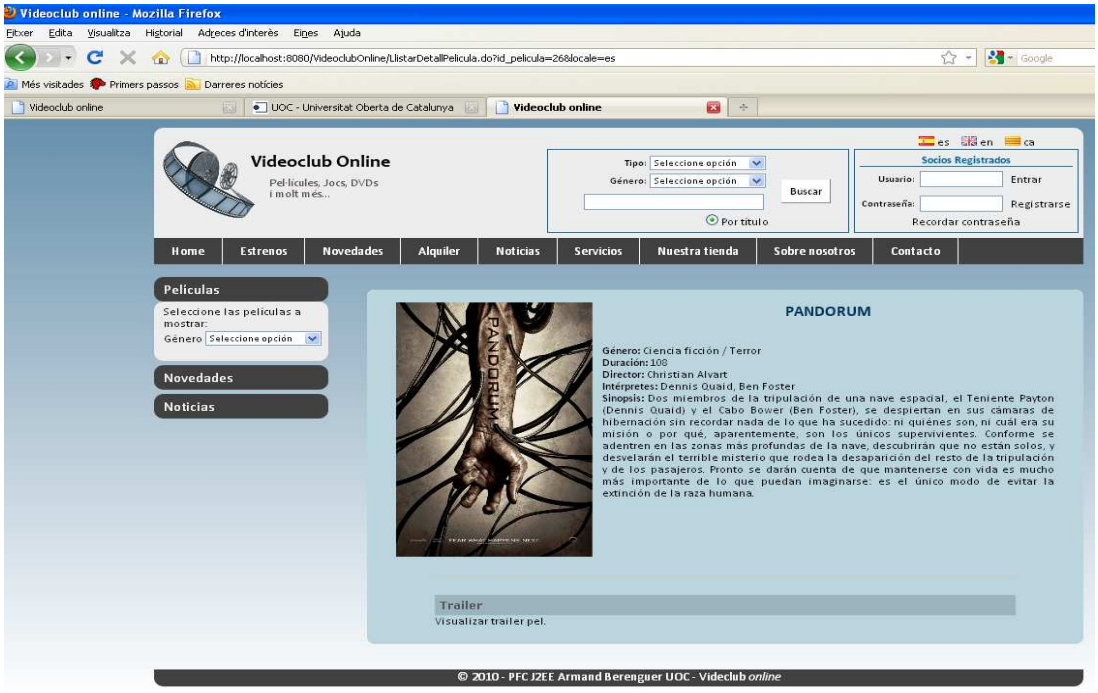


Figura 43.- Detall exemplar

Llistat de menús de manteniment de l'aplicació:

