Arquitectura, Integració e Implementació d'un projecte Modular J2EE.

Invoicing Intranet

José Carlos Fernández Luque
ETIS

Vicenç Font Sagristà
16 de Gener del 2012

Dedicatòries i agraïments.

Dedicat als meus pares, als que sempre els hi ha fet tanta il·lusió que el seu fill tingues una titulació universitària.

Agraïments al la família i amics que sempre m'han donat el suport necessari als moments difícils durant tots aquests anys.

Resum del treball de fi de carrera.

El treball presentat es pot subdividir en dos vessants, el que seria la creació d'un sistema de treball fiable i rentable per un projecte multitudinari, i la implementació d'una aplicació.

La primera i amb mes pes consisteix en **l'Arquitectura** e **Integració** d'un projecte **J2EE Modular**, amb una estructura **MCV** creada i mantingut per **Maven** a traves del seu arquetipus **flexmojos-archetypes-modular-webapp**. Aquesta estructura ha de permetre reprendre el desenvolupament inicial, nous desenvolupaments i manteniments d'una aplicació, amb un cost temporal mínim per la part de l'equip de desenvolupament, i està pensat per que poden conviure, tant a la Implementació com al Manteniment, molts grups de treball, segurament distanciats geogràficament, i amb una quantitat de persones entre 50 i 300. Amb només instal·lacions bàsiques de l'entorn de Desenvolupament, importacions mínimes del projecte mitjançant el **SVN Subclipse** i l'execució dels goals basics de **Maven**, un desenvolupador podrà iniciar de forma productiva les seves tasques en local sense precisar mes coneixements que els imprescindibles de programació. La complexitat màxima resideix a la integració a traves de **Maven** de **Spring**, **Hibernate** a la part **Java**, a la capa de la vista **Flex** amb **Parsley**, **SpaceLib** i **Flexlib**, y comú a les dues capes tindríem **BlazeDS**. Aquesta integració, ha de funcionar correctament tant a les compilacions que genera **Maven** com a l'entorn de Desenvolupament a traves d'**Eclipse**. Per altre banda, s'han utilitzat tres **Entorns** (desenvolupament, integració i producció) per recrear una sistemàtica real de treball.

La segona es l'**Anàlisis**, **Disseny** e **Implementació** d'una aplicació que permetrà la iniciació en mode de visita d'un usuari genèric amb accés al **Manteniment** d'una entitat Facturació, relacionada amb el usuari autenticat, mitjançant les tecnologies i frameworks integrats per **Maven**. La base de dades utilitzada es **MySql** y com servidor de aplicacions **Tomcat**. La seva temàtica està orientat a la gestió de **Flotes de Taxi**, un tipus de producte, que encara no existeix al mercat, ja que aquestes gestions son normalment atribuïdes a administració, i sense contacte interactiu amb l'usuari final a traves d'una **Intranet**.

www.lukysite.net/war-1

www.lukysite.net/tfc/videos/video1.mp4

Índex General.

1 Introducció.

1.1 Justificació del TFC i context.

La principal motivació d'aquest projecte resideix als continus problemes detectats al desenvolupament de qualsevol projecte J2EE al mercat, desenvolupat per diferents grups de treball, ubicacions geogràfiques llunyanes i treball col·lectiu asíncron, on l'arquitectura, integració i desenvolupament corren per camins diferents, provocant dependencies dificultosament superables entre ells.

Simplificant la qüestió, l'estructura jeràrquica general en un projecte de gran envergadura, parteix de la connexió entre el client final (Funcionals i Clients finals) i el grup d'integració, que es qui ha de mostrar els avanços. Els integradors i Funcionals tindrien per sota un parell de grups, arquitectura i desenvolupament, on son traspassats els requeriments dels clients finals, per cercar solucions generals a la banda d'arquitectura e I+D, o solucions especifiques a la banda del desenvolupament.

Maven, un projecte Modular i un SVN son imprescindibles actualment per assolir implementacions rapides i estables en un conjunt de treball com el descrit, que es principalment el aconseguit a aquest TFC.

1.2 Objectius del TFC.

L'objectiu principal del TFC consisteix en la creació d'una complexa estructura modular J2EE basada i mantinguda per Maven, amb la utilització com frameworks Spring, Hibernate i Flex principalment. Aquesta estructura permet reprendre el desenvolupament inicial, nous desenvolupaments i manteniments d'una aplicació, amb un cost temporal mínim per la part de l'equip de desenvolupament.

Amb només instal·lacions bàsiques de l'entorn de Desenvolupament, importacions mínimes del projecte mitjançant Subclipse sobre l'entorn i l'execució dels goals basics de Maven, per la importació de gairebé totes les llibreries utilitzades, un desenvolupador podrà iniciar de forma productiva les seves tasques en local sense precisar mes coneixements que els imprescindibles de programació. Aquest entorn mostrarà els resultats diaris mitjançant Eclipse amb el seu servidor d'aplicacions Tomcat i una Base de dades MySql local, que amb la generació dels wars corresponents podran ser exportats a la resta d'entorns.

A mes, a la banda d'integració, es minimitzaran les dependencies amb la resta d'equips amb la utilització de Maven, ja que qualsevol canvi introduït a l'arquitectura actual passarà per ell, i podrà ser provat als diferents entorns. Tanmateix, es gaudirà a aquesta banda de dos entorns mes, que podríem anomenar Integració i Producció.

L'entorn d'integració serà el entorn Local de Tomcat sense la utilització d'Eclipse i compartirà la BD de desenvolupament per estalviar una copia exacta d'aquesta. A l'entorn de Producció gaudirem de les versions mes estables creades, comprovades a l'entorn d'Integració, i pujades a un servidor extern amb la seva pròpia base de Dades. També podrem gaudir a aquest servidor d'un SVN com controlador de versions, imprescindible per un treball modular de tal embargadora.

L'aplicació en si permetrà la iniciació en mode de visita d'un usuari genèric amb les dades necessàries per ser provat als tres entorns. Llavors tindrem un únic perfil Usuari, que amb privilegis a tota l'aplicació.

- **Usuari**: Tota aquella persona que tingui accés al sistema. Aquest usuari podrà navegar per la totalitat de l'aplicatiu, fer el manteniment i llistar informes sobre la seva Facturació.

Per portar a terme aquest desenvolupament, l'aplicació serà dividida en tres subsistemes, que descriurem a la resta de la documentació. Aquests seran:

- **Subsistema de connexió.**
- **Subsistema de manteniment.**
- **Subsistema de informes.**

Els subsistemes son tots remots, i l'usuari despondrà d'una interfície gràfica de connexions client/servidor.

1.3 Enfocament del TFC.

L'enfocament del TFC, ha estat la iniciació del projecte modular a partir de la estructura que ens genera Maven amb el seu arquetipus flexmojos-archetypes-modular-webapp, que gairebé només ens dona una estructura inicial de carpetes i fitxers i que ha de ser desenvolupada per poder ser importada al nostre entorn de treball, l'Eclipse.

A partir d'aquesta importació, tot framework i tecnologia introduïda ha de funcionar correctament tant a la part de Maven com a la banda de l'Eclipse per tenir un paral·lelisme complet sense problemes afegits. Aquí resideix la principal dificultat ja que els resultats obtinguts a l'entorn de Desenvolupament han de ser els mateixos que genera Maven. Les compilacions d'Eclipse, el típic **Clean** que compila els nostres projectes, i dels de Maven, amb els seus goals **Clean e Install**, han de obtenir els mateixos resultats i les dependencies utilitzades han de conviure als dos d'una forma estable.

El principal problema d'integració el porta la part de Flex, ja que localment l'Eclipse utilitza un SDK preinstal·lat i Maven una dependència diferent. De la bona elecció d'aquestes dependencies resideix bona part del correcte funcionament d'aquest paral·lelisme funcional.

Amb la resta de dependencies passa el mateix, encara que les importacions de les llibreries de Maven passen directament als projectes, son Java i son assolides mes còmodament per l'Eclipse. He fet que cada mòdul sigui independent, i que es comuniquin mitjançant el nostre repositori local, es descarreguen les dependencies necessàries intramedul·lars (cada mòdul es descarrega les seves llibreries, swc s'importa a swf, swf s'importa a war, persistència i serveis es podrien implementar i s'importaria a war també, etc).

Tot el projecte està enfocat per assolir una estructuració MCV i un enfocament Orientat a Objectes. Per suposat tota la part Java està completament integrada a aquestes dues tecnologies, però també podrem observar que la capa de la Vista les suporta perfectament, cosa que ens dona reutilització, herència, etc a aquesta capa.

Els principals problemes detectats inicialment van ser:

- ✓ Incompatibilitats entre Window 7 i la resta de tecnologies segons versions, Eclipse, plugin de Maven, plugin de FlexBuilder,...
- ✓ Incompatibilitats entre Eclipse i versions de SDK per Flex.
- ✓ Incompatibilitats entre SDKs i compiladors de FlexMojos de Maven.
- ✓ Repositoris amb dependencies molt diferents a les esperades.
- ✓ Instal·lacions del SVN al meu Host personal.
- ✓ Bugs a gairebé tots els frameworks utilitzats.
- ✓ Manca d'exemples que realment funcionin a Internet.

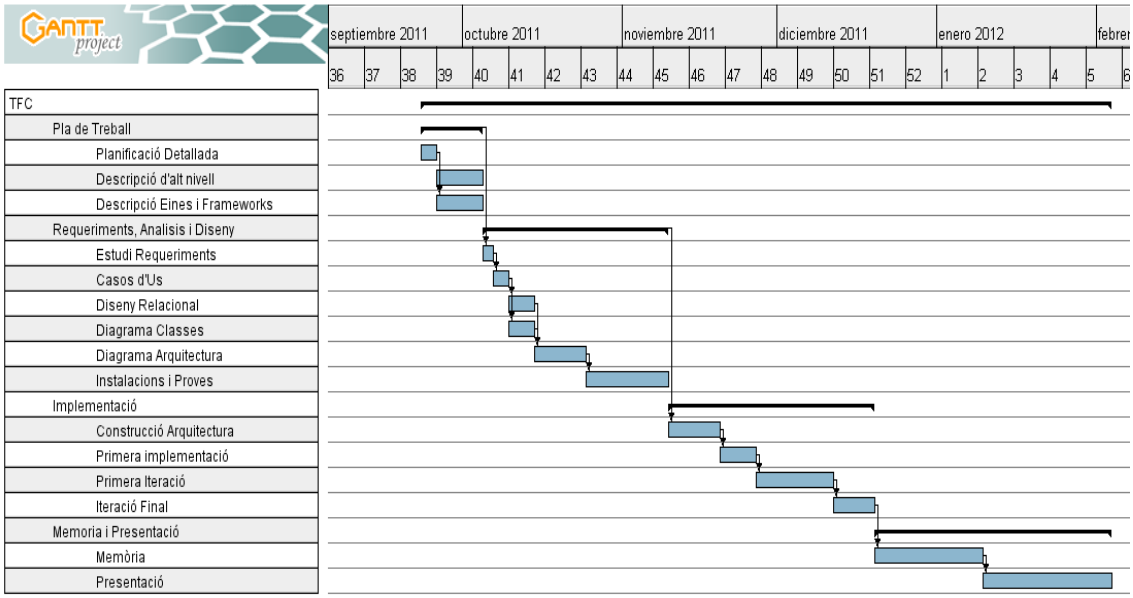
Totes aquestes dificultats han estat superades, i el resultat del projecte es completament el descrit al Pla de Treball inicial.

1.4 Planificació del projecte.

Prenent com a dates claus les entregues de les Pacs principals, s'ha creat tota la planificació del projecte a traves de GanttProject, Diagrames de Gantt i de Recursos. Algunes de les tasques han estat solapades al 50% del treball temporal, i totes les dependencies han estat senyalades també. Podem visualitzar el Project sencer a la següent url:

http://www.lukysite.net/tfc/project/project.html

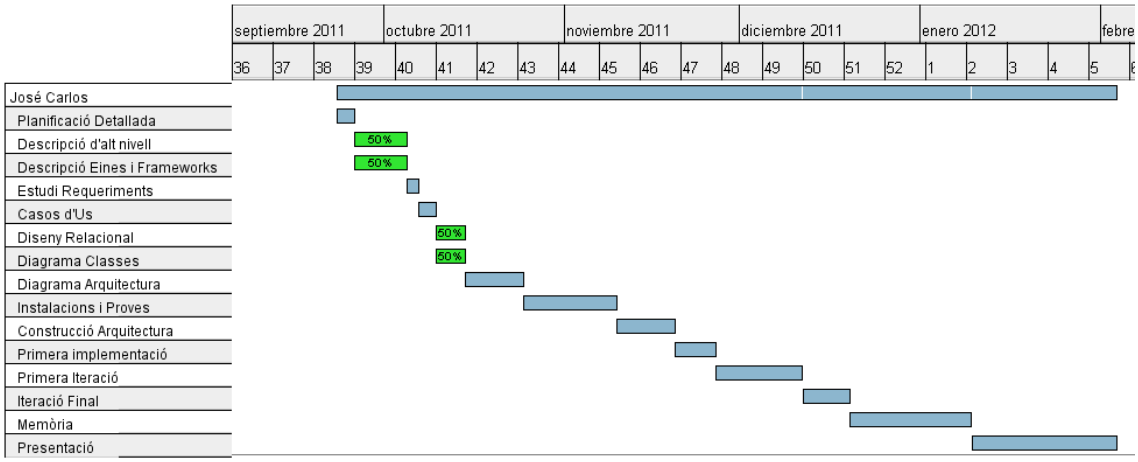
Al diagrama de Ganntt podem observar que gairebé totes les tasques tenen antecessors per ser contemplades, ja que les dependencies creades ens obliguen a seguir un curs temporal lineal.



Podem observar les dates claus a aquest resum:

Nombre	Fecha de inicio	Fecha de fin	Coordinador	Recursos
TFC	23/09/11	4/02/12		
Pla de Treball	23/09/11	5/10/11		
Planificació Detallada	23/09/11	26/09/11	José Carlos	José Carlos
Descripció d'alt nivell	26/09/11	5/10/11	José Carlos	José Carlos
Descripció Eines i Frameworks	26/09/11	5/10/11	José Carlos	José Carlos
Requeriments, Analisis i Diseny	5/10/11	10/11/11		
Estudi Requeriments	5/10/11	7/10/11	José Carlos	José Carlos
Casos d'Us	7/10/11	10/10/11	José Carlos	José Carlos
Diseny Relacional	10/10/11	15/10/11	José Carlos	José Carlos
Diagrama Classes	10/10/11	15/10/11	José Carlos	José Carlos
Diagrama Arquitectura	15/10/11	25/10/11	José Carlos	José Carlos
Instalacions i Proves	25/10/11	10/11/11	José Carlos	José Carlos
Implementació	10/11/11	20/12/11		
Construcció Arquitectura	10/11/11	20/11/11	José Carlos	José Carlos
Primera implementació	20/11/11	27/11/11	José Carlos	José Carlos
Primera Iteració	27/11/11	12/12/11	José Carlos	José Carlos
Iteració Final	12/12/11	20/12/11	José Carlos	José Carlos
Memoria i Presentació	20/12/11	4/02/12		
Memòria	20/12/11	10/01/12	José Carlos	José Carlos
Presentació	10/01/12	4/02/12	José Carlos	José Carlos

He inclòs també aquest diagrama de recursos, encara que només tindrem un recurs a aquest desenvolupament. La correcte concatenació y superposició de les tasques reflecteix el 100% de la utilització del recurs.



1.5 Productes obtinguts.

El producte obtingut, el podem desglossar en les funcionalitats que ens dona el projecte. Aquestes funcionalitats sobre el meu projecte modular son:

- Diferents grups de desenvolupadors de qualsevol mida poden descarregar-se el projecte d'un SVN i poder fer una importació rapida amb unes mínimes configuracions, començar a produir en 30 minuts.
- Amb els goals mínims (clean, install), tot un mon de llibreries es descarreguen automàticament al repositori local o de servidor (500 MB aproximadament), per que tot el grup pugui treballar només amb el que arquitectura vol que es treballi. A la major part dels projectes empresarials, cada desenvolupador acaba introduint les llibreries que pensen convenients per separat, amb el que al final el treball global no funciona correctament. Un desenvolupador nou pot trigar setmanes per muntar un entorn estable.
- Una arquitectura modular per dividir els grups. He creat només tres mòduls, però lo normal seria tenir també de Serveis i Persistència al menys (jo ho he separat en packages al mòdul war).
- Cada mòdul es independent, i mitjançant el repositori , es descarreguen les dependencies necessàries intermodulars (cada modul es descarrega les seves llibreries, swc s'importa a swf, swf s'importa a war, persistència i serveis s'importaria a war també, etc). A Maven està gestionat tot això.
- Creació automàtica de les taules a la BD, només amb l'existència d'una base de dades anomenada "invoicing". Tota la estructura de persistència es governada des de Java, el que suposa un estalvi de personal i de temps considerablement gran.
- Integrar en tot aquest mon sobre Flex (Parsley, SpaceLib, Flexlib), BlazeDS (per la comunicació entre Vista i Controlador), Spring e Hibernate (pel context, capa de Model, integració Controlador-Model), entre d'altres. Aquesta integració ha de

servir per que Eclipse funcioni correctament per desenvolupar (es pot depurar perfectament a tots els mòduls), i per altre costat, Maven ha de generar tot l'entramat sense problemes.

- La capa vista amb Flex te la seva pròpia arquitectura MCV com s'explicarà. A mes es un dels únics llenguatges de la capa Vista gairebé Orientat a Objectes com es pot veure (herència per exemple a DateRangeValidator, reutilització a GenericArrayEvent i ConditionalEvent, etc).
- Tota una intranet que permetrà la iniciació en mode de visita d'un usuari genèric amb accés al manteniment d'una entitat Facturació, relacionada amb el usuari autenticat, manteniment completament compacte a una sola pantalla. L'aplicatiu es agradable a la vista, navegació sense cap problema, cap incongruència, correcte tabulació per no tenir que utilitzar gairebé el ratolí, tol tips a tot arreu, internacionalització, etc...

Com podem observar, les principals funcionalitats estan orientades a Arquitectura e Integració, amb el que obtindrem com a resultat un còmode, ràpid i eficaç desenvolupament de qualsevol aplicació, de qualsevol mida, i amb qualsevol nombre i grups d'individus.

1.6 Descripció de la resta de capítols de la memòria.

A la resta de capítols desglossarem tot el que ha representat el treball realitzat des de l'inicia de la primera configuració de l'entorn i tota la construcció del Projecte Modular fins a la implementació de l'aplicatiu Invocing Intranet.

2 Entorn de desenvolupament

L'entorn de desenvolupament consistirà bàsicament en tot un conjunt d'eines que envolten el nostre IDE a utilitzar, Eclipse. Com que l'arquitectura requereix de components diferents als basics en un entorn J2EE, haurem d'importar alguns plug-ins especials pel desenvolupament. A mes precisarem d'un entorn d'execució per Flex, una Base de Dades local Mysql i la instal·lació de Maven en la seva versió mes estable. El conjunt complet de instal·lacions seria el següent:

2.1 Maven.

Maven serà l'eina principal del projecte, amb la que gestionarem la generació del projecte. Han estat testejades les versions 2.2.1 i 3.0.3 a la maquina amb resultats semblants. La instal·lació al sistema operatiu, precisa de declaracions de variables d'entorn. A aquests vincles podrem trobar guies d'us e instal·lació:

- http://maven.apache.org/
- http://www.chuidiang.com/java/herramientas/maven.php

2.2 Tomcat.

Tomcat es el servidor d'aplicacions escollit ja que conté tots els serveis necessaris pels requeriments del projecte J2EE. La seva instal·lació es immediata, i en servirà tant per utilitzar-ho a l'entorn de Desenvolupament mitjançant la integració amb l'Eclipse, com al de Integració per desplegar els wars generats als entorns de Integració com Producció. També podem gaudir de moltes versions, i l'escollida per la seva estabilitat comprovada es la 6.0.32.

- http://tomcat.apache.org/
- http://tomcat.apache.org/download-60.cgi
- http://profesor.antonio.com.mx/?p=129

2.3 Flex.

L'entorn Flex precisa de la instal·lació de Adobe Flash Builder 4, que per la seva vinculació amb el IDE de desenvolupament, serà inclosa al següent punt. A banda de les propietats que dona al nostre Eclipse, podem destacar la inclusió de SDKs necessàries per l'execució de les pel·lícules generades.

2.4 Eclipse.

Sense dubtes, Eclipse sembla la millor eina de desenvolupament per les necessitats del projecte. La seva versatilitat per la integració de diferents tipus de projectes, ens assegura un correcte inici de l'arquitectura escollida. A la banda del nostre IDE precisaríem les següents instal·lacions:

1. **Descarrega e instal·lació d'Eclipse.** Gairebé totes les versions d'Eclipse ens servirien, a aquest desenvolupament s'ha utilitzat la versió Ganymede, Eclipse IDE for Java Developers, que es un dels mes complets. La versió Helios també sembla molt estable.
   - http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/indigosr1
2. **Instal·lació del Maven 2 a Eclipse,** una versió prou estable i comprovada per assegurar la manca de incidències. Al sistema operatiu, podem instal·lar qualsevol versió que sigui superior. Podem trobar moltes guies de instal·lació per Internet:
   - http://chuwiki.chuidiang.org/index.php?title=Integraci%C3%B3n_de_Maven_y_Eclipse
   - http://www.latascadexela.es/2008/09/configurar-eclipse-para-trabajar-con.html
3. **Instal·lació de l'entorn Flex.** Inicialment precisarem la instal·lació de Adobe Flash Builder 4. Actualment tenim mes versions. Aquesta instal·lació està inclosa a Eclipse, ja que s'ha de vincular al IDE escollit pel desenvolupament.

- http://www.adobe.com/support/flashplayer/downloads.html

Al projecte estem utilitzant la versió , que inclou:

- Flash Builder 4 Standalone install folder
- Flash Builder 4 Eclipse Plug-in install folder
- Extras folder that includes:
    - LiveCycle Data Services 3.0 installer
    - Adobe Application Modeling Plug-in for Eclipse 3.4
    - Adobe Application Modeling Plug-in for Eclipse 3.5
    - LiveCycle Service Discovery Plug-in for Flash Builder 4.0
- Flex 4 Test Automation Plug-in

A mes com opció, es pot incloure Flex Formatter, un plug-in que ens estalvia molt de treball a l'hora de formatar la totalitat del codi Flex, ja que Eclipse no està preparat per aquest tipus de codi. El plug-in es pot importar directament des de Eclipse:

- http://flexformatter.googlecode.com/svn/trunk/FlexFormatter/FlexPrettyPrintCommandUpdateSite

4. **Instal·lació de Subclipse**. Aquesta fase es primordial pel correcte desenvolupament conjunt, ja que no hi ha cap altre forma d'organització mes còmode i convenient. Des de Eclipse podem instal·lar aquest plug-in, que ens permetrà integrar la Subversion (SVN) amb Eclipse, in ens permet gestionar tot el que precisem d'un controlador de versions:

    ✓ Browse remote repository
    ✓ Share project to the repository and checkout projects from the repository
    ✓ Synchronize project to see incoming and outgoing changes
    ✓ Commit, update and revert changes
    ✓ See resource change history
    ✓ Merge changes

    Els vincles d'informació e instal·lació son:

    - http://www.eclipse.org/subversive/
    - http://tratandodeentenderlo.blogspot.com/2009/09/manejo-de-subversion-desde-eclipse.html

5. **Millora de memòria.** A banda de la configuració inicial que ens proporciona Eclipse al seu arxiu eclipse.ini, hauríem d'incloure o pujar el nivell de memòria utilitzada, ja que el desenvolupament Flex, Tomcat i tot el conjunt d'eines i plug-ins utilitzats acaben donant errors de memòria:

    -Xms768m

    -Xmx1024m

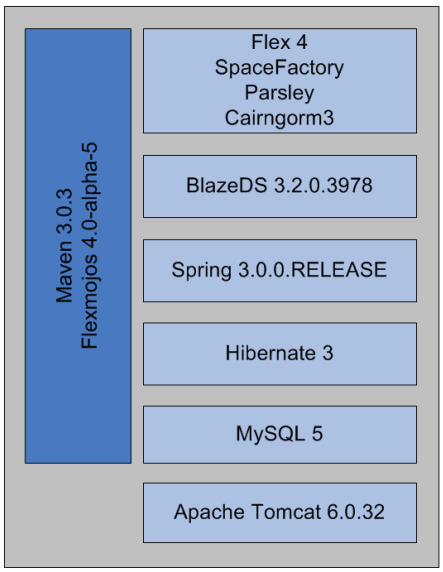
6. **Normativa UTF-8.** Per no tenir problemes amb els caràcters especials, hem de configurar els workspaces a UTF-8, fen aquesta selecció als següents espais:

- Windows > Preferences > General
  - Content Types > Text
    - Java Properties files
    - Java Source File
  - Workspace > Text file encoding > Other

Amb aquestes instal·lacions, ja tindríem l'entorn de treball preparat per una immediata iniciació de la producció. Encara que el mes recomanat es la generació d'una maqueta inicial estable.

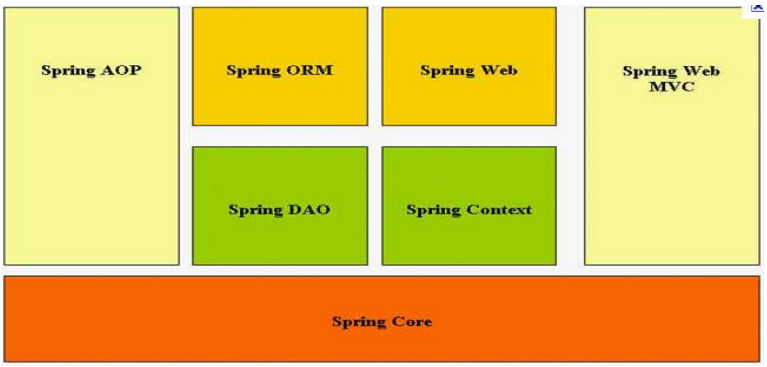
3 Arquitectura.

L'arquitectura utilitzada partirà com ja hem explicat d'una estructura MVC amb la part client Flex a la capa de la Vista, i la part servidora de Java, creada i mantinguda per Maven. Bàsicament s'utilitzen els components, que ens proporcionen a cadascuna de les capes les utilitats que precisem:



Encara que en aquesta primera implementació utilitzem un molt petit percentatge de les funcionalitats que ens proporcionen els frameworks utilitzats, una de les finalitats del projecte es la possibilitat d'ampliació e introducció de nous requeriments que les precisarien. Amb Spring, Hibernate i les llibreries de Flex podem cobrir gairebé tots els requeriments imaginables.

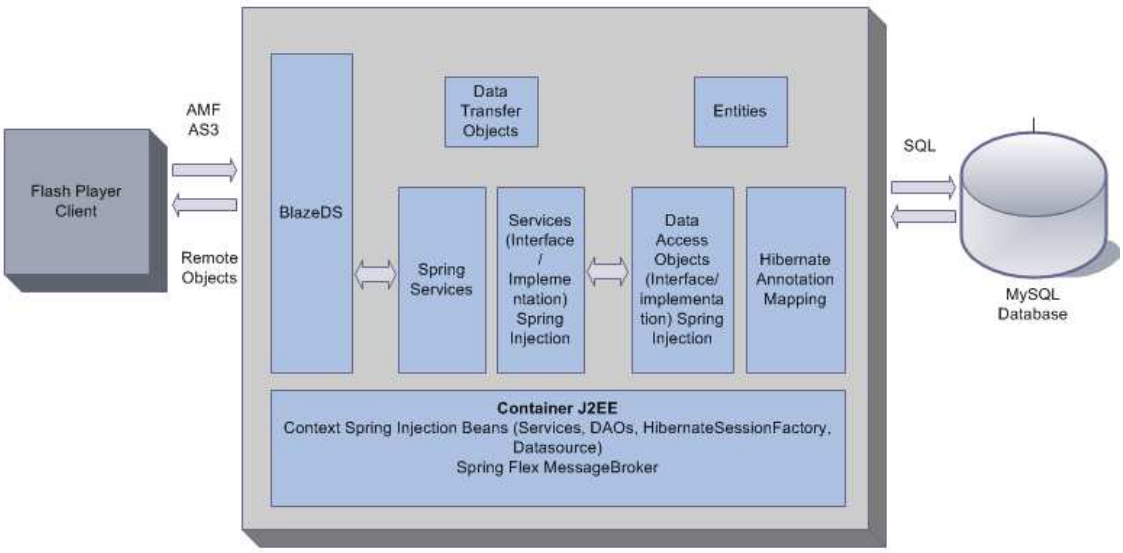
Al projecte podem destacar la presencia de Spring a gairebé totes les capes, ja que es un framework molt complet i amb un nivell d'integració molt alt, com poden veure a la seguen imatge.



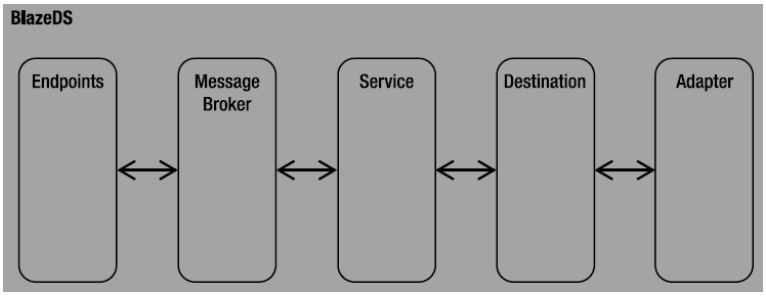
3.1 Capa Servidor.

Als següents esquemes, descriurem la banda del servidor i la banda client per separat. Podem observar completament separades les capes de Model i Controlador, on els Serveis i els DTOs per un costat i, les Entitats i els DAOs formen principalment cada subdivisió. Aquesta encapsulació, ens permetria resoldre el problema de canviar d'arquitectura sense gaires canvis a la capa que no volem modificar.

Server Layer



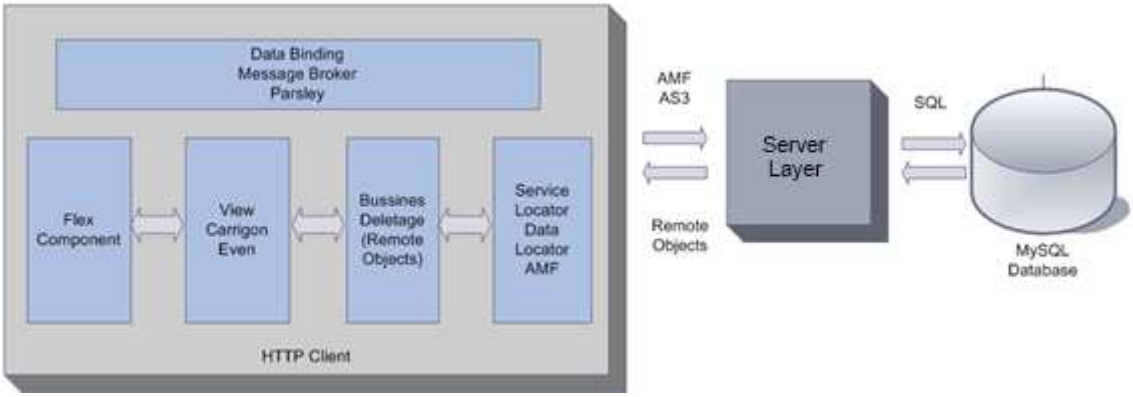
BlazeDS es una part molt important de l'arquitectura, que mitjançant la integració de Spring ens permet una perfecte comunicació entre la capa Client i Servidor. La seva estructuració i metodologia ve reflectida al seguen diagrama:



3.2 Capa Client.

La capa client, com ja vam comentar, a la seva vegada te una estructura MVC que podem apreciar al diagrama. Els mateixos objectes DTOs del servidor tenen el seu paral·lelisme a aquesta capa, que serien el seu Model. La capa Controladora serien els Bussines Delegate, en el nostre cas els Remote Objects, on podem gestionar modificacions sobre el Model segons les necessitats. I la part de la Vista pura serien els components de Flex.

Client Layer



4 Integració.

Maven ha estat l'eina escollida per la creació inicial, desenvolupament a nivell de integració, compilació de codi i empaquetat. A mes, aprofitarem al màxim la seva adaptació a la Red, amb la qual gestionarem integrament totes les llibreries, plug-ins i algunes de les utilitats que se'ns proporcionen un mon de repositoris.

Per la vista, s'ha triat Flex entre d'altres raons, per la gran capacitat per crear entorns visuals complexes molt agradables, la molt avançada Orientació a Objectes que suporta, i una molt completa integració amb moltes de les ultimes tecnologies aplicades a la banda del Servidor.

Partint d'aquesta base, mitjançant el arquetipus de flexmojos, flexmojos-archetypes-modular-webapp, crearem l'estructura inicial del projecte. El resultat de la creació

automàtica del projecte, ha de ser modificada, ja que dona molts errors. Aquesta tindrà la seguen forma:

- **Projecte Pare**: Aquest projecte inclourà al seu POM (Project Obect Model) els repositoris a utilitzar de forma comú a la resta de projectes. També tindrem els propietats sobre versions (SNAPSHOT), paths, etc. L'herència que ens proporciona Maven, ens dona simplicitat a l'hora de configurar solucions comuns. A mes podrem fer una compilació completa, o qualsevol dels "goals" que precisem, de tots els mòduls fills en l'ordre establert.
    - **Mòdul SWC:** Llibreria Flex on pot treballar arquitectura i desenvolupament conjuntament. Al nostre cas serà utilitzat per emmagatzemar els objectes de suport dels components Flex, DTOs. Es podria aprofitar per la creació o herència de components també, en general per albergar solucions globals.
    - **Mòdul SWF:** Projecte Flex on romandrà la totalitat de la capa Vista, amb dependència sobre el projecte SWC.
    - **Mòdul WAR:** Projecte Java on es generarà el war necessari per ser deplojat al nostre cas a Tomcat. Aquest dependrà del projecte SWF, particularment de la seva pel·lícula swf generada.

Cadascun dels mòduls gaudirà de les seves pròpies dependencies, plug-ins, repositoris, segons les necessitats individuals, que seran declarats al seu pom.

Com ja s'ha explicat, tant la banda servidora com la vista mitjançant Java I Flex, estaran completament encaminades a la Orientació a Objectes, ja que les dues tecnologies la suporten prou be. Aquesta qualitat ens donarà reutilització, fins i tot a la capa de la Vista, que com ja sabem normalment es molt especifica. Ens proporcionarà també molta independència entre les dues capes, pel que en qualsevol moment podrem canviar de tecnologies a la capa de la Vista sense gaires canvis a la capa de Controlador i cap a la capa del Model.

Implícitament, a la capa de la Vista, Flex, podrem observar que a banda de la Orientació a Objectes, gaudirem també d'una arquitectura MVC, amb Models que representen les dades, Vista per la representació visual de les dades i Controlador responsable de canviar les dades del model.

També cal recalcar la possible divisió modular segons la magnitud del projecte (mòduls de serveis, persistència,etc).