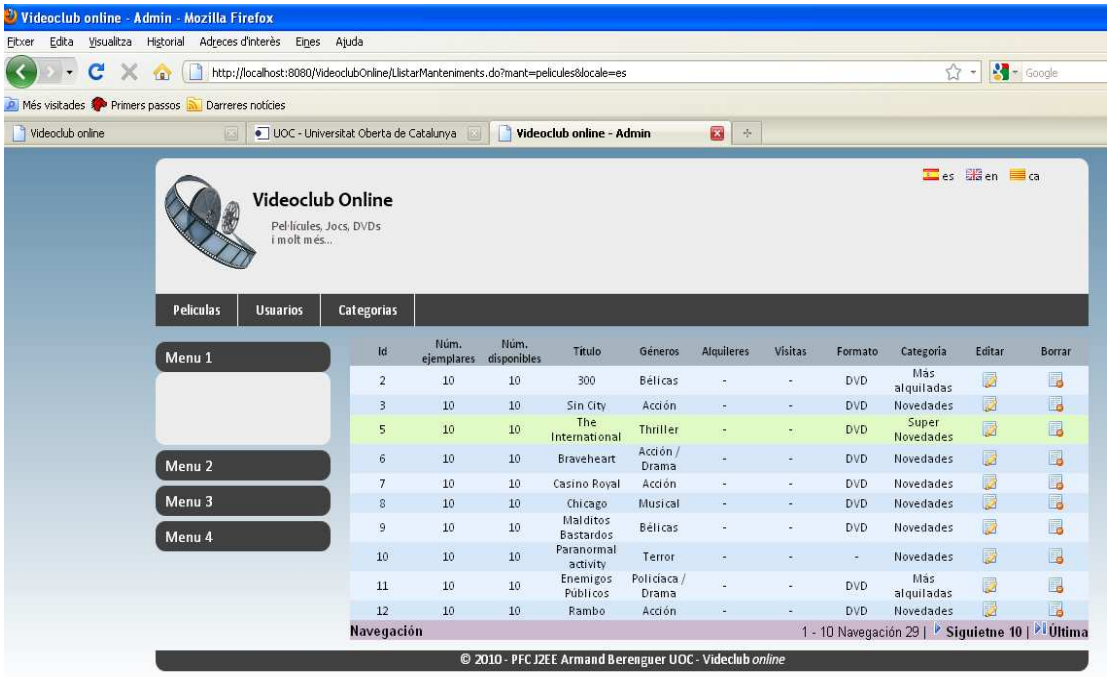


Figura 44.- Llistat de manteniment

Capítol 6: Conclusions

Conclusions

La primera conclusió important ha estat la de realitzar un bon anàlisi i la realització de un bon estudi previ al desenvolupament d'aplicacions a nivell d'arquitectura seleccionant aquelles eines i patrons de disseny que millor s'adaptin a les nostres necessitats.

La segona conclusió important ha estat la de els avantatges de la utilització e integració de diferents *frameworks*.

Hem pogut observar com la interactuació entre diferents *frameworks* aconsegueix obtenir millors resultats en el desenvolupament de l'aplicació. Per exemple *Struts Tiles* afegeix una nova funcionalitat a *Struts* agilitzant el desenvolupament de les vistes i menús. Com Spring afegeix noves funcionalitats a *Hibernate* per tal de gestionar les sessions i les transaccions.

Una altra conclusió important obtinguda és de les avantatges de utilitzar llibreries de *tags* en la capa de presentació i com s'utilitzen, com permeten realitzar certs desenvolupaments i resoldre problemes que les accions estàndard no resolen adaptant-los a les nostres necessitats.

Una altre conclusió és que no totes les eines, *frameworks* i utilitats donen suport i resolen totes les nostres necessitats sempre és possible que haguem de realitzar certes adaptacions o personalitzacions.

Finalment sobre les llibreries de *tag* s'ha desenvolupat una llibreria que es configurable a diferents estils i pot ser reutilitzada per altres aplicacions agilitzant el desenvolupament de certes accions.

Glossari

J2EE (Java Second Enterprise Edition): Estàndard desenvolupat per la empresa Sun Microsystems ara pertanyent a Oracle per al desenvolupament d'aplicacions empresarials en entorns distribuïts.

Framework: Marc de treball reutilitzable orientat a agilitzar el desenvolupament d'aplicacions i reutilitzar el codi.

Design Pattern (patró de disseny): Plantilla que es fa servir com a disseny per a una solució de un determinat problema durant el desenvolupament.

Online: Sistema que està fent ús de internet. Pot tractar-se de persones, sistemes, aplicatius web o dispositius que fan us de la xarxa de internet.

Capa de presentació: També anomenada **capa web** en el model *J2EE* la seva principal funció és separar la interfície de la lògica de negoci i facilitar el accés de les peticions dels diferents clients a la lògica de negoci.

Thin Client (Client lleuger): Ordinador amb uns requeriments de hardware mínims que permet connectar a internet i realitzar peticions web, pot ser un ordinador personal (PC), portàtil o dispositiu de baix rendiment.

Capa de negoci: Aquesta capa en el model *J2EE* és la capa més important doncs encapsula tota la lògica de negoci de l'aplicació. Una de les funcions més importants és la de separar la presentació del accés a dades i per tant no estar afectada per aquestes capes.

Capa de dades: Aquesta capa en el model *J2EE* és la encarregada d'emmagatzemar de forma persistent les dades de l'aplicació al SGBD sobrevivint a l'execució de l'aplicació.