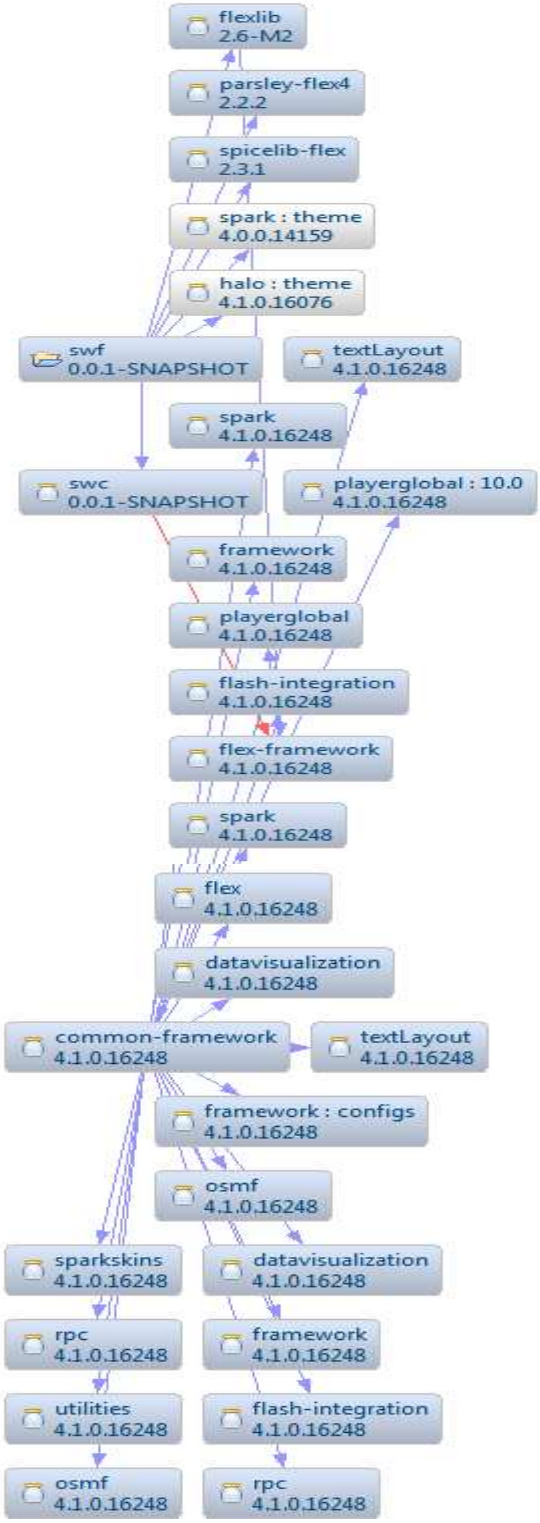
4.1 Jerarquia de dependencies.

Gairebé tots els mòduls, dependran del repositori que Maven utilitza per defecte. A mes, de forma comú, utilitzarem el flex-mojos-repository, el flex-mojos-plugin-repository i cairgorm3-repository.

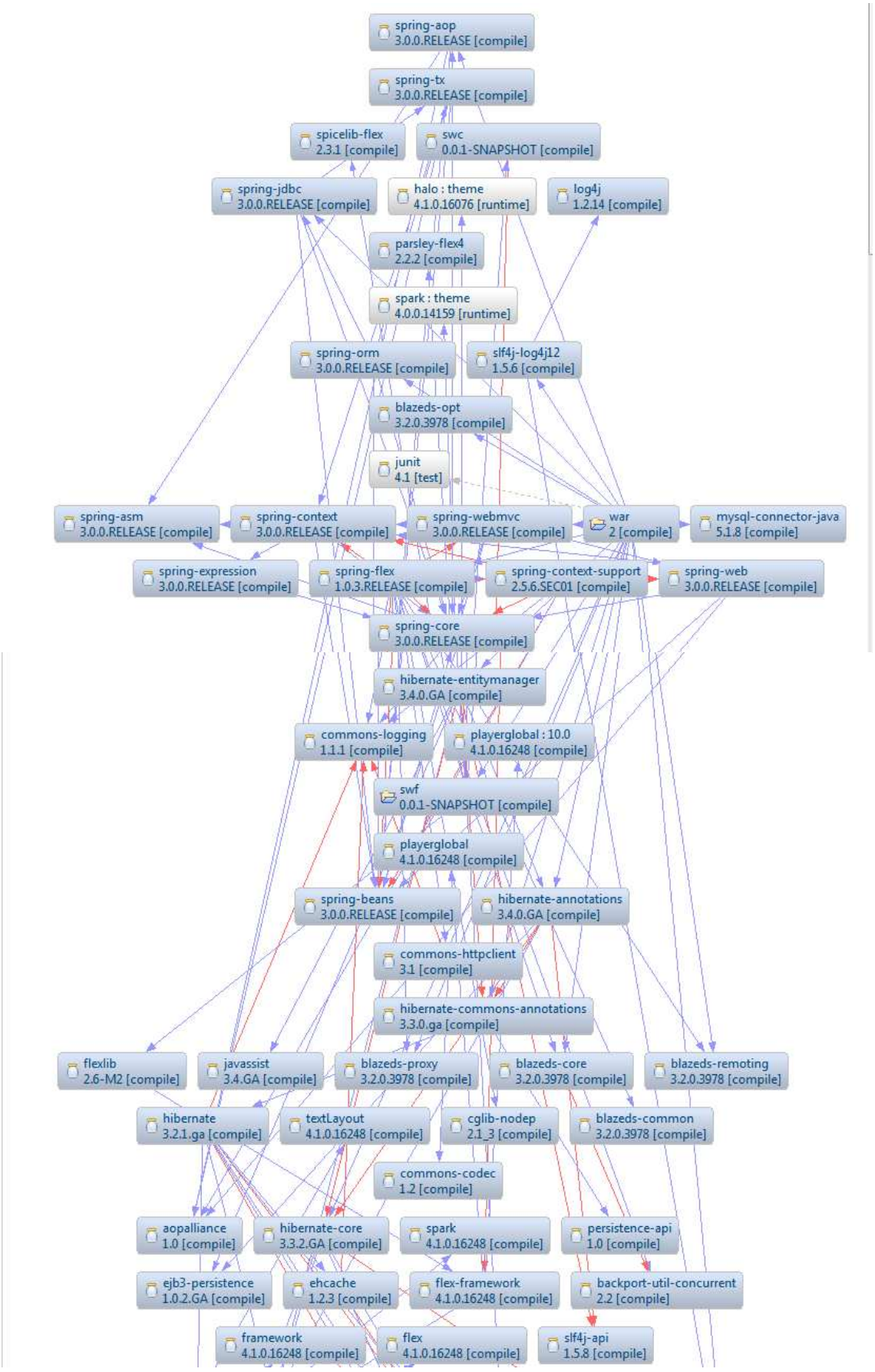
Amb aquest tres repositoris, gairebé podrem obtindre la totalitat de les dependencies que precisem. Als següents punts mostrarem les dependencies que es generen de cada un dels projectes, mitjançant un graf, al que podrem visualitzar la totalitat d'eines i frameworks utilitzats:

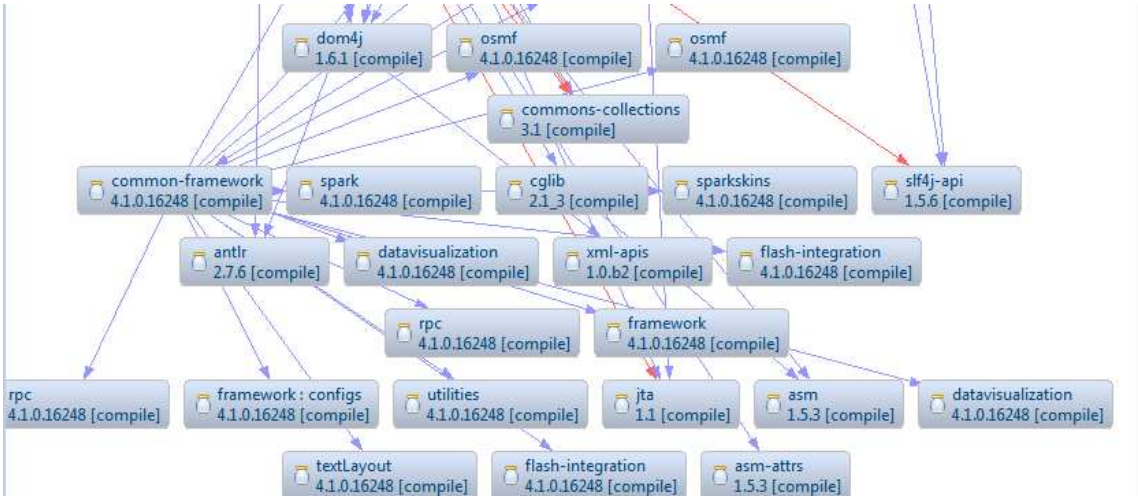
- **Projecte Pare:** Conté el conjunt de objectes comuns a tots els mòduls, repositoris principalment. No te dependencies especifiques.
- **Mòdul SWC:** Conté les dependencies mínimes per la compilació de la llibreria swc, per la seva simplicitat. Com llibreria es podria utilitzar per albergar tot un mon de objectes a utilitzar al projecte, i podem veure a la seva jerarquia que podria albergar sobrescriptura de components,

- **Mòdul SWF:** Aquest mòdul, una mica mes complexa, conté dependencies a llibreries swc i repositoris especials:



- **Mòdul WAR:** Aquest mòdul conté totes les dependencies de la part Java, i tots els frameworks utilitzats:


spring-aop
3.0.0.RELEASE [compile]
spring-tx
3.0.0.RELEASE [compile]
spicelib-flex
2.3.1 [compile]
swc
0.0.1-SNAPSHOT [compile]
spring-jdbc
3.0.0.RELEASE [compile]
halo : theme
4.1.0.16076 [runtime]
log4j
1.2.14 [compile]
parsley-flex4
2.2.2 [compile]
spark : theme
4.0.0.14159 [runtime]
spring-orm
3.0.0.RELEASE [compile]
slf4j-log4j12
1.5.6 [compile]
blazeds-opt
3.2.0.3978 [compile]
junit
4.1 [test]
spring-asm
3.0.0.RELEASE [compile]
spring-context
3.0.0.RELEASE [compile]
spring-webmvc
3.0.0.RELEASE [compile]
war
2 [compile]
mysql-connector-java
5.1.8 [compile]
spring-expression
3.0.0.RELEASE [compile]
spring-flex
1.0.3.RELEASE [compile]
spring-context-support
2.5.6.SEC01 [compile]
spring-web
3.0.0.RELEASE [compile]
spring-core
3.0.0.RELEASE [compile]
hibernate-entitymanager
3.4.0.GA [compile]
commons-logging
1.1.1 [compile]
playerglobal : 10.0
4.1.0.16248 [compile]
swf
0.0.1-SNAPSHOT [compile]
playerglobal
4.1.0.16248 [compile]
spring-beans
3.0.0.RELEASE [compile]
hibernate-annotations
3.4.0.GA [compile]
commons-httpclient
3.1 [compile]
hibernate-commons-annotations
3.3.0.ga [compile]
flexlib
2.6-M2 [compile]
javassist
3.4.GA [compile]
blazeds-proxy
3.2.0.3978 [compile]
blazeds-core
3.2.0.3978 [compile]
blazeds-remoting
3.2.0.3978 [compile]
hibernate
3.2.1.ga [compile]
textLayout
4.1.0.16248 [compile]
cglib-nodep
2.1_3 [compile]
blazeds-common
3.2.0.3978 [compile]
commons-codec
1.2 [compile]
aopalliance
1.0 [compile]
hibernate-core
3.3.2.GA [compile]
spark
4.1.0.16248 [compile]
persistence-api
1.0 [compile]
ejb3-persistence
1.0.2.GA [compile]
ehcache
1.2.3 [compile]
flex-framework
4.1.0.16248 [compile]
backport-util-concurrent
2.2 [compile]
framework
4.1.0.16248 [compile]
flex
4.1.0.16248 [compile]
slf4j-api
1.5.8 [compile]



Com es pot observar, al major part de la complexitat del projecte recau sobre la integració de tot aquest mon de frameworks i eines, que Maven ens permet entrellaçar.

A partir de la simple estructura XML del pom de cada projecte, podem configurar amb exactitud el que es precisi al projecte. Forma part llavors del grup d'arquitectura e integració la manipulació d'aquests, per noves configuracions, decisions sobre que frameworks utilitzar, versions, detalls del empaquetat del projecte segons les necessitats, compilacions, etc.

Podem observar també, que realment s'aprofitarà una part molt petita de la totalitat de les funcionalitats que es poden extreure d'una infraestructura tan complexa com aquesta.

4.2 Integració Maven-Eclipse.

El principal problema d'una Integració d'aquest tipus, es obtindre un resultat equivalent entre les compilacions i generacions vaires que fa l'Eclipse, utilitzades per valorar el correcte funcionament del equip de Desenvolupament, i les que fa Maven, utilitzades per crear un versionat estable i compacte del projecte. Els primers passos ja han estat solvatats amb la jerarquia de dependencies del punt anterior, ja que moltes de les execucions dels goals principals de Maven son internes a cadascuna de les dependencies descrites.

L'adaptació dels plug-ins utilitzats, les versions dels frameworks, etc es un punt crucial a la integració. Un mon d'errors son esperats en les primeres implementacions de la integració, on el que funciona correctament a la banda del Eclipse, no funciona als llançaments dels goals de Maven. El desitjat resultat final, porta molt de treball d'investigació i testeig, i els trams mes conflictius son explicats als següents punts. Els goals llençats son Clean e Install.

```
[INFO] Reactor Summary:
[INFO]
[INFO] invoicing-web ..................................... SUCCESS [0.376s]
[INFO] swc Library ....................................... SUCCESS [19.284s]
[INFO] swf-project ....................................... SUCCESS [10.455s]
[INFO] war ............................................... SUCCESS [16.528s]
```

4.2.1 Pom swc.

A quest pom, per la seva simplicitat inicial, no precisa de grans accions. A banda de les compilacions que fan les dependencies, només precisa una copia de dependencies pel seu correcte funcionament al desenvolupament:

- Copia de dependencies:

```
<plugin>
    <artifactId>maven-dependency-plugin</artifactId>
    <executions>
        <execution>
            <id>copy-dependencies</id>
            <phase>generate-sources</phase>
            <goals>
                <goal>copy-dependencies</goal>
            </goals>
            <configuration>
                <includeTypes>swc</includeTypes>
                <outputDirectory>${basedir}/libs</outputDirectory>
                <overWriteReleases>true</overWriteReleases>
                <excludeGroupIds>com.adobe.flex.framework</excludeGroupIds>
            </configuration>
        </execution>
    </executions>
</plugin>
```

4.2.2 Pom swf.

La complexitat d'aquest pom resideix en el paral·lelisme que ha d'existir entre les versions que obtenim als repositoris externs i les que utilitza l'Eclipse amb el seu Flex Builder 4. Exemples de comptabilitat son per exemple, que algunes llibreries del theme utilitzat (Halo) no accepten locales es_ES. Les accions inserides que aquest pom realitzarà son:

- Copia de dependencies:

```
<plugin>
    <artifactId>maven-dependency-plugin</artifactId>
    <executions>
        <execution>
            <id>copy-dependencies</id>
            <phase>generate-sources</phase>
            <goals>
                <goal>copy-dependencies</goal>
            </goals>
            <configuration>
                <includeTypes>swc</includeTypes>
                <outputDirectory>${basedir}/libs</outputDirectory>
                <overWriteReleases>true</overWriteReleases>
                <excludeGroupIds>com.adobe.flex.framework</excludeGroupIds>
            </configuration>
        </execution>
    </executions>
</plugin>
```

- Copia de resources al target:

```
<plugin>
    <artifactId>maven-resources-plugin</artifactId>
    <version>2.5</version>
    <executions>
        <execution>
            <id>copy-resources</id>
            <phase>validate</phase>
            <goals>
                <goal>copy-resources</goal>
            </goals>
            <configuration>
                <outputDirectory>${basedir}/target/resources</outputDirectory>
                <resources>
                    <resource>
                        <directory>src/main/resources</directory>
                        <filtering>true</filtering>
                    </resource>
                </resources>
            </configuration>
        </execution>
    </executions>
</plugin>
```

- Plug-in per aparellar les versions de Locale i rutes:

```
<plugin>
    <groupId>org.sonatype.flexmojos</groupId>
    <artifactId>flexmojos-maven-plugin</artifactId>
    <version>${flex.mojos.version}</version>
    <extensions>true</extensions>
    <configuration>
        <sourceFile>swf.mxml</sourceFile>
        <localesSourcePath>${project.build.directory}/resources/locales/{locale}</localesSourcePath>
        <localesCompiled>
            <locale>en_US</locale>
            <locale>es_ES,fr_FR</locale>
        </localesCompiled>
        <allowSourcePathOverlap>true</allowSourcePathOverlap>
    </configuration>
</plugin>
```

- Plug-in per aparellar les versions el theme Halo utilitzat al projecte sense desestimar Spark:

```
<plugin>
    <artifactId>maven-dependency-plugin</artifactId>
    <executions>
        <execution>
            <id>copy-themes-for-flex4</id>
            <phase>generate-resources</phase>
            <goals>
                <goal>copy-dependencies</goal>
            </goals>
            <configuration>
                <includeTypes>swc</includeTypes>
                <outputDirectory>target/themes</outputDirectory>
                <overWriteIfNewer>true</overWriteIfNewer>
                <includeGroupIds>com.adobe.flex.framework</includeGroupIds>
                <includeArtifactIds>spark,halo</includeArtifactIds>
                <includeClassifiers>theme</includeClassifiers>
                <stripVersion>true</stripVersion>
            </configuration>
        </execution>
    </executions>
</plugin>
```

4.2.3 Pom war.

Aquest es el mes complexa, ja que ha de igualar les generacions arxius necessaris a la compilació d'Eclipse. A mes la generació del Wrapper està inclosa a aquest pom, on

- Versionat de compilació:

```
<plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
        <source>1.6</source>
        <target>1.6</target>
    </configuration>
</plugin>
```

- Copia de resources Flex.

```
<plugin>
    <groupId>org.sonatype.flexmojos</groupId>
    <artifactId>flexmojos-maven-plugin</artifactId>
    <version>3.6.1</version>
    <executions>
        <execution>
            <goals>
                <goal>copy-flex-resources</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

- Generació del Wrapper:

```
<plugin>
    <artifactId>maven-antrun-plugin</artifactId>
    <executions>
        <execution>
            <phase>compile</phase>
            <configuration>
                <tasks>
                    <taskdef className="flex.ant.HtmlWrapperTask" name="html-wrapper"
                        classpath="C:\Program Files (x86)\Adobe\Adobe Flash Builder 4
                        Plug-in2\sdks\4.0.0\ant\lib\flexTasks.jar" />
                    <html-wrapper title="TITLE:${flash.html.title}"
                        file="index.html" height="100%" width="100%" bgcolor="red"
                        application="swf" swf="swf-0.0.1-SNAPSHOT" version-major="9"
                        version-minor="0" version-revision="0" history="true"
                        output="${project.build.directory}/${project.build.finalName}" />
                </tasks>
            </configuration>
            <goals>
                <goal>run</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

- Configuració d'Hibernate3 amb MySQL. Ens permet generar des de Maven les taules necessàries si no existeixen a partir de les anotacions i els mapatge de les entitats.

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>hibernate3-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <verbose>true</verbose>
        <components>
            <component>
                <name>hbm2ddl</name>
                <implementation>jdbcconfiguration</implementation>
            </component>
            <component>
                <name>hbm2hbmxml</name>
                <outputDirectory>src/main/resources</outputDirectory>
            </component>
        </components>
        <componentProperties>
            <drop>true</drop>
            <configurationfile>/src/main/resources/hibernate-cfg.xml</configurationfile>
        </componentProperties>
    </configuration>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.9</version>
        </dependency>
    </dependencies>
</plugin>
```

- Configuració necesaria pel llençament dels tests Junit:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.11</version>
    <configuration>
        <includes>
            <include>**/Test*.java</include>
            <include>**/*Test.java</include>
            <include>**/*TestCase.java</include>
        </includes>
    </configuration>
</plugin>
```

- Configuració per la generació del Javadoc:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <executions>
        <execution>
            <id>attach-javadocs</id>
            <goals>
                <goal>jar</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

4.3 Integració de Frameworks.

A aquest apartat descriurem el conjunt de fitxers i configuracions que calen per que la nostra arquitectura MVC funcioni correctament amb els frameworks utilitzats. La major part d'ells romandran a la part servidora i l'altre a Flex:

4.3.1 Configuracions Servidor.

Al servidor podrem destacar el següent conjunt de fitxers:

- **web.xml:** Aquest fitxer contindrà les configuracions del messagebroker de flex, que son gestionades per un DispatcherServlet. A mes estaran configurats els escoltadors i la configuració del context mitjançant Spring:

```
<servlet>
    <servlet-name>flex</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>flex</servlet-name>
    <url-pattern>/messagebroker/*</url-pattern>
</servlet-mapping>

<display-name>Invoicing Intranet</display-name>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        classpath*:spring/app-config.xml
    </param-value>
</context-param>

<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<listener>
    <listener-class>flex.messaging.HttpFlexSession</listener-class>
</listener>
```

- **app-config.xml**: Aquest fitxer contindrà les configuracions dels beans principals tant de Serveis com de DAOs, datasources, fitxer de propietats jdbc i la plantilla de SessionFactory per la comunicació amb Hibernate:

```
<context:annotation-config />
<context:component-scan base-package="com.invoicing.services" />
<bean id="invoicingService" class="com.invoicing.services.InvoiceServiceImpl" />
<bean id="userService" class="com.invoicing.services.UserServiceImpl" />

<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="${jdbc.driverClassName}" />
    <property name="url" value="${jdbc.url}" />
    <property name="username" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
</bean>
<bean id="propertyConfigurer"
    class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
        <list>
            <value>classpath:jdbc.properties</value>
        </list>
    </property>
</bean>
<bean name="userDAOI" class="com.invoicing.persistence.UserDAOImpl">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
    <bean name="invoiceDAOI" class="com.invoicing.persistence.InvoiceDAOImpl">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>

<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="configLocation" value="classpath:hibernate-cfg.xml" />
    <property name="hibernateProperties">
        <value>
            hibernate.dialect= org.hibernate.dialect.MySQL5InnoDBDialect
            hibernate.query.substitutions=true 'Y', false 'N'
            hibernate.cache.use_second_level_cache=false
            hibernate.hbm2ddl.auto=update
            hibernate.use_sql_comments=true
            hibernate.show_sql=true
    </value>
    </property>
</bean>
```

- **hiberante-config.xml**: Aquest fitxer contindrà la declaració de les entitats pel seu correcte mapatge:

```
<hibernate-configuration>
    <session-factory name="sessionFactory">
        <mapping class="com.invoicing.entities.User" />
        <mapping class="com.invoicing.entities.Invoice" />
    </session-factory>
</hibernate-configuration>
```

- **jdbc.properties**: Contindrà les configuracions d'accés a les diverses Bases de Dades (Local i del servidor extern).
- **services-config.xml**: Contindrà les declaracions dels canals de comunicació amb flex, i altres configuracions per posteriors ampliacions, com per exemple, les configuracions de seguretat basades en Rols.

```
<services>
    <default-channels>
        <channel ref="my-amf" />
         <channel ref="my-polling-amf"/>
         <channel ref="my-secure-amf"/>
    </default-channels>
</services>

<channel-definition id="my-amf"
    class="mx.messaging.channels.AMFChannel">
    <endpoint
        url="http://{server.name}:{server.port}/{context.root}/messagebroker/amf"
        class="flex.messaging.endpoints.AMFEndpoint" />
</channel-definition>

<channel-definition id="my-secure-amf"
    class="mx.messaging.channels.SecureAMFChannel">
    <endpoint
        url="https://{server.name}:{server.port}/{context.root}/messagebroker/amfsecure"
        class="flex.messaging.endpoints.SecureAMFEndpoint" />
    <properties>
        <add-no-cache-headers>false</add-no-cache-headers>
    </properties>
</channel-definition>
```

- **flex-servlet.xml**: Aquí tindrem les declaracions dels Remote Objects amb els que es comunicaran els serveis amb Flex.

```
<flex:message-broker>
    <flex:remoting-service
        default-channels="my-amf, my-secure-amf" />
    <flex:message-service
        default-channels="my-streaming-amf,my-longpolling-amf,my-polling-amf" />
</flex:message-broker>

<flex:remoting-destination ref="invoicingService" />
<flex:remoting-destination ref="userService" />
```

4.3.2 Configuracions Client.

A la banda client Flex, també caldran algunes configuracions i tècniques especials per la comunicació amb la banda servidora.

- **Remote Objects**: Aquest objectes han d'apuntar als canals que hem configurat a la banda servidora, allà on siguin utilitzats (Forms, Screens,...)

```
<mx:RemoteObject id="authentification"
                 destination="userService"
                 result="onResult(event)"
                 fault="onFault(event)"
                 endpoint="http://localhost:8080/war/messagebroker/amf"/>
```

- **Anotacions**: Anotacions especials ens caldran per la sincronització dels nostres DTOs de la part Java a la Flex. Les anotacions també es podrien utilitzar per crear Delegates en comptes de instanciar RemoteObjets, amb Spring.

```
[RemoteClass(alias="com.invoicing.dtos.UserDTO")]
```

- **InvoicingMod.mxml**: Aquí es definirà la configuració que volem tindre. En cas d'utilitzar mòduls diferents, tindríem mes d'una:

```
<fx:Declarations>
    <parsley:ContextBuilder>
        <parsley:FlexConfig type="{InvoicingConfig}"/>
    </parsley:ContextBuilder>
</fx:Declarations>
```

- **InvoicingConfig.mxml**: Contindrà la declaracions per la internacionalització i la utilització de Parsley:

```
<fx:Script>
    import forms.AuthentificationForm;
    import screens.*;
</fx:Script>
<fx:Metadata>
    [ResourceBundle("invoicingProperties")]
</fx:Metadata>
<fx:Declarations>
    <screens:Screen/>
    <screens:AuthentificationScreen/>
    <screens:HomeScreen/>
    <screens:InvoicingScreen/>
    <filters:InvoicingFilter/>
    <grids:InvoicingGrid/>
    <forms:AuthentificationForm/>
    <parsley:View type="{Screen}"/>
    <parsley:View type="{AuthentificationScreen}"/>
    <parsley:View type="{HomeScreen}"/>
</fx:Declarations>
```

- .**actionScriptProperties:** Aquest es un dels mes conflictius, ja que hi ha que indicar moltes configuracions diferents sobre les dependencies que genera Maven, els themes, els Locales, etc.

```
<actionScriptProperties mainApplicationPath="swf.mxml" projectUUID="6ae7edaa-306d-472c-b564-169940dbcc2c"
  <compiler additionalCompilerArguments="-theme=invoicing.css -qualified-type-selectors=false -locale=es_
    <compilerSourcePath>
      <compilerSourcePathEntry kind="1" linkType="1" path="src/main/resources"/>
      <compilerSourcePathEntry kind="1" linkType="1" path="src/main/resources/theme"/>
    </compilerSourcePath>
    <libraryPath defaultLinkType="1">
      <libraryPathEntry kind="1" linkType="1" path="libs"/>
      <libraryPathEntry kind="4" path="">
        <excludedEntries>
          <libraryPathEntry kind="3" linkType="1" path="${PROJECT_FRAMEWORKS}/libs/flex.swc" useDefaultLi
        </excludedEntries>
      </libraryPathEntry>
    </libraryPath>
    <sourceAttachmentPath/>
  </compiler>
  <theme themeIsDefault="false" themeIsSDK="true" themeLocation="${SDK_THEMES_DIR}/frameworks/themes/Halo
  <applications>
    <application path="swf.mxml"/>
  </applications>
  <modules/>
  <buildCSSFiles/>
</actionScriptProperties>
```

A mes de tota aquest configuració, si nomes es tingues que generar un war per desplegar, amb el nostre Eclipse un la generació automàtica d'aquets fitxers ja tindríem prou. La dificultat resideix llavors en la compenetració de les compilacions que fa Eclipse en Local, i les que fa Maven. Compenetrar les rutes, les sortides al Target, les versions utilitzades de compilació i desenvolupament, etc suposen un punt crucial en aquest tipus de projecte.

Una mala configuració inicial provocaria molts problemes de dependencies al desenvolupament, el que es transforma en moltes hores de improductivitat i un cost econòmic molt elevat. De la bona primera arquitectura e integració, dependrà bona part del èxit final del projecte.

5 Incoicing Intranet.

Fins aquest punt del document, hem fet una descripció de l'arquitectura e integració que pot ser utilitzat per qualsevol tipus de projecte de gran envergadura. A partir d'ara especificarem els detalls sobre l'aplicació implementada, Invoicing Intranet.

5.1 Requeriments.

A Invoicing Intranet serà el inici d'una gran aplicació que pot acaparar qualsevol requeriment a posteriori, per la gran quantitat de tecnologies inserides. L'aplicatiu partirà d'una Autenticació, que donarà accés a totes les funcionalitats. Per una banda tindrem la descripció general del projecte, mitjançant descarregues de codi, captures de pantalla, memòria, vincles a les planes de les planes de les tecnologies utilitzades, vídeos, etc, amb el que podrem contemplar l'amplada del projecte. Per altra banda, tindrem accés al Manteniment d'una entitat Facturació, relacionada amb el usuari autenticat, mitjançant la connexió entre Flex i Java d'aquest projecte J2EE. Aquesta entitat serà un conjunt numèric diari del treball d'un Taxista.

Per la seva temàtica, aquest tipus de aplicació podria estarà orientat a la gestió de Flotes de Taxi. Amb una aplicació d'aquest tipus, aconseguiríem un estalvi de temps a la part de secretaria, ja que els propis usuaris finals podrien introduir les dades diàries individualment, a banda de gaudir dels resums numèrics que es podrien obtenir. Un tipus de producte, que encara no existeix al mercat, ja que aquestes gestions son normalment atribuïdes a administració, i sense contacte interactiu amb l'usuari final, els taxistes.

Tota la interfície ha de ser sobradament robusta, per suportar un sistema de treball intensiu, pel que es centrarà principalment en no deixar de banda cap incertesa sobre les possibles opcions en cada estat de l'aplicació. També gaudirà d'internacionalització, amb un parell d'idiomes disponibles i les necessitats d'accessibilitat web establertes.

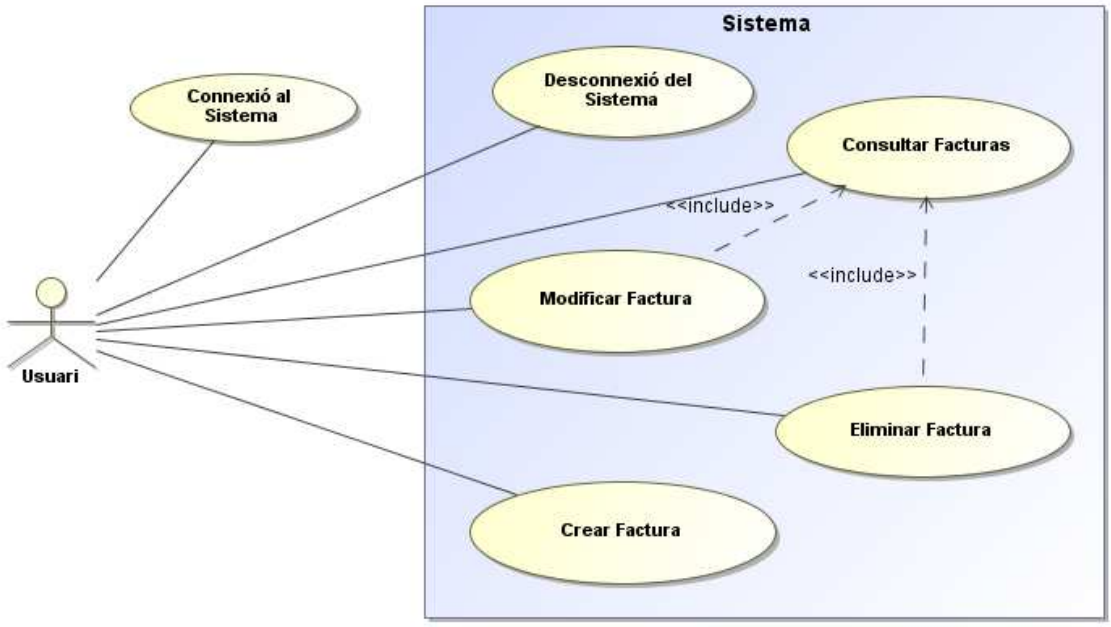
Naturalment, la ubicació serà el servidor d'aplicacions, on tindrem una part pública inicial i una part privada amb necessitat d'autentificació.

5.2 Anàlisis.

En aquest apartat es descriuran el cas d'us d'Accés al Sistema, Gestió de Factures (que aglutinarà per la seva semblança, modificació, eliminació i creació d'una Factura) i Consulta de Factures i Sortir del Sistema. Hem d'especificar la relació d'inclusió entre modificació i eliminació amb la consulta, ja que es un pas previ abans de la selecció d'aquesta per afrontar l'acció seleccionada.

A mes, l'aplicació s'ha creat inicialment per tenir un únic actor, el Usuari, que es qualsevol persona que pugui accedir a l'aplicació.

El diagrama resultant es el següent:



5.2.1 Cas d'us Connexió al sistema.

Cas d'Ús	Connexió al sistema
Funcionalitat	La aplicació sol·licitarà un nom d'usuari i una contrasenya per poder accedir al sistema. Verificarà que el usuari existeix a la BD i que la contrasenya es correcte.
Actor	Qualsevol usuari que pugui accedir a l'aplicació.
Casos de uso relacionats	
Precondició	El Usuari ha d'existir a la base de dades del sistema i haurà de conèixer la contrasenya.
Postcondició	El sistema ha validat les dades introduïdes pel Usuari

	donant accés a l'aplicació amb un missatge de benvinguda, i navegació directa a la plana inicial.
Procés normal principal	1. El Usuari introdueix el seu nom de usuari i la contrasenya. 2. El sistema valida les dades introduïdes i dona accés al sistema. 3. El usuari tindrà accés a totes les funcionalitats i navegacions dels sistema.
Alternatives de procés i excepcions	1. El Usuari introdueix un nom i una contrasenya que no existeix a cap registre de la base de dades. 2. El Usuari serà alertat del error comès amb un missatge a la pantalla

5.2.2 Cas d'us Gestió de Factures.

Cas d'Ús	**Gestió de Usuari**
Funcionalitat	Proporciona les operació de creació, eliminació i modificació d'una Factura en funció del Usuari en curs, que ha estat autenticat correctament i accedeix al menú d'edició.
Actors	Usuari.
Casos d'ús relacionats	Nova Factura, Eliminar Factura, Modificar Factura i Consulta de Factures.
Precondició	El Usuari ha d'estar autenticat al sistema i seleccionar una de les opcions si estan habilitades. Modificació i eliminació només romandran habilitades si s'ha fet una selecció al resultat de la Consulta de Factures.
Postcondició	Totes les dades de cada Factura son obligatòries, ja que el resum diari les proporciona totes i son necessàries per conservar la congruència de la Facturació. Si s'han introduït correctament, es efectuarà la acció en curs.
Procés normal principal	1. El Usuari indica al sistema l'acció a realitzar (creació, eliminació o modificació). El sistema visualitza la pantalla que ha sol·licitat. 2. El Usuari després de la selecció de la Factura en el cas de Modificació i Eliminació, i omplint tots els camps del formulari en el cas de Modificació i Creació, podrà realitzar l'acció demanada. 3. El sistema sol·licitarà confirmació per realitzar l'acció després de la seva petició. 4. Després de la confirmació, el sistema realitzarà l'acció demanada.