Tag: Etiqueta o marca de un llenguatge basat en *XML* com per exemple *HTML*. Les etiquetes permeten associar atributs i propietats que configuren el element.
Per el que fa a les ***llibreries de tags*** entenem com un conjunt d'etiquetes reutilitzables i encapsulades per a ser utilitzades en diferents aplicacions en el desenvolupament de les vistes de la capa de presentació. En quant a ***Custom Tag Libs*** entenem com llibreries personalitzades que encapsulen una sèrie de funcionalitats davant de unes necessitats concretes i ajuden en el desenvolupament.

MVC (Model-View-Controller): Patró de disseny de programari que separa la lògica de control de l'aplicació (Controller o Controlador) , de la lògica de negoci (Model) i la lògica de presentació (View o Vista).

Struts: Framework de presentació open source desenvolupat per Apache i que implementa el patró de disseny MVC.

Struts Tiles: *Framework* creat per Apache Software Foundation per tal de realitzar les vistes de l'aplicació web seguint el patró de Vista Composta (*Composite View*) per tal de jerarquitzar e independitzar la estructura de la web en diferents vistes reutilitzables.

JSF (Java Server Faces): *Framework* per a la capa de presentació basat en components reutilitzables desenvolupat per Sun Microsystems.

Spring MVC: *Framework* per a la capa de presentació desenvolupat dins el projecte *open source Spring*.

OOP (Object-Oriented Programming): paradigma de programació que defineix els programes en termes de classes d'objectes i en ells encapsula estat (dades), comportament (mètodes i prodeciments) i identitat.

API (Application Programming Interface): conjunt d'especificacions de comunicació entre components de programari que proporciona abstracció entre els nivells inferiors i superiors d'una aplicació.

AOP (Aspect-Oriented Programming): paradigma de programació orientada a aspectes que pretén aconseguir una modularització apropiada i una millor separació de conceptes, que disminueixi o elimini les dependències inherents entre diferents mòduls.

IoC (Inversion of Control): principi de la programació orientada a objectes pel qual es redueix l'acoblament inherent en els programes informàtics. El control es passa de l'aplicació al *framework*.

Spring: Comunitat de desenvolupadors que han desenvolupat un *framework* propi *open source* que aporta solucions a cada una de les capes i aplica el principi de inversió de control mitjançant la tècnica de injecció de dependències.

Hibernate: *Framework* de persistència *open source* desenvolupat per Red Hat ampliament utilitzat per a realitzar la persistència d'objectes java a la Base de Dades.

HQL (Hibernate Query Language): llenguatge de consulta de dades creat i usat per Hibernate que ofereix una *API* per construir les consultes programàticament anomenada "criteria".

JPA: *API* de persistència desenvolupada per a la plataforma *J2EE* i que ha estat adoptada per Sun Microsystems com estàndard per als *EJB* 3.0.

iBatis (MyBatis): *Framework* de codi obert i desenvolupat per Apache Software Foundation per a la capa de persistència.

Servlet: Classe java que rep peticions HTTP i genera contingut dinàmic com a resposta a aquestes.

JSP (Java Server Pages): Documents de text que s'executen com a *servlets* però que permeten una aproximació més natural a la creació de contingut estàtic. També s'anomena **vistes** doncs les *JSP* implementen les vistes del model MVC en la capa de presentació.

JSTL (JSP Standard Tag Library): component de la especificació *J2EE* de Sun Microsystems que encapsula les funcionalitats principals per escriure vistes *JSP*.

Tomcat: Contenidor de *Servlets* i *JSP* desenvolupat per el projecte de Apache Software Foundation.

MySQL: Sistema gestor de bases de dades (SGBD), multi-usuari, multi-plataforma i de codi obert.

IDE (Integrated Development Environment): entorn integrat de desenvolupament que incorpora diferents eines (editor, compilador....) que faciliten la tasca del programador.

TLD (Tag Library Descriptor): Són descriptors de llibreries de *tags*, fitxers de configuració *XML* que defineixen les característiques dels tags o accions utilitzats en la llibreria com són els seus atributs, el tipus d'accions.