	5. Es visualitzarà una confirmació de la realització de l'acció demanada, si no hi ha hagut cap problema al servidor.
Alternatives de procés y excepcions	1. Si no s'han omplert tots els camps demanats, el sistema romandrà a l'espera amb avisos a la pantalla sobre els camps que han de ser omplerts correctament.

5.2.3 Cas d'us Consulta de Factures.

Cas d'Ús	**Consulta de Factures**
Resumen funcionalitat	Proporciona les operació de creació, eliminació i modificació d'una Factura en funció del Usuari en curs, que ha estat autenticat correctament i accedeix al menú d'edició.
Actores	Usuari.
Casos de uso relacionats	Gestió de Factures.
Precondició	El Usuari ha d'estar autenticat al sistema. La consulta sempre romandrà habilitada.
Postcondición	Si s'han omplert correctament les dades del criteri de selecció desitjades, el sistema mostrarà un llistat de Factures.
Procés normal principal	1. El Usuari omple els criteris de selecció desitjats. 2. El Usuari demana executar l'acció. 3. El sistema recupera la informació sobre les Factures que compleixen el criteri de selecció i la mostra per pantalla.
Alternatives de procés y excepcions	1. El sistema no recupera cap Factura, i avisa a l'usuari amb un missatge.

5.2.4 Cas d'us Desconnexió del Sistema.

Cas d'Ús	**Desconnexió del Sistema**
Funcionalitat	La aplicació sol·licitarà una confirmació per sortir dels sistema després de que l'usuari accedeixi al botó de desconnexió.
Actor	Qualsevol Usuari que hagi accedit al sistema.
Casos de uso relacionats	Connexió al Sistema
Precondició	El Usuari ha d'estar autenticat al sistema. La consulta sempre romandrà habilitada.
Postcondición	El Sistema després de la confirmació del Usuari neteja la totalitat de l'aplicació i retorna al Usuari a la pantalla Inicial.

Procés normal principal	1. El Usuari pren el botó de desconnexió. 2. El sistema demana la confirmació de desconnexió mitjançant un missatge. 3. El Usuari confirma la desconnexió. 4. El sistema neteja la totalitat de la pantalla i navega fins a la pantalla inicial.
Alternatives de procés y excepcions	1. El Usuari en rebre el missatge de confirmació de desconnexió, cancel·la l'acció. El sistema romandrà a l'espera de noves accions al mateix estat al que es trobava abans del accés a la desconnexió.

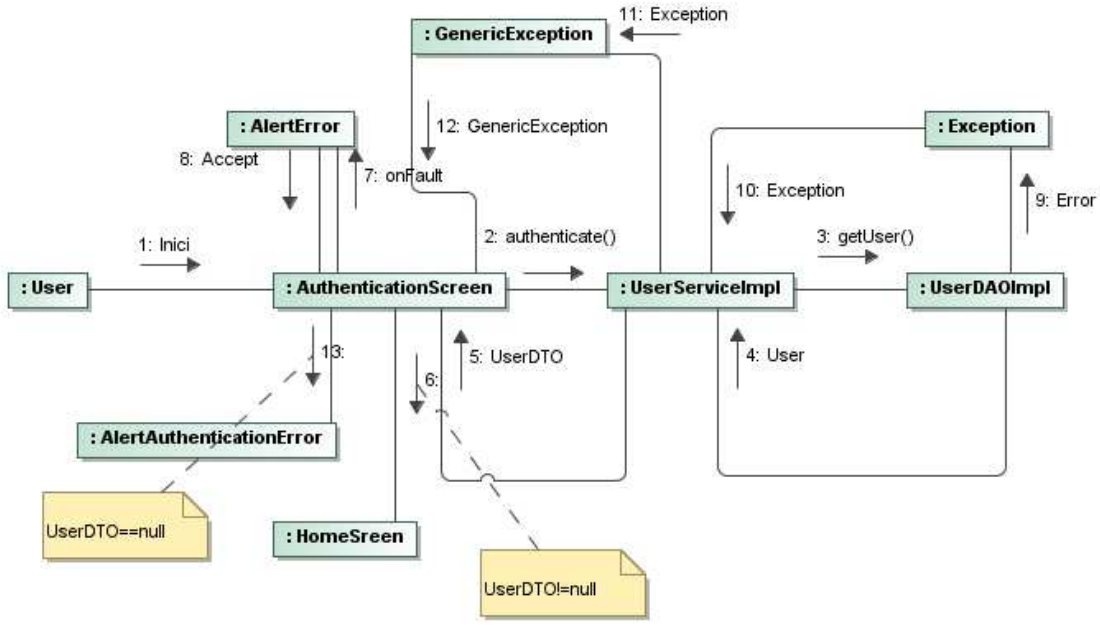
5.3 Disney.

5.3.1 Diagrames de Col·laboració.

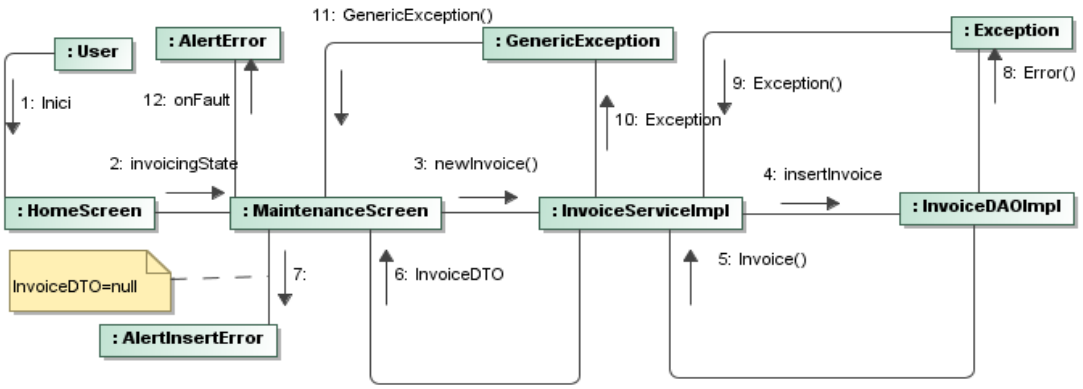
Per la naturalesa del projecte, s'han desenvolupat diagrames de col·laboració en comptes d'estats o seqüència, ja que els objectes d'aquesta implementació inicial no fluiran per gairebé cap estat diferent. També podrem associar millor amb aquest diagrama les tres capes MCV.

He triat tres casos d'us mes representatius, Connexió, Creació de Factura i Consulta de Factures, i els seus paral·lelismes amb son els següents:

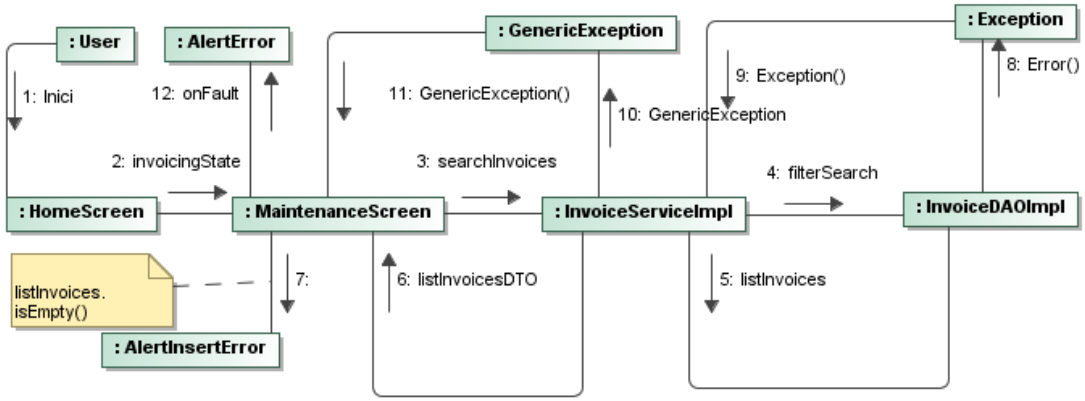
I Diagrama Connexió al Sistema



II Diagrama Alta de Factura

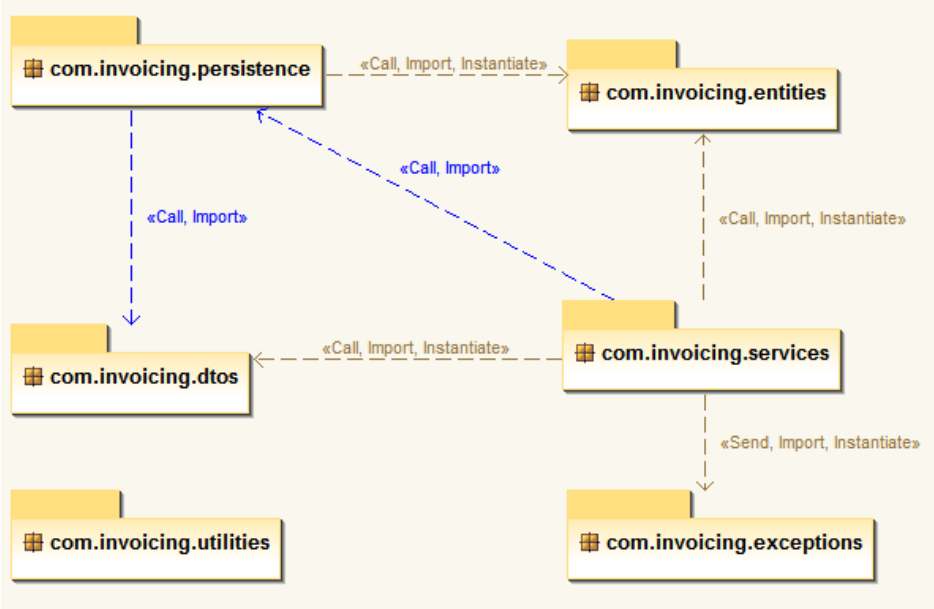


III Diagrama Consulta de Factures



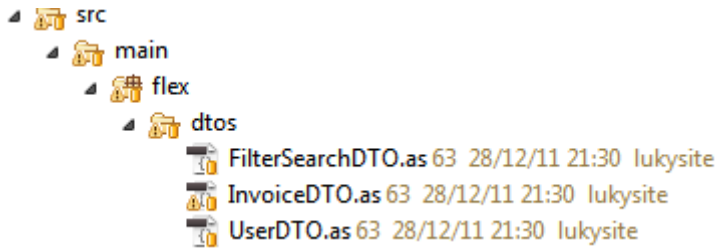
5.3.2 Diagrames de Packages.

El packages han estat creats en funció a l'arquitectura MVC. A la part Java, podrem trobar tota la capa de Controlador als packages Services i DTOs (aquest serà un package que contindrà classes paral·leles a les utilitzades a la vista). A la capa Model tindrem els packages Persistència i Entitats. Utilitats i Excepcions, seran utilitzats per les altres packages on sigui necessari implementar solucions comuns, tant estàtiques com dinàmiques.

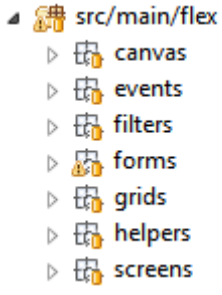


A la capa de la vista, l'organització es mes plana, segons els tipus d'objectes. Domain (que en realitat el tindrem al mòdul SWC), contindrà la totalitat d'objectes paral·lels als DTOs de Java. La resta de packages son suficientment descriptius per la seva nomenclatura:

SWC:

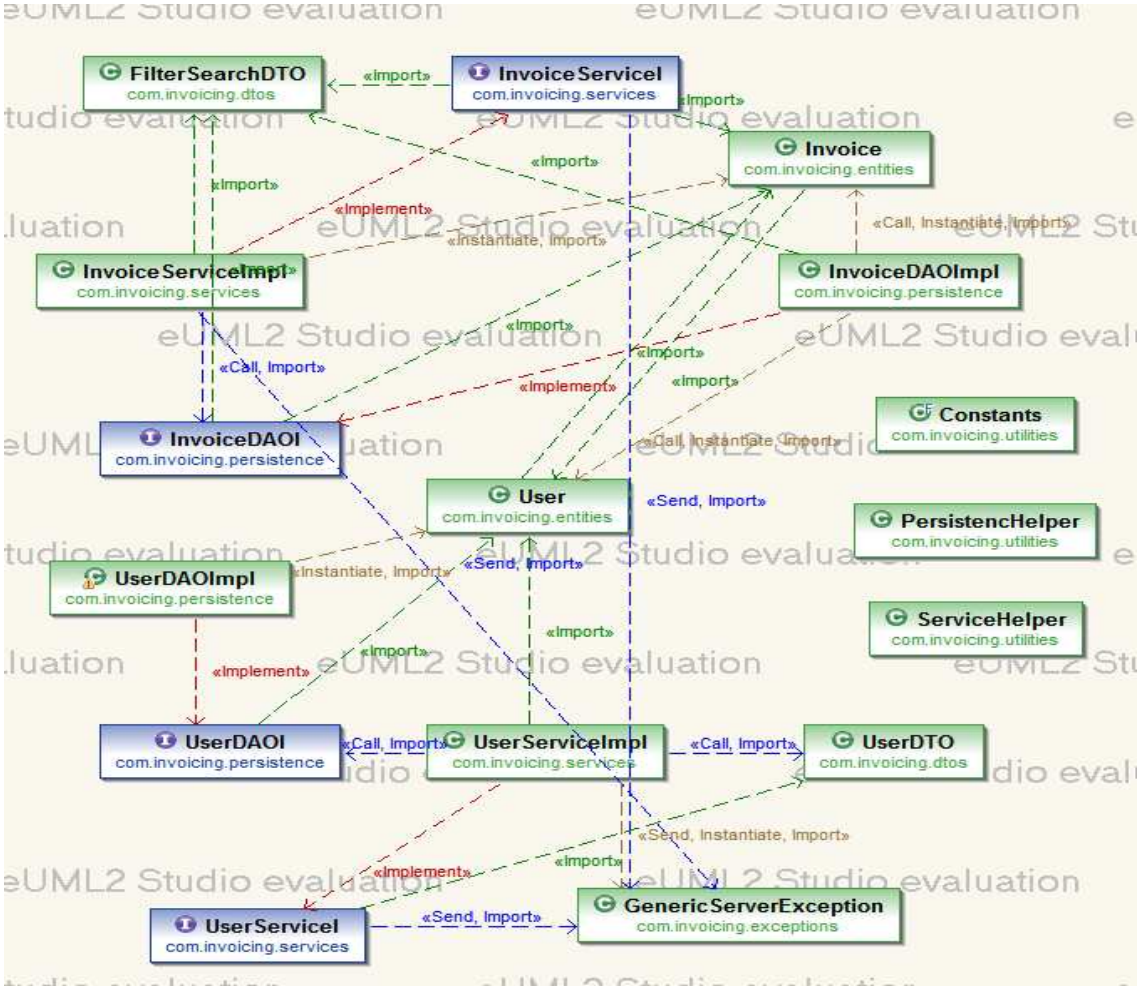


SWF:



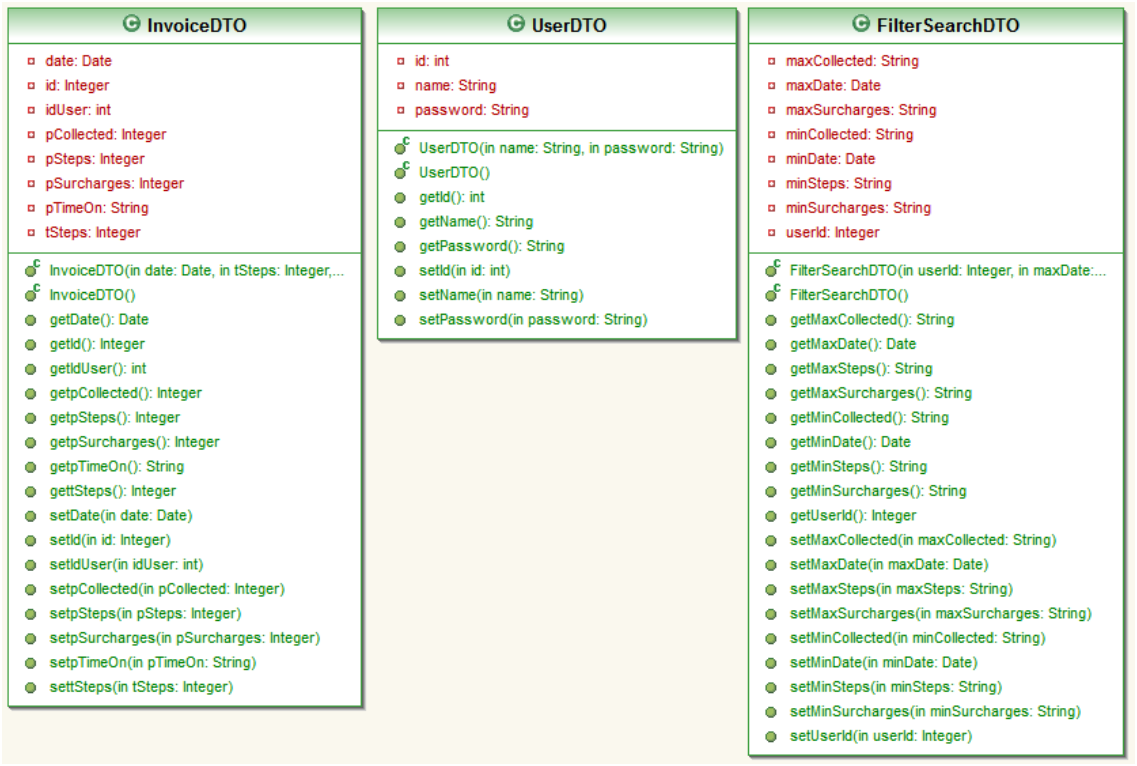
5.3.3 Diagrames de Classe.

En aquest apartat s'ha fet una recopilació de totes les classes utilitzades tant a la part Model i Controlador com a la part de la Vista. La part Model i Controlador, per la seva complexitat ha estat subdividida segons els seus packages per obtenir una descripció mes precisa. A la part de la Vista s'han destacat únicament les classes Frontera, ja que la resta d'Objectes necessaris son paral·lels a la capa Controladora (DTOs), o son objectes genèrics de Flex.



I Diagrama DTOs

Els DTOs (Data Object Transfer) son els objectes paral·lels entre la capa de la Vista i la Controladora. Aquests objectes contindran la informació necessària tant per mostrar per pantalla com per gestionar el negoci de l'aplicació.



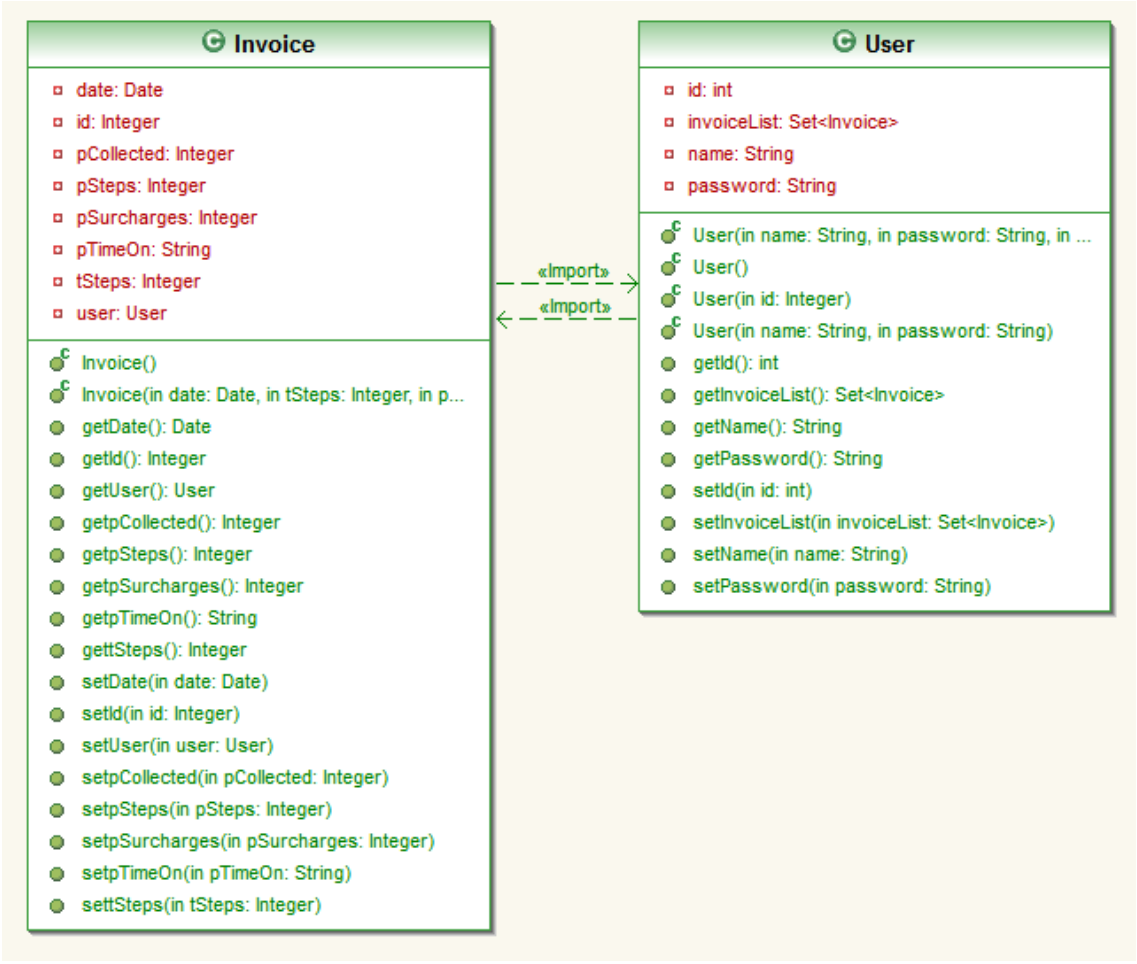
DTO	**UserDTO**	
Descripció de la classe	Classe utilitzada per emmagatzemar les dades relacionades amb un usuari e instanciada en operacions relacionades amb ell mateix.	
Tipus entitat	Propietat: Classe principal	
Característiques	Concreta	
Atributo	**Tipo**	**Descripció**
id	int	Identificador de usuari
name	String	Nom identificatiu del usuari
password	String	Contrasenya del usuari

DTO	**FilterSearchDTO**	
Descripció de la classe	Classe utilitzada per emmagatzemar les dades relacionades amb els criteris de selecció en la Consulta de Factures.	
Tipus classe	Propietat: Classe principal	
Característiques	Concreta	
Atributo	**Tipo**	**Descripció**
maxDate	Date	Data fins a la que es te que filtrar.
maxSteps	Integer	Número de salts màxims fins als que es te que filtrar.
maxSurcharges	Integer	Número de suplements màxims fins als que es te que filtrar.
maxCollected	Integer	Número de salts màxims fins als que es te que filtrar.
minDate	Date	Data des de la que es te que filtrar.

minSteps	Integer	Número de salts màxims des de els que es te que filtrar.
minSurcharges	Integer	Número de suplements màxims des de els que es te que filtrar.
minCollected	Integer	Número de salts màxims des de els que es te que filtrar.

DTO	**InvoiceDTO**	
Descripció de la entitat	Classe utilitzada per emmagatzemar les dades relacionades amb una Factura.	
Tipus entitat	Propietat: classe principal	
Característiques	Concreta	
Atributo	**Tipus**	**Descripció**
id	Integer	Identificador de la Factura.
date	Date	Data de la Factura.
tSteps	Integer	Número de salts totals de la Factura
pSteps	Integer	Número de salts parcials de la Factura
pSurcharges	Integer	Import parcial dels suplements de la Factura
pCollected	Integer	Import parcial recaptat de la Factura
pTimeOn	String	Temps parcial en actiu de la Factura
idUser	Integer	Identificador del Usuari de la Factura.

II Diagrama d'Entitats.

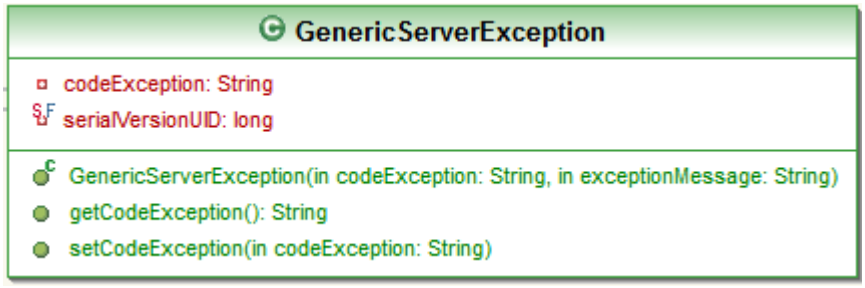


Entitat	**Invoice**	
Descripció de la entitat	Classe utilitzada per emmagatzemar les dades relacionades amb una Factura a la Base de Dades.	
Tipus classe	Propietat: Classe principal	
Característiques	Concreta	
Atributo	**Tipus**	**Descripció**
id	Integer	Identificador de la Factura.
date	Date	Data de la Factura.
tSteps	Integer	Número de salts totals de la Factura
pSteps	Integer	Número de salts parcials de la Factura
pSurcharges	Integer	Import parcial dels suplements de la Factura
pCollected	Integer	Import parcial recaptat de la Factura
pTimeOn	String	Temps parcial en actiu de la Factura
user	User	Objecte Usuari de la Factura.

Entitat	**User**
Descripció de la classe	Classe utilitzada per emmagatzemar les dades relacionades amb un Usuari a la Base de Dades.
Tipus classe	Propietat: Classe principal

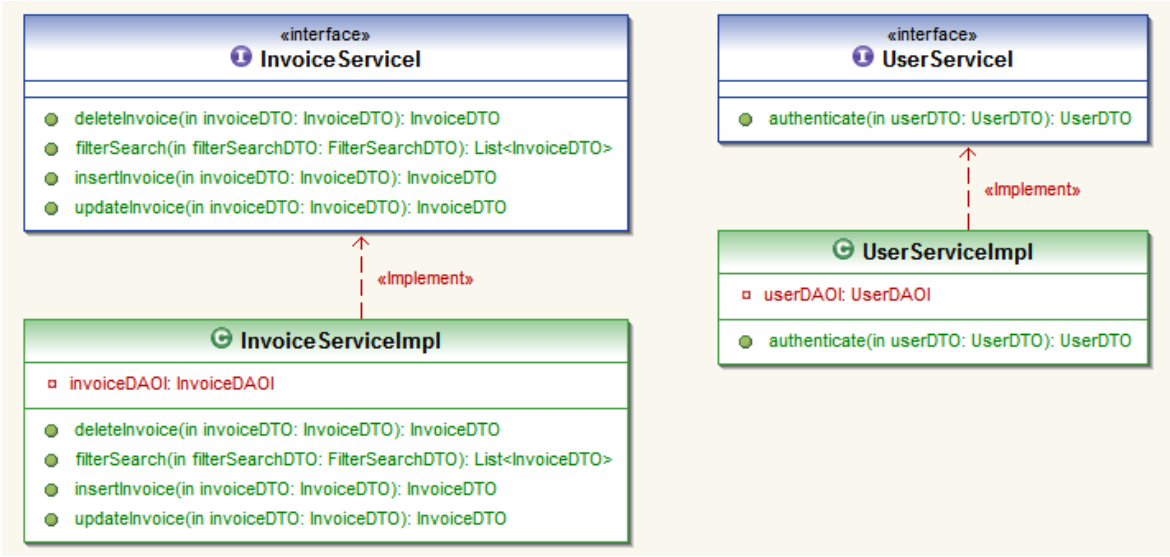
Característiques	Concreta	
Atributo	**Tipus**	**Descripció**
id	int	Identificador de usuari
name	String	Nom identificatiu del usuari
password	String	Contrasenya del usuari
invoiceList	Set	Llista de Factures d'un usuari.

III Diagrama d'Excepcions.



Excepció	**GenericException hereta d'Exception**	
Descripció de la classe	Classe utilitzada per emmagatzemar les dades relacionades amb qualsevol excepció capturada al servidor. A la banda de la vista aquest codi en servirà per donar informació al client.	
Tipus classe	Propietat: classe auxiliar	
Característiques	Concreta, composta.	
Atributo	**Tipus**	**Descripció**
codeException	String	Codi identificatiu de l'excepció.

IV Diagrama de Serveis.



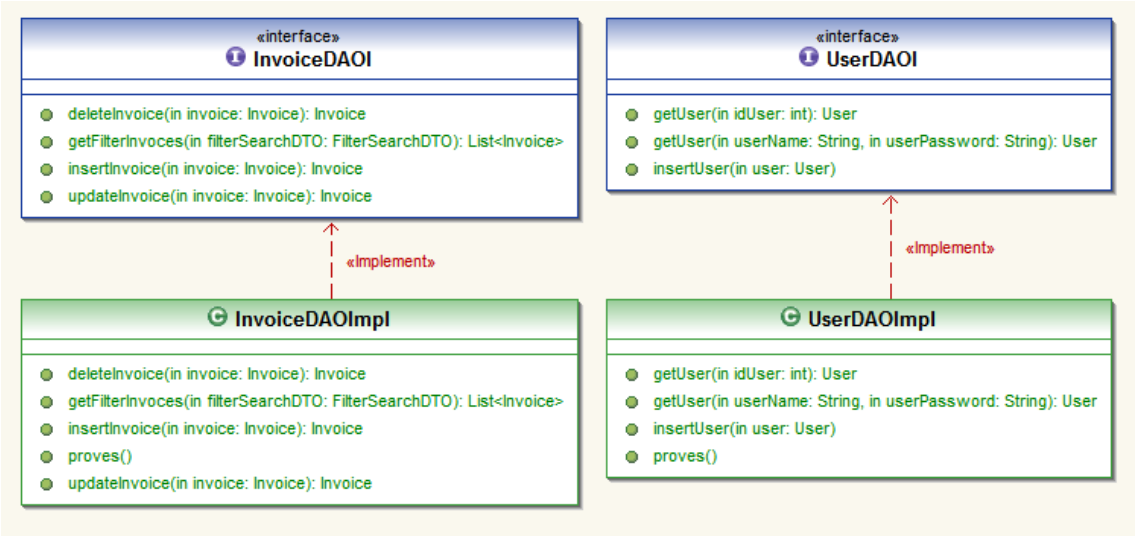
Servicio	**InvoiceServiceI**
Descripció de la Classe	Interface que encapsula els mètodes necessaris de negoci pel manteniment de Factures. Utilitza les Classes de persistència per adequar el Model (Factura) amb la Vista (FacturaDTO).
Tipus de Classe	Propietat: Classe auxiliar
Característiques	Interfície
Responsabilitats	**Col·laboracions/Col·laboradors**
Interface que encapsula els mètodes a implementar i que poden ser utilitzats en instanciar un Objecte que implementi aquesta Interface.	Invoice, FilterSearchDTO, GenericException.
Accés y Mètode	
filterSearch(FilterSearchDTO filterSearchDTO): Declaració del mètode que retorna una llista de InvoiceDTO en funció als criteris de cerca per ser mostrades en pantalla. insertInvoice(InvoiceDTO invoiceDTO): Declaració del mètode que insereix una Factura a la Bases de Dades en funció del InvoiceDTO rebut. Retorna el InvoiceDTO inserit. updateInvoice(InvoiceDTO invoiceDTODeclaració del mètode que modifica una Factura a la Bases de Dades en funció del InvoiceDTO rebut. Retorna el InvoiceDTO actualitzat. deleteInvoice(InvoiceDTO invoiceDTO): Declaració del mètode que elimina una Factura a la Bases de Dades en funció del InvoiceDTO rebut. Retorna el InvoiceDTO eliminat.	

Data Access Object	**UserServiceI**
Descripció de la Classe	Interface que encapsula els mètodes necessaris de negoci pel manteniment de Usuaris. Utilitza les Classes de persistència per adequar el Model (User) amb la Vista (UserDTO).
Tipus de Classe	Propietat: Classe auxiliar
Característiques	Interfície
Responsabilitats	**Col·laboracions/Col·laboradors**
Interface que encapsula els mètodes a implementar i que poden ser utilitzats en instanciar un Objecte que implementi aquesta Interface.	User
Accés y Mètode	
authenticate(UserDTO userDTO):Declaració del mètode que retorna un el Usuari en funció al UserDTO rebut.	

Data Access Object	**InvoiceServicempl implementa InvoiceServiceI**
Descripció de la Classe	Classe que implementa els mètodes necessaris de negoci pel manteniment de Factures. Utilitza les Classes de persistència per adequar el Model (Invoice) amb la Vista (InvoiceDTO).
Tipus de Classe	Propietat: Classe auxiliar
Característiques	Concreta, composta
Responsabilitats	**Col·laboracions/Col·laboradors**
Implementa els mètodes necessaris pel manteniment de instancies Factura amb la Base de Dades, gracies a la utilització de InvoiceDAOImpl.	Invoice, FilterSearchDTO, User.
Accés y Mètode	
getFilterInvoces(FilterSearchDTO filterSearchDTO): Retorna una llista de factures en funció als criteris de cerca. insertInvoice(Invoice invoice): Insereix una Factura a la Bases de Dades. Retorna el InvoiceDTO inserit. updateInvoice(Invoice invoice): Actualitza una Factura de la Bases de Dades. Retorna el InvoiceDTO actualitzat. deleteInvoice(Invoice invoice): Elimina una Factura de la Bases de Dades. Retorna el InvoiceDTO eliminat.	

Data Access Object	**UserServiceImpl implementa UserDAOI**
Descripció de la Classe	Classe que implementa els mètodes necessaris de negoci pel manteniment de Usuaris. Utilitza les Classes de persistència per adequar el Model (User) amb la Vista (UserDTO).
Tipus de Classe	Propietat: Classe auxiliar
Característiques	Concreta, composta
Responsabilitats	**Col·laboracions/Col·laboradors**
Implementa els mètodes necessaris per la autenticació d'un Usuari sobre la Bases de Dades gracies a la utilització de UserDAOimpl.	GenericException, UserDaoI, UserDTO.
Accés y Mètode	
getUser(String userName, String userPassword):Retorna un el Usuari en funció al nom introduït pel client.	

V Diagrama de Persistència.



Data Access Object	**InvoiceDAOI**
Descripció de la Classe	Interface que encapsula els mètodes necessaris pel manteniment de Factures sobre la Bases de Dades.
Tipus de Classe	Propietat: Classe auxiliar
Característiques	Interfície
Responsabilitats	**Col·laboracions/Col·laboradors**
Interface que encapsula els mètodes a implementar i que poden ser utilitzats en instanciar un Objecte que implementi aquesta Interface.	Invoice, FilterSearchDTO.
Acceso y Método	
getFilterInvoces(FilterSearchDTO filterSearchDTO): Declaració del mètode que retorna una llista de Factures en funció als criteris de cerca. insertInvoice(Invoice invoice): Declaració del mètode que insereix una Factura a la Bases de Dades. Retorna la Factura inserida . updateInvoice(Invoice invoice): Declaració del mètode que actualitza una Factura de la Bases de Dades. Retorna la Factura inserida actualitzada. deleteInvoice(Invoice invoice): Declaració del mètode que elimina una Factura de la Bases de Dades. Retorna la Factura eliminada.	