Slider: És un element de les nostres vistes que permet ser seleccionat i realitzar certes accions de forma dinàmica. (El *slider* usat al PFC canvia d'element dinàmicament mitjançant un temps determinat)

Layout: Representa la estructura amb la que és mostrarà la informació a la pàgina web. Hi hauràn diferents regions i apartats diferenciats i tot això es resumeix en esquemes de pàgina.

Template: Els templates son les plantilles que defineixen en *Struts Tiles* una estructura de planes i pensades per a ser reutilitzades posteriorment en diferents apartats del lloc web.

Ajax (Asynchronous JavaScript And XML): técnica de desenvolupament web per crear aplicacions interactives que s'executen al navegador mantenint una comunicació asíncrona amb el servidor en segon pla. (Es realitzen canvis a les planes web sense recarregar-les).

JQuery: biblioteca o *framework* de JavaScript creada per John Resig que permet desenvolupar animacions, interactuar amb els elements del document i usar *AJAX* a les planes web.

Bean: Component de software reutilitzable. Disposen de una interface pública per tal de instanciar-los i uns mètodes per accedir als atributs i insertar valors als atributs.

VO (Value Object, també anomenat DTO o Data Transfer Object): Son simplement *beans* contenidors de dades sense cap lògica. S'usen per enviar informació entre diferents capes de l'aplicació.

ORM (Object-Relational Mapping): Técnica de programació per convertir dades del tipus utilitzat en el llenguatge de programació orientat a objectes i les dades usades a una base de dades relacional. S'utilitza un *framework* per a realitzar aquestes tasques.

JDBC (Java DataBase Connectivity): *API* de Java que indica al client com s'ha de connectar a una base de dades relacional.

DAO (Data Access Object): Patró de disseny utilitzat per a la capa de persistència realitzant la abstracció i encapsulament els mètodes d'accés a dades.

Pool de Connexions (Connection Pool): Tècnica usada per tal de mantenir un conjunt de connexions actives a la BD i gestionar de manera més eficient el accés a dades. S'utilitza amb ***JNDI (Java Naming and Directory Interface)*** unes *API* de Java per definir serveis de directoris i localització de noms per accedir al *pool* definit.

CSS (Cascading style sheets): llenguatge formal utilitzat per a definir l'estil d'un document *html* o *xml*.

EJB (Enterprise Java Bean): *API* per a *J2EE* implementada per Sun Microsystems que ofereix un model de components del costat del servidor per a desenvolupar la lògica de negoci en aplicacions distribuïdes solventant la problemàtica de gestió de la concurrència, transaccions, seguretat..

POJO (Plain Old Java Object): classe definida en Java que no és de cap tipus especial (*EJBs*), ni implementa cap interfície específica.

XML (eXtensible Markup Language): metallenguatge extensible que permet definir la gramàtica de llenguatges específics. Estàndard per a l'intercanvi d'informació estructurada entre diferents plataformes.

Open Source: Codi obert, amb aquest terme es coneix el software que es distribuït i desenvolupat llibrement per la comunitat.

HTML (Hyper Text Markup Language): llenguatge de marques utilitzat per el desenvolupament de planes web.

URL (Uniform Resource Locator): Seqüència de caràcters usada seguint un format estàndard per anomenar recursos, documents e imatges a internet. S'utilitza per tal de poder localitzar-los.

BLOB (Binary Large Object): Elements utilitzats a la base de dades per emmagatzemar dades de gran tamany que varien de forma dinàmica. Poden ser fitxers, imatges..