Data Access Object	**UserDAOI**
Descripció de la Classe	Interface que encapsula els mètodes necesaris per la autenticació d'un Usuari sobre la Bases de Dades
Tipus de Classe	Propietat: Classe auxiliar
Característiques	Interfície

Responsabilitats	Col·laboracions/Col·laboradors
Interface que encapsula els mètodes a implementar i que poden ser utilitzats en instanciar un Objecte que implementi aquesta Interface.	User
Acceso y Método	
getUser(String userName, String userPassword):Declaració del mètode que retorna un el Usuari en funció al nom introduït pel client.	

Data Access Object	InvoiceDAOImpl extiende de HibernateDaoSupport implementa InvoiceDAOI
Descripció de la Classe	Classe que implementa els mètodes necesaris pel manteniment de Factures sobre la Base de Dades.
Tipus de Classe	Propietat: Classe auxiliar
Característiques	Concreta, composta
Responsabilitzades	**Col·laboracions/Col·laboradors**
Implementa els mètodes necessaris pel manteniment de instancies Factura amb la Base de Dades. L'herència ens dona tota totes les funcionalitats necessàries per treballar amb Hibernate.	Invoice, FilterSearchDTO, User.
Access y Mètode	
getFilterInvoces(FilterSearchDTO filterSearchDTO): Retorna una llista de Factures en funció als criteris de cerca. insertInvoice(Invoice invoice): Insereix una Factura a la Bases de Dades. Retorna la Factura inserida. updateInvoice(Invoice invoice): Actualitza una Factura de la Bases de Dades. Retorna la Factura actualitzada. deleteInvoice(Invoice invoice): Elimina una Factura de la Bases de Dades. Retorna la Factura eliminada.	

Data Access Object	UserDAOI extiende de HibernateDaoSupport implementa UserDAOI
Descripció de la Classe	Classe que implementa els mètodes necessaris per la Autenticació d'un Usuari sobre la Base de Dades.
Tipus de Classe	Propietat: Classe auxiliar
Característiques	Concreta, composta
Responsabilitats	**Col·laboracions/Col·laboradors**
Implementa els mètodes	User.

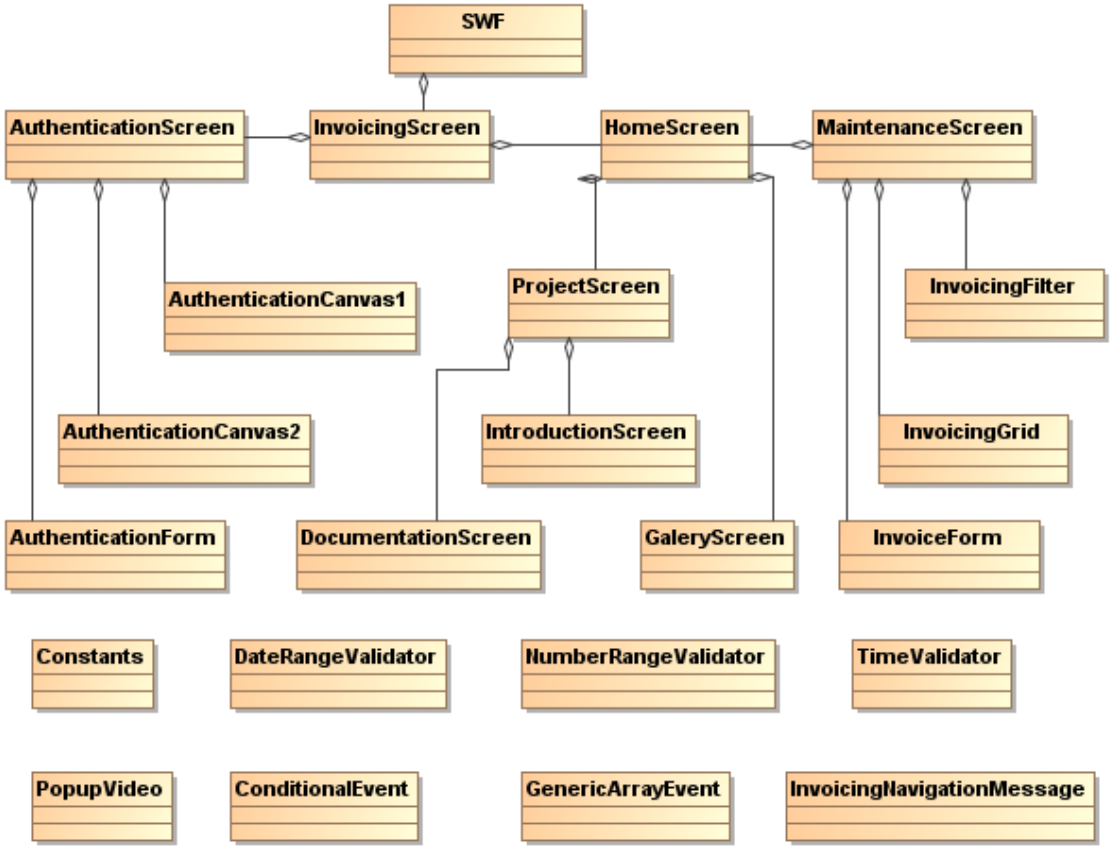
necessaris per la autenticació d'un Usuari sobre la Bases de Dades. L'herència ens dona tota totes les funcionalitats necessàries per treballar amb Hibernate.	
Accés y Mètode	
getUser(String userName, String userPassword):Retorna un Usuari en funció al nom i la contrasenya introduïda pel client. getUser(int idUser):Retorna un Usuari en funció al id introduit.	

VI Diagrama de Utilitats.

Les Classes encapsulades dintre del package Utilitats, implementaran solucions comuns com constants estàtiques per ser utilitzades a tota la part Java. Les necessitats es sorgiran a la fase de Implementació.

VII Diagrama de la Vista.

En aquest diagrama les relacions son d'associació, ja que unes classes MXML inclouen a altres utilitzant-les i comunicant-se amb elles. Aquest diagrama ha estat estructurat també segons la seva navegació, que es correspondrà amb la navegació real de l'aplicació. Derrera de cada Screen, tenim un arxiu .as per l'implementació funcional de cada classe, que ens dona un codi mes net. S'ha utilitzat herencia per poder inplementar noves validacions i events generics reutilitzables a tot arreu.

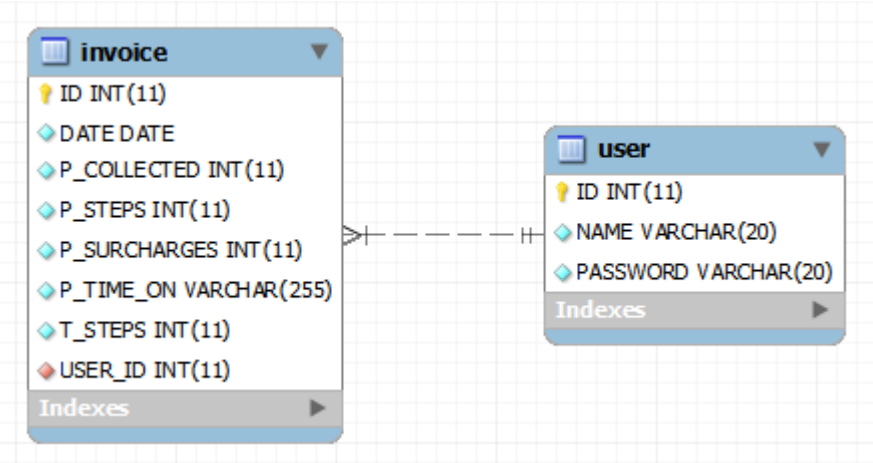


5.3.4 Disseny de Persistència.

El model ER està completament vinculat a la capa de persistència, ja que es genera a partir de les anotacions que ens permet Hibernate inserir a les entitats. Es poden optar per una varietat de configuracions molt ample, de forma que obtenim un paral·lelisme perfecte entre la part Java i la Base de Dades.

Als següents punts descriurem el codi que genera les taules, i les plantilles de cadascuna d'elles. El esquema que es genera es d'una relació One To Many de User a Invoice on Hibernate ens planteja la solució a aquesta problemàtica amb una única clau forana a Invoice, i resol automàticament la llista de Factures per Usuari amb joins interns.

Part important de l'arquitectura es la creació automàtica de les taules si no existeixen. En una instal·lació nova, només hi tindrem que crear una BD amb el nom "invoicing", i en desplegar el war del projecte, es generaran les taules corresponents. Podem imaginar-nos el estalvi de temps que suposa això en un projecte amb una quantitat de taules entre 50 i 1000. De totes formes s'han inclòs els scripts de creació a aquest document:



I Taula User.

Anotacions

```
public class User {

    /** The id. */
    @Id
    @GeneratedValue(strategy = IDENTITY)
    @Column(name = "ID", unique = true, nullable = false)
    private int id;

    /** The name. */
    @Column(name = "NAME", unique = true, nullable = false, length = 20)
    private String name;

    /** The password. */
    @Column(name = "PASSWORD", unique = false, nullable = false, length = 20)
    private String password;

    /** The invoice list. */
    @OneToMany(fetch = FetchType.EAGER, mappedBy = "user")
    private Set<Invoice> invoiceList = new HashSet<Invoice>(0);
```

Fitxa

Clau Primària	Clau Forana	Taula Forana	Nombre	Tipus	Valor Nul	Clau Única
Si	No		ID	int	No	Si
No	No		NAME	varchar	No	Si
No	No		PASSWORD	varchar	No	si

Script

```
CREATE TABLE `user` (
  `ID` int(11) NOT NULL auto_increment,
  `NAME` varchar(20) NOT NULL,
  `PASSWORD` varchar(20) NOT NULL,
  PRIMARY KEY  (`ID`),
  UNIQUE KEY `ID` (`ID`),
  UNIQUE KEY `NAME` (`NAME`),
  UNIQUE KEY `NAME_2` (`NAME`),
  UNIQUE KEY `PASSWORD` (`PASSWORD`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1
```

II Taula Invoice.

Anotacions

```
@Entity
@Table(name = "invoice", catalog = "invoicing")
public class Invoice {

    /** The id. */
    @Id
    @GeneratedValue(strategy = IDENTITY)
    @Column(name = "ID", unique = true, nullable = false)
    private Integer id;

    /** The date. */
    @Temporal(TemporalType.DATE)
    @Column(name = "DATE", nullable = false, length = 10)
    private Date date;

    /** The t steps. */
    @Column(name = "T_STEPS", nullable = false)
    private Integer tSteps;

    /** The p steps. */
    @Column(name = "P_STEPS", nullable = false)
    private Integer pSteps;

    /** The p surcharges. */
    @Column(name = "P_SURCHARGES", nullable = false)
    private Integer pSurcharges;

    /** The p collected. */
    @Column(name = "P_COLLECTED", nullable = false, precision = 2)
    private Integer pCollected;

    /** The p time on. */
    @Column(name = "P_TIME_ON", nullable = false)
    private String pTimeOn;

    /** The user. */
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "USER_ID", nullable = false)
    private User user;
```

Fitxa

Clau Primària	Clau Forana	Taula Forana	Nom	Tipus	Valor Nul	Clau Única
Si	No		ID	int	No	Si
No	No		DATE	date	No	No
No	No		P_COLLECTED	int	No	No
No	No		P_STEPS	int	No	No
No	No		P_SURCHARGES	int	No	No
No	No		P_TIME_ON	varchar	No	No
No	No		T_STEPS	int	No	No
No	Si	User	USER_ID	int	No	No

Script

```
CREATE TABLE `invoice` (
 `ID` int(11) NOT NULL auto_increment,
 `DATE` date NOT NULL,
 `P_COLLECTED` int(11) NOT NULL,
 `P_STEPS` int(11) NOT NULL,
 `P_SURCHARGES` int(11) NOT NULL,
 `P_TIME_ON` varchar(255) NOT NULL,
 `T_STEPS` int(11) NOT NULL,
 `USER_ID` int(11) NOT NULL,
 PRIMARY KEY  (`ID`),
 UNIQUE KEY `ID` (`ID`),
 KEY `FK74D6432D99D2FC56` (`USER_ID`),
 CONSTRAINT `FK74D6432D99D2FC56` FOREIGN KEY (`USER_ID`) REFERENCES `user` (`ID`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1
```

5.3.5 Interfície d'usuari.

Com a punt important destacarem la interface gràfica utilitzada, que gracies a Flex, ens ofereix un conjunt d'eines, llibreries i utilitats que en permet implementar solucions grates i còmodes per l'usuari final. Podem observar la gran varietat d'objectes visuals i moviment de components en aquest vincle:

http://examples.adobe.com/flex3/devnet/networkmonitor/main.html

Els canvis de mida dels components adaptant-se als continguts, themes visualment molt agradables, completa integració amb els estàndards de usabilitat i arquitectures J2EE i un mon de qualitats, fan que aquesta sigui una de les millors interfícies a implementar per gaudir d'un entorn de treball acollidor i atractiu.

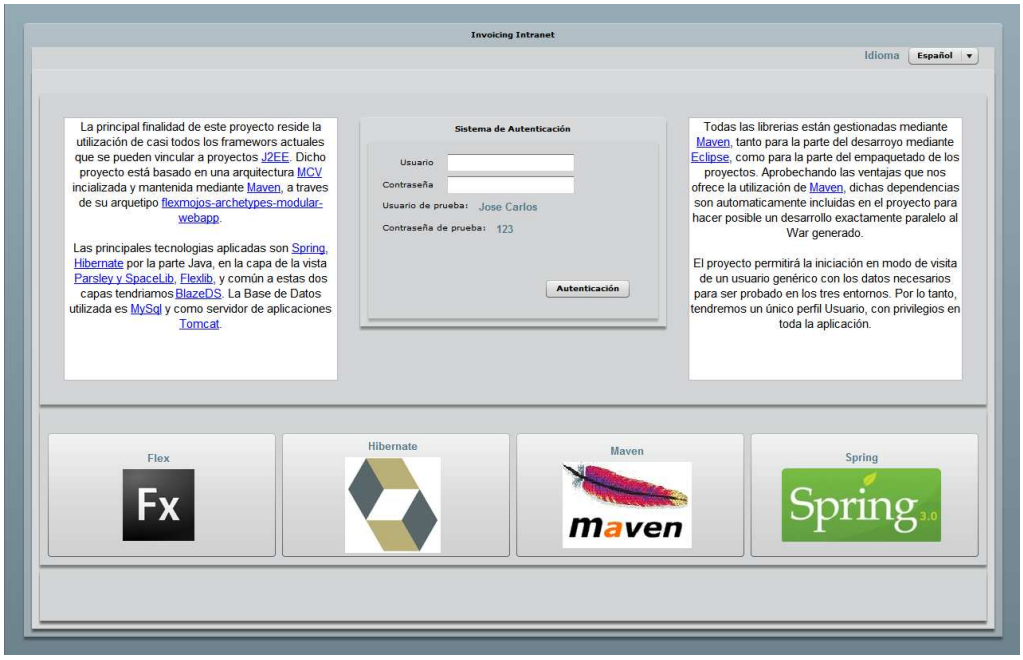
La selecció del Idioma, la podrem obtindre a qualsevol de les pantalles, amb un ràpid canvi de text. Una vegada autenticats, podrem observar el nostre nom d'usuari i la capacitat de desconnectar a la banda superior dreta, sobre la selecció de idioma.

Una vegada autenticats, a totes les pantalles podrem navegar per un menú horitzontal que ens portarà all on vulguem anar.

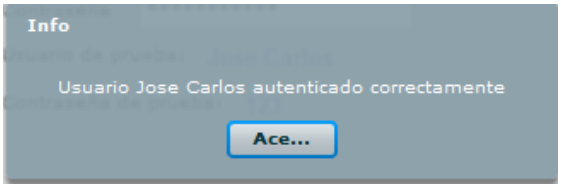
A mes, s'ha implementat Internacionalització a tota la aplicació, amb traducció al castellà i angles:

I Pantalla de Autenticació.

A aquesta tindrem el formulari d'autenticació al centre, amb un petit text explicatiu del projecte i una sèrie de botons amb imatges que ens obriran altres exploradors amb els vincles o imatges seleccionades.