Bibliografia

Camps i Riba, Josep Maria: J2EE. Una plataforma de components distribuïda
Material UOC. (FUOC • P06/11059/01150)

Pradel i Miquel, Jordi / Raya Martos, José Antonio: Catàleg de patrons
Material UOC. (Universitat Oberta de Catalunya • P05/11058/00504 • Mòdul 2)

Hall, Marty: core Servlets and JavaServer Pages
Prentice Hall & Sun Microsystems

Llibres

Bergsten, Hans: Java Server Pages (2ond Edition)
O'Reilly, 2002
ISBN: 0-596-00317-X

Johnson, Rod / Hoeller, Juergen: Expert One-on-One J2EE Development without EJB
Wiley Publishing Inc, 2004
ISBN: 978-0-7645-5831-3

Bayern, Shawn: JSTL in Action
Manning, 2003
ISBN: 1-930110-52-9

Goodwill, James / Hightower, Richard: Professional Jakarta Struts
Wrox Press, 2004

ISBN:0764544373

Cavaness, Chuck: Jakarta Struts
O'Reilly, 2002
ISBN :0596003285

Doray, Arnold: Beginning Apache Struts: From Novice to Professional
Apress, 2006
ISBN : 1590596048

Bergsten, Hans: Java Server Faces
O'Reilly, 2004
ISBN: 0-596-00539-3

Mann, Kito D.: Java Server Faces in Action
Manning, 2005
ISBN: 1-932394-11-7

Ladd Seth / Donald, Keith.: Expert Spring MVC and Webflows
Appres, 2006
ISBN: 978-1-59059-584-8

Johnson, Rod / Hoeller, Juergen / Arendsen, Alef / Risberg, Thomas / Sampaleanu, Colin:
Professional Java Development with the Spring Framework
Wiley Publishing Inc, 2005
ISBN:0764574833

Begin, Clinton/ Goodin, Brandon / Meadors, Larry: iBatis in Action
Manning, 2007
ISBN: 1-932394-82-6

Bauer, Christian / King, Gavin: Java Persistence with Hibernate revised edition of Hibernate in Action
Manning, 2007
ISBN: 1-932394-88-5

Llocs web

Oracle (darrera consulta Novembre 2010). J2EE Patterns Catalog.
URL: http://www.oracle.com/technetwork/java/catalog-137601.html

Autentia Formación y desarrollo (Java, J2EE, UML, XML, etc.). Tutoriales sobre nuevas tecnologías.
URL: http://www.adictosaltrabajo.com/index.php?pagina=home

The Jakarta Project: Taglibs
http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html

The Struts Taglib
http://struts.apache.org/1.x/struts-taglib/

Annexos

Versions de Programari

Com a gestor de bases de dades MySQL 5.1
(http://www.mysql.com/downloads/mysql/5.1.html)

Com a servidor Web i contenidor de servlets i JSP Tomcat 6.0
(http://tomcat.apache.org/download-60.cgi)

Com a versio de java JDK 1.6_14
(https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jre-6u14-oth-JPR@CDS-CDS_Developer)

Com a IDE de desenvolupament Eclipse Galileo (Conte ja molts pluguins)
(http://www.eclipse.org/galileo/)

Com a eina d'explotació de dades i reporting Toad for MySQL 5.0
(http://www.toadworld.com/DOWNLOADS/Freeware/ToadforMySQLFreeware/tabid/561/Default.aspx)

Llibreries del Projecte

El projecte requereix incloure aquestes llibreries a la carpeta WEB-INF/lib del nostre projecte:

La versió d'*Struts* usada és la 1.3.10 i per descarregar-les es poden obtenir d'aquest mirror de la Web de Apache.

http://apache.rediris.es/struts/library/struts-1.3.10-lib.zip

La versió actual de les llibreries de tags Standard i *JSTL*

http://archive.apache.org/dist/jakarta/taglibs/standard/jakarta-taglibs-standard-current.zip

La versió 3.3.0 de la llibreria de *Hibernate*
http://sourceforge.net/projects/hibernate/files/hibernate3/3.3.0.GA/

Descarreguem la versió 2.5.6 del *framework Spring*
http://www.springsource.org/download

Descarreguem el conector per Java de *MySQL JDBC* connector
http://www.mysql.com/downloads/connector/j/

Llibreria de tags VcoTagLib

Aquesta llibreria ha estat desenvolupada per tal de agilitzar certes accions a les vistes de l'aplicació web del VideoClub Online. La llibreria s'ha d'incloure en la carpeta WEB-INF/lib del nostre projecte es **vcotaglib_1_0.jar**. Aquesta llibreria té certes dependències externes amb la llibreria estàndard *Pager Tag Lib* (Realització de paginació sobre resultats) existent a http://jsptags.com/tags/navigation/pager/download.jsp. No obstant com la llibreria VcoTagLib requereix del ús d'aquesta llibreria s'han adaptat certs mètodes i funcionalitats i per això s'ha creat el **vcopagertaglib_1_0.jar** també a incloure al nostre WEB-INF/lib que conté les funcionalitats estàndards de la llibreria i també permeten la comunicació amb aquesta. Per a més informació del seu ús consultar la web esmentada anteriorment.
Una altre dependència que té és l'ús de llibreries Javascript. La llibreria que requereix és la de *JQuery* y els pluguin d'*Slider* per realitzar un *slider* d'articles o productes.
Les llibreries les podem trobar a http://docs.jquery.com/Release:jQuery_1.2 (Ver. 1.2 de *JQuery* versió min, és la feta servir). La versió 1.1.1 del *Pluguin Coda Slider* per tal de realitzar el *Slider* i que conté els fitxers javascript necessaris. La ruta de instal·lació és la següent http://www.ndoherty.biz/demos/coda-slider/1.1.1/coda-slider.1.1.1.zip

La llibreria VcoTagLib disposa de una sèrie de *tags* que permeten realitzar aquestes accions:

1.- Realitzar un llistat de manteniment sobre les taules de la B.D.
2.- Realitzar la presentació de un catàleg d'elements tipus (Pelicula, Joc..) de un videoclub online
3.- Realitzar un *slider* per mostrar novetats, ofertes...
S'ha desenvolupat una sèrie de tags:

Tags Llistats de Manteniment

Tag mantLlistat

Ús Llibreria:	<%@ taglib uri="WEB-INF/tld/vco-taglib.tld" prefix="vco"%>
Definició tag :	vco:mantLlistat
Exemple d'ús:	vco:mantLlistat items="${sessionScope['manteniment_titols']}" width="8%" styleClass="headerTable"

Aquest *tag* és obligatori com a element pare per la resta de tags.

Atribut	Descripció
items	Obligatori. Llista que mostra els títols de les capçaleres de la taula del llistat de manteniment.
styleClass	Opcional. Aquesta serà la classe d'estils *CSS* que s'aplicarà a la capçalera de la taula de manteniment.
width	Opcional. Indicar l'amplada que volem que tinguin les capçaleres i elements del llistat.
height	Opcional. Indicar la alçada que volem que tinguin les capçaleres i elements al llistat.

Tag mantItem

Aquest *tag* és un *tag* fill de mantLlistat i ha d'estar inclòs en aquest. Representa una fila del llistat de manteniment i per tant un element del llistat i és obligatori.

Ús Llibreria:	<%@ taglib uri="WEB-INF/tld/vco-taglib.tld" prefix="vco"%>
Definició tag :	vco:mantItem
Exemple d'ús:	vco:mantItem list="${sessionScope['manteniment_pelicules']}" fields="${sessionScope['camps']}" oddClass="oddTable" evenClass="evenTable" mouseOverClass="mouseOverTable" pagination="true" var="item"

Atribut	Descripció
list	Obligatori. Representa un llistat amb els elements sobre els que es fa el manteniment. Serà un vector de Objectes de tipus *VO* o la classe que vulguem fer servir que representi els elements i camps a tractar. Nota: Per utilitzar-la requereix de una classe *VO* on els diferents atributs tinguin els seus mètodes accessors i també els atributs siguin públics.
fields	Obligatori. Llistat amb els noms dels camps a mostrar en el llistat i a accedir per obtenir els seus valors.
oddClass	Opcional. Indicar la classe del estil *CSS* per a mostrar en les files senars.
evenClass	Opcional. Indicar la classe del estil *CSS* per a mostrar a les files parells.
mouseOverClass	Opcional. Indica la classe i estil *CSS* que aplicarà a la fila quan situem el ratolí en aquella fila.
var	Obligatori. Nom de la variable per a l'element actual que recupera aquest element per poder usar-lo després amb altres *tags* o elements al llistat.
pagination	Opcional. Indicar si es vol realitzar paginació sobre els llistats de manteniment no. En el cas de indicar el paràmetre a "true" aquest s'utilitza amb el *tag*

	pg:pager on es defineixen els elements a mostrar per pàgina i elements totals. Més informació a: http://jsptags.com/tags/navigation/pager/pager-taglib-2.0.html