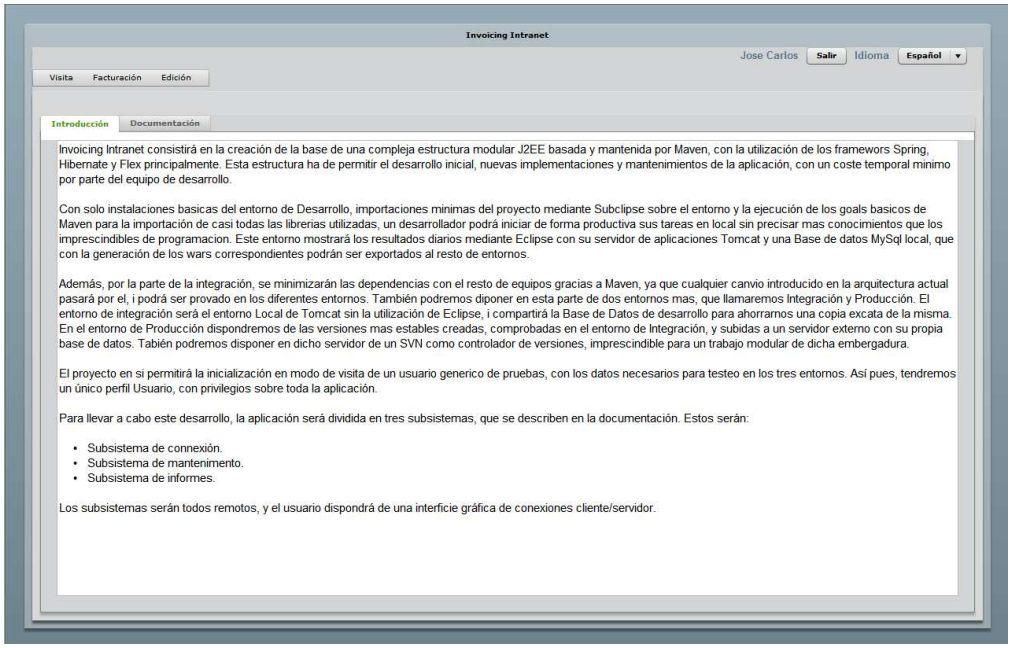


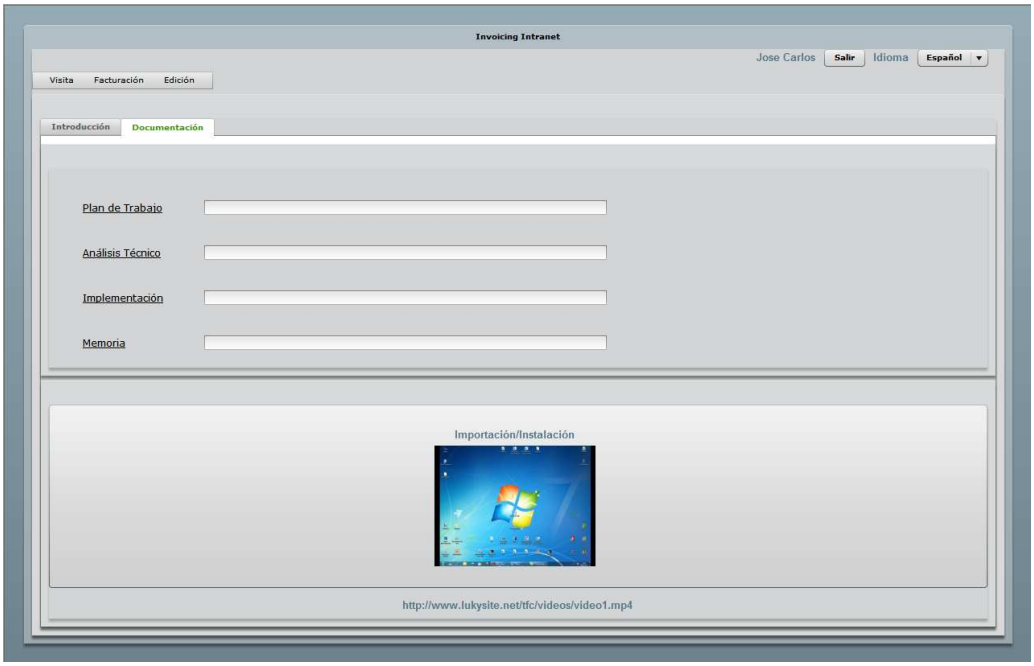
En omplir el formulari, ens confirmarà la nostra entrada al sistema, o el error corresponent.



II Visita / El Projecte.

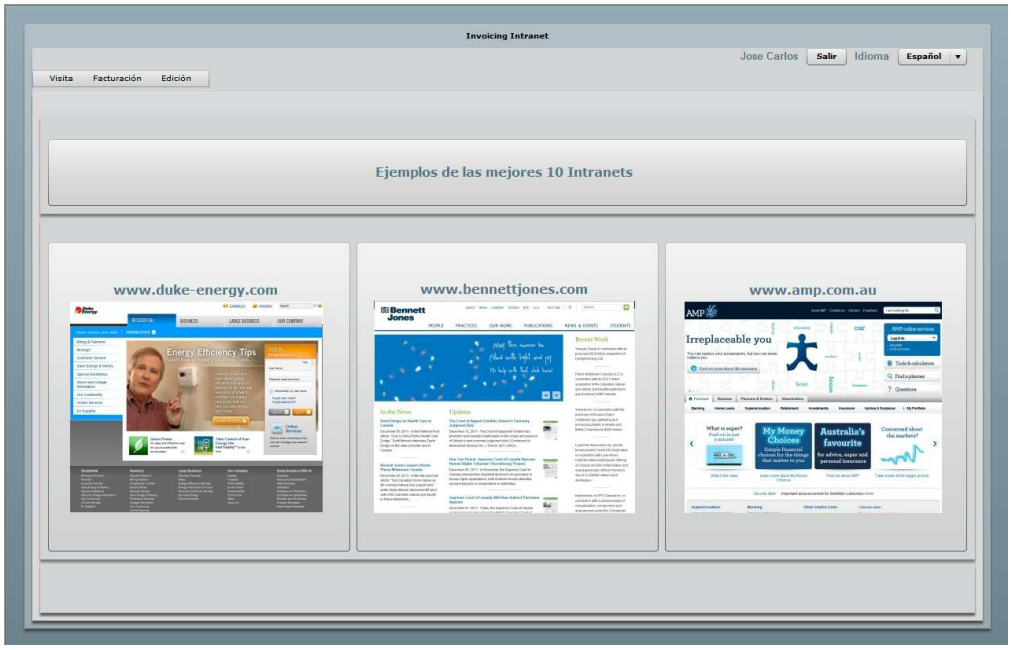
La primera pantalla després de l'autenticació ens portarà a la visita, el projecte, on mitjançant dos tabs (Introducció i Documentació) tindrem un text explicatiu complet del projecte i un espai de vincles i descarregues de tota la documentació generada al projecte (Pacs, Codi, Memòria, video explicactiu, etc).





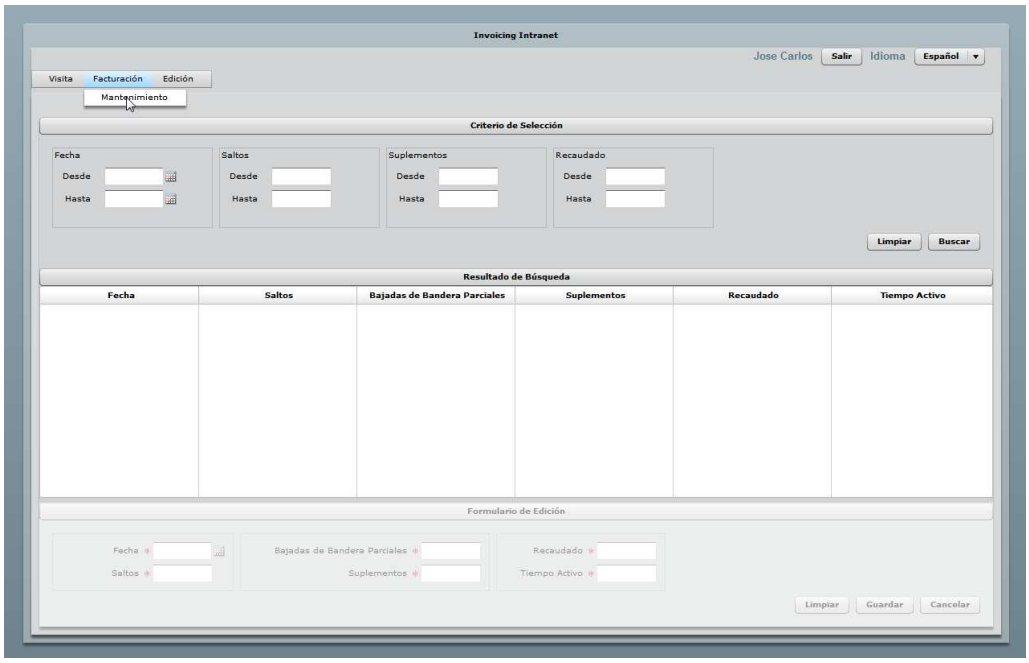
III Visita / Galeria.

La pantalla de Galeria podrem tindrem un vincle a les 10 millors intranets del mercat, i tres exemples d'aquestes.



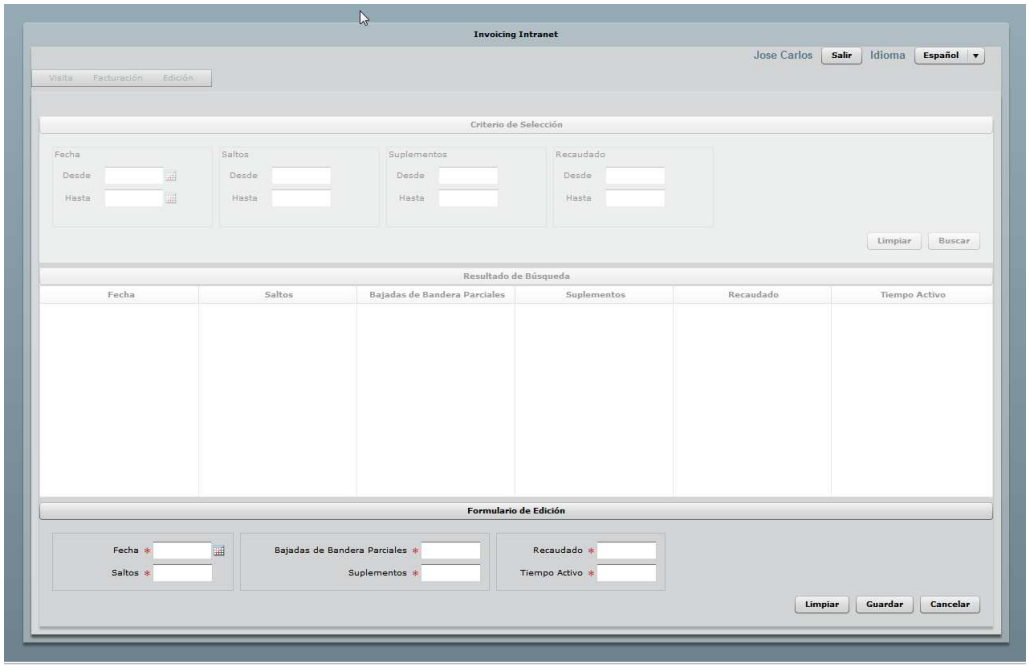
IV Facturació /Manteniment.

A la pantalla de Manteniment podrem accedir per la segona pestanya del menú horitzontal, i englobarà la totalitat funcional dels subsistemes de manteniment e informes.

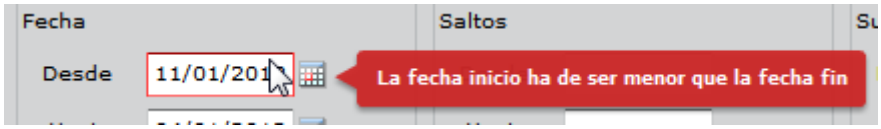


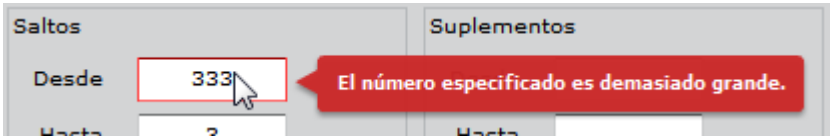
A la darrera pestanya, tindrem les possibles funcionalitats sobre el Manteniment. Podem observar la inhabilitació d'aquest segons les possibilitats portar a terme o no alguna acció. Per exemple, si no tenim carregat el Grid central, i no hem fet una selecció, el botó de modificació i eliminació romandran deshabilitats.

Si estem fent una modificació, els altres dos panells romandran deshabilitats. D'aquesta forma estalviarem als usuaris un mon d'errors de navegació i possibles pèrdues de dades. A mes, l'usuari sempre tindrà el control total del formulari d'edició i del filtre amb opcions de cancel·lar, netejar i guardar/cercar.

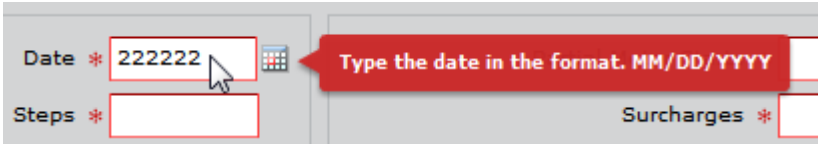


Per donar-li consistència a l'aplicació, les entrades als camps de text han estat validats per obligatorietat i restringides als seus valors permesos, per exemple, els camps numèrics només permeten dígits numèrics, les quantitats d'hores han de ser de la forma HH:MM, les dates DD/MM/YYYY, etc:





La internacionalització també ha estat implementada a totes les validacions:



A mes, la tabulació sense utilització de ratolí ha esta implementada a tot el manteniment per un us intensiu i còmode de l'aplicació. També s'han passat gairebé tots els punts de control com podem observar a un testeig de Hera:

Estado de los puntos de control

Prioridad	Verificar	Bien	Mal	N/A
P1 HERA WCAG 1.0	11	--	--	6
P2 HERA WCAG 1.0	19	3 ✔	1 ✘	6
P3 HERA WCAG 1.0	11	1 ✔	1 ✘	6

5.4 Valoració Econòmica.

La valoració que farem en aquest capítol, no serà una valoració del que suposaria la implementació d'aquesta aplicació, sinó del estalvi que suposaria la utilització d'aquesta Arquitectura e Integració sobre un projecte multitudinari.

Suposem que 400 desenvolupadors d'una gran empesa, ubicats a Barcelona, Madrid, Granada i Pamplona han d'implementar tot el manteniment de facturació de clients en un any. Cada ubicació te 40 desenvolupadors Java (serveis), 40 de la capa vista (Flex), 20 de persistència (BBDD).

Aquesta problemàtica precisa d'una divisió modular i un SVN, amb el que es podran fer incorporacions de codi, sempre després de actualitzar la versió sobre la que treballem i comprovar que els nostres canvis no produeixen cap problema global. Les llibreries i SNAPSHOTs tant externes (hibernate, spring, struts...) com internes (mòdul swf, swc, persistència, serveis, Dtos,...) estarien a un repositori servidor (per exemple http://servidor.empresa.com.repositori) i tothom utilitzaria les mateixes, ja que son descarregades al repositori local (C:\Users\Jose\.m2\repository). Cap desenvolupador podria tenir accés a modificacions sobre els fitxers de configuració amb restriccions sobre el SVN (ignores). Una vegada descarregats els canvis, Maven clean-install ens actualitza i comprova el correcte funcionament de totes les dependencies intermodulars (noves generacions de persistència, serveis, capa vista, canvis d'arquitectura, noves llibreries, tests, ...). En aquest moment si no hi ha problemes, podem reincorporar els nostres canvis. Integració i la comunicació amb el client (Funcionals normalment) sempre te una versió estable al repositori global per mostrar.

Aquesta em sembla la única forma fiable de treballar en un projecte multitudinari... I sense Maven, un projecte Modular i un SVN tindríem un mon de problemes, que suposa un cost addicional en hores perdudes per errors de sincronització entre equips, canvis a l'arquitectura que arriben massa tard als equips de desenvolupadors, etc.

Una hora d'inactivitat o treball perdut per aquestes situacions, multiplicada per 400 desenvolupadors, podem imaginar quin cost pot tenir. A la meva curta experiència com programador, gairebé puc assegurar que una tercera part del temps productiu es malgastat en canvis innecessaris per una incorrecta arquitectura e integració.

5.5 Conclusions.

Com podran observar, les meves inquietuds es centren mes a la Integració i Arquitectura de projectes J2EE. Un bon treball en aquestes dos parts pot facilitar i multiplicar l'avanç productiu dels diferents grups de treball en un projecte distribuït d'àmbit nacional o internacional.

Sobre els llenguatges utilitzats, completament integrats a la Orientació a Objectes, puc denotar que es la forma a la qual mes m'agrada de programar, i que sempre procuro utilitzar.

La meva experiència a la creació d'aquest projecte ha estat molt enriquidora, ja gairebé totes les tecnologies utilitzades eren noves per mi, sobretot la utilització de Maven, i la creació de projectes modulars. La gestió de temps i la divisió dels problemes considero que es una part molt important de qualsevol estudi e implementació, i ens fa donar compte de les nostres errades abans de que es produeixin.

6 Bibliografia.

http://maven.apache.org/

http://maven.apache.org/plugins/maven-eclipse-plugin/reactor.html

http://maven.apache.org/guides/mini/guide-ide-eclipse.html

http://docs.codehaus.org/display/MAVENUSER/Multi-modules+projects

http://es.wikipedia.org/wiki/Java_EE

http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

http://flexmojos.sonatype.org/

http://www.springsource.org/

http://www.hibernate.org/

http://www.spicefactory.org/parsley/

http://code.google.com/p/flexlib/

http://opensource.adobe.com/wiki/display/blazeds/BlazeDS

http://www.eclipse.org/

Beginning Java and Flex Migrating Java, Spring, Hibernate, and Maven Developers to Adobe Flex (Filippo di Pisa).

Maven: The Complete Reference(Tim O'Brien).

7 Glossari.

Artefacte(Artifact): Maven utilitza aquest terme per denominar la unitat mínima que gestiona al seu repositori. Pot ser per exemple un jar, un ear, un war, un zip, etc.

Goal(Maven): Son unitats mínimes d'execució. Poden programar-se en diferents llenguatges, però la majoria estan fets en Java. Un grup de goals formen un plug-in.

IDE (Integrated Development Environment): Programa informàtic compost per un conjunt d'eines de programació.

J2EE: Plataforma de programació per desenvolupar i executar software d'aplicacions en llenguatge Java amb arquitectura de N capes distribuïdes, i que es recolza àmpliament en components de software modulars executant-se sobre un servidor d'aplicacions.

Maven: Eina de software per la gestió i construcció de projectes Java.

SDK(Software Development Kit): Conjunt d'eines de desenvolupament per la creació d'aplicacions per un sistema concret.

Subsistema: Mòdul de la aplicació encarregat d'una part funcional de la mateixa.

Subversion(SVN): Sistema de control de versions dissenyat específicament per substituir al popular CVS. Es software lliure sota una llicencia de tipus Apache/BSD i es mes conegut com SVN, per ser el nom de l'eina utilitzada a la línea de comandes.