Tag mantAction

Aquest *tag* és un *tag* fill de mantItem i ha d'estar inclòs en aquest. Una acció del llistat (Editar, Consultar, Esborrar..) és un *tag* opcional.

Ús Llibreria:	<%@ taglib uri="WEB-INF/tld/vco-taglib.tld" prefix="vco"%>
Definició tag :	vco:mantAction
Exemple d'ús:	vco:mantAction action="editPelicula.do" id="id_pelicula" label="Editar" actionIcon="img/edit.png"

Atribut	Descripció
Action	Obligatori. Acció d'*Struts* o *URL* a la que cridar per tal de realitzar la acció associada.
Id	Obligatori. Camp que fa de ID (identificador) en el *VO* utilitzat i que s'utilitzarà amb la *Struts* action per tal de realitzar la acció associada.
Label	Obligatori. Etiqueta que mostrarem sobre el llistat d'accions a les capçaleres
actionIcon	Obligatori. *URL* de la Imatge a utilitzar amb la acció de manteniment associada

Tags del Catàleg

Tag cataleg

Ús Llibreria:	<%@ taglib uri="WEB-INF/tld/vco-taglib.tld" prefix="vco"%>
Definició	vco:cataleg

tag :	
Exemple d'ús:	vco:cataleg items="${sessionScope['llistat_titols']}" styleClass="ficha_fichas"

Aquest *tag* és un *tag* pare i per tant és obligatori.

Atribut	Descripció
items	Obligatori. Llistat de camps que es mostraran a la dreta de la imatge a mostrar.
styleClass	Opcional. Aquesta serà la classe d'estils *CSS* que s'aplicarà a la fitxa del catàleg.
width	Opcional. Indicar l'amplada que volem que tingui la fitxa del catàleg.
height	Opcional. Indicar la alçada que volem que tingui la fitxa del catàleg.

Tag catalegElem

Ús Llibreria:	<%@ taglib uri="WEB-INF/tld/vco-taglib.tld" prefix="vco"%>
Definició tag :	vco:catalegElem
Exemple d'ús:	vco:catalegElem list="${sessionScope['llistat_pelicules']}" fields="${sessionScope['tagFields']}" titleField="titol" imageAction="ServletImages" detallAction="LlistarDetallPelicula.do" pagination="true" id="id_pelicula" imageStyleClass="ficha_imagen" var="item"

Aquest *tag* és un tag fill de catàleg a incloure en aquest i obligatori.

Atribut	Descripció
list	Obligatori. Representa un llistat amb els elements sobre els que es fa el manteniment. Serà un vector de Objectes de tipus *VO* o la classe que vulguem fer servir que representi els elements i camps a tractar. Nota: Per utilitzar-la requereix de una classe *VO* on els diferents

	atributs tinguin els seus mètodes *accessors* i també els atributs siguin públics.
fields	Obligatori. Llistat amb els noms dels camps a mostrar en el llistat i a accedir per obtenir els seus valors.
boxClass	Opcional. Estil *CSS* a aplicar a la caixa o fitxa de cada element.
titleField	Obligatori. Indicar el nom del camp títol que es mostrarà a la capçalera de la fitxa.
imageStyleClass	Opcional. Indicar la classe del estil *CSS* per a la imatge del element.
imageAction	Obligatori. *URL* o acció que carrega la imatge. S'utilitza un *servlet* Images per tal de carregar fitxers de imatge tipus *BLOB* de la base de dades a format byte[] java.
detallAction	Obligatori. Acció d'*Struts* o *URL* per anar a la acció de consultar el detall del element.
mouseOverClass	Opcional. Indica la classe i estil *CSS* que aplicarà a la fila quan situem el ratolí en la fitxa.
id	Obligatori. Camp que fa de ID (identificador) en el *VO* utilitzat i que s'utilitzarà amb la *Struts* action per tal de realitzar la acció associada.
var	Obligatori. Nom de la variable per a l'element actual que recupera aquest element per poder usar-lo després amb altres *tags* o elements al llistat.
pagination	Opcional. Indicar si es vol realitzar paginació sobre els llistats de manteniment no. En el cas de indicar el paràmetre a "true" aquest s'utilitza amb el tag pg:pager on es Defineixen els elements a mostrar per pàgina i elements totals. Més informació a: http://jsptags.com/tags/navigation/pager/pager-taglib-2.0.html

Tags de Novetats (*Slider*)

Tag novetats

Ús Llibreria:	<%@ taglib uri="WEB-INF/tld/vco-taglib.tld" prefix="vco"%>
Definició tag :	vco:novetats
Exemple d'ús:	vco:novetats items="${sessionScope['llistat_titols_novetats']}" styleClass="panelContainer"

Aquest *tag* és un *tag* pare i per tant és obligatori.

Atribut	Descripció
items	Obligatori. Llistat de camps que es mostraran a la dreta de la imatge a mostrar. Aquest camps han d'estar inclosos dins del *VO* que recuperarem en el tag fill.
styleClass	Opcional. Aquesta serà la classe d'estils *CSS* que s'aplicarà a l'*Slider* on es mostrarà la informació.

Tag novetatsElem

Ús Llibreria:	<%@ taglib uri="WEB-INF/tld/vco-taglib.tld" prefix="vco"%>
Definició tag :	vco:novetatsElem
Exemple d'ús:	vco:novetatsElem list="${sessionScope['llistat_novetats']}" fields="${sessionScope['tagFieldsNov']}" titleField="titol" descField="descripcio" imageAction="ServletImages" detallAction="LlistarDetallPelicula.do" id="id_pelicula" imageStyleClass="imagen" trailerAction="#" trailerIcon="img/icono-triler.gif" var="item"

Aquest tag és un tag fill i ha de novetats i és obligatori.

Atribut	Descripció
list	Obligatori. Representa un llistat amb els elements sobre els que es fa el manteniment. Serà un vector de Objectes de tipus *VO* o la classe que vulguem fer servir que representi els elements i camps a tractar. Nota: Per utilitzar-la requereix de una classe *VO* on els diferents atributs tinguin els seus mètodes accessors i també els atributs siguin públics.
fields	Obligatori. Llistat amb els noms dels camps a mostrar en el llistat i a accedir per obtenir els seus valors.
boxClass	Opcional. Estil *CSS* a aplicar a la caixa o fitxa de cada element.
titleField	Obligatori. Indicar el nom del camp titol que es mostrarà a la capçalera de la fitxa.
descField	Obligatori. Nom del camp de descripció del element a mostrar al *Slider*.
descLen	Opcional. Longitud en caràcters del camp descripció a mostrar al *Slider*.
imageStyleClass	Opcional. Indicar la classe del estil *CSS* per a la imatge del element.
imageAction	Obligatori. *URL* o acció que carrega la imatge. S'utilitza un *servlet* Images per tal de carregar fitxers de imatge tipus *BLOB* de la base de dades a format byte[] java.
detallAction	Obligatori. Acció d'*Struts* o *URL* per anar a la acció de consultar el detall del element.
trailerAction	Opcional. Acció d'*Struts* o *URL* per anar a la consulta del Tràiler del element actual.
trailerIcon	Opcional. Imatge a mostrar per a la consulta del tràiler.
id	Obligatori. Camp que fa de ID (identificador) en el *VO* utilitzat i que s'utilitzarà amb la *Struts* action per tal de realitzar la acció associada.
var	Obligatori. Nom de la variable per a l'element actual que recupera aquest element per poder usar-lo després amb altres *tags* o elements al llistat.