Títol del Treball Fi de Carrera

Jordi Fonoll Masalias
ETIG

Jose Juan Rodriguez
25-06-2008

Índex

1. Resum.

He decidit optar per l'**Àrea TFC-J2EE,** perquè volia crear una aplicació Web generada amb software lliure, que pogués treballar en programació d'Orientació a Objectes, fàcil de distribuir.

El treball m'ha permès conèixer l'arquitectura J2EE:

- Estructura d'una aplicació Web
- La configuració del J2EE: Servlets, pàgines JSP(Request i Response),servidors d'aplicacions...
- El servidor d'aplicacions TOMCAT. Quina és la seva funció. Qué és un .WAR i com es pot desplegar dintre del servidor d'aplicacions
- El framework(Struts) que és i quina funció realitza.
- Les diferents capes MVC: qui forma part de la capa vista, model i controlador.
- Què és una capa de persistència, el framework Hibernate.
- AJAX que es i perquè s'utilitza?
- L'editor Eclipse.
- Llibreries externes (DisplayTag)
- Fulls d'estil, codi HTML, Navegadors i les seves utilitats, FireFox(depuració de codi, facilitat d'edició d'HTML).
- Poder conèixer una BBDD de software lliure(MySQL). Eines d'administració(control d'usuaris connectats), de consulta de dades, pool de connexions, tractament de BLOB's.
- Connectors de BBDD, jdbc. Fitxer dataSource.
- Eines de modelatge, UML.(Enterprise architect), construcció de diagrames de seqüència, casos d'ús, diagrama de classes.
- Gestió d'errors, creació de fitxers de traça d'errors (LOG's).

També m'ha permès integrar coneixements que he estudiat al llarg del cicle d'estudis: com crear una planificació i gestió de l'aplicació(Pla de Treball), com elaborar l'anàlisi Funcional i les parts que constarà, que s'ha de tenir en compte en un disseny Tècnic per tal d'agilitar la feina a l'equip de construcció, la codificació amb java i documents d'instal·lació i proves.

2. Agraïments.

Agraït :

- Al Programari Lliure i a tota la gent que hi treballa / ha treballat i l'han fet possible.
- A tota la documentació lliure que pots trobar per la Web.
- Als buscadors i les enciclopèdies digitals.
- Al Jose Juan per l'ajut de com encaminar aquesta difícil tasca de crear un projecte i com presentar-lo
- I també als companys de feina que m'han ajudat a familiaritzar-me en el món del J2ee.

3. Memòria.

3.1.1 Justificació del TFC i context en el qual es desenvolupa

Perquè vaig em he decidit fer aquest projecte?

Fa molts anys que estic treballant com a informàtic. També en fa molts que veig com cada cop viatja més depresa la informació i com cada cop més ens costa desplaçar-nos físicament en menys temps(sobretot en les grans ciutats). Per tant volia fer una aplicació per reduir els temps de transport del dia a dia a la feina als treballadors i també les empreses per reduir els seus costos de transport.

L'objectiu es que no necessiti desplaçar-se a una gran distancia per treballar i així poder-ho fer des d'un lloc mes pròxim a casa o fins i tot des de la mateixa casa, d'aquí ve el nom de la pàgina Work in my House(WimHouse). Les empreses que desitgin una metodologia de treball sense grans costos d'infrastructura poden optar per aquest tipus de contractació.

La decisió d'apostar per aquesta pàgina ha estat:

- La de reduir emissions en atmosfera degut a desplaçaments laborals diaris.
- Reducció de pèrdues de temps en desplaçaments a la feina amb tots els inconvenients que això comporta.
- Que viatgi la informació i no les persones. És mes net.
- Millora en estalvi econòmic i per tant en benestar.
- Donar una possibilitat a gent que té el temps molt just durant la jornada i que sacrifica el seu temps lliure per temps de desplaçaments laborals.

També volia que fos una aplicació creada a partir de software lliure, (potser l'únic però que hi trobo és el no haver-la fet amb sota LINUX). Però el software lliure que em vaig proposar utilitzar al final l'he pogut fer servir:
Java, framework struts, hibernate ,BBDD mySql, Editor Eclipse, FireFox i TOMCAT.

Volia treballar amb Orientació amb Objectes. Sempre he pensat que el codi ha d'estar ben diferenciat, encapsulat, que es pugui reutilitzar fàcilment independentment del projecte. D'aquí la meva aposta per fer-lo amb Java.

Volia crear una aplicació Web perquè l'aplicació és molt més fàcil de distribuir, hi pot arribar tothom sense necessitat d'instal·lar software addicional, amb les dificultats de distribució que això implicaria.

En quan al punt de partida:

- Ha estat un projecte creat des de zero amb la dificultat que això comporta.
- S'ha de tenir en compte que no havia treballat amb l'arquitectura J2EE , que si ho havia fet en java, (durant els estudis d'Enginyeria a la UOC en les semestres anteriors), i que tot això m'ha obligat a fer un gran esforç per entendre tota la complexitat que comporta aquesta arquitectura: els frameworks struts i hibernate, la divisió de capes MVC, persistència, els editors, el servidor d'aplicacions, les llibreries que s'han d'afegir i el propi llenguatge java(abstracció, generalitzacions,...).

3.1.2 Objectius del TFC

Es compleix l'objectiu proposat en el pla de treball :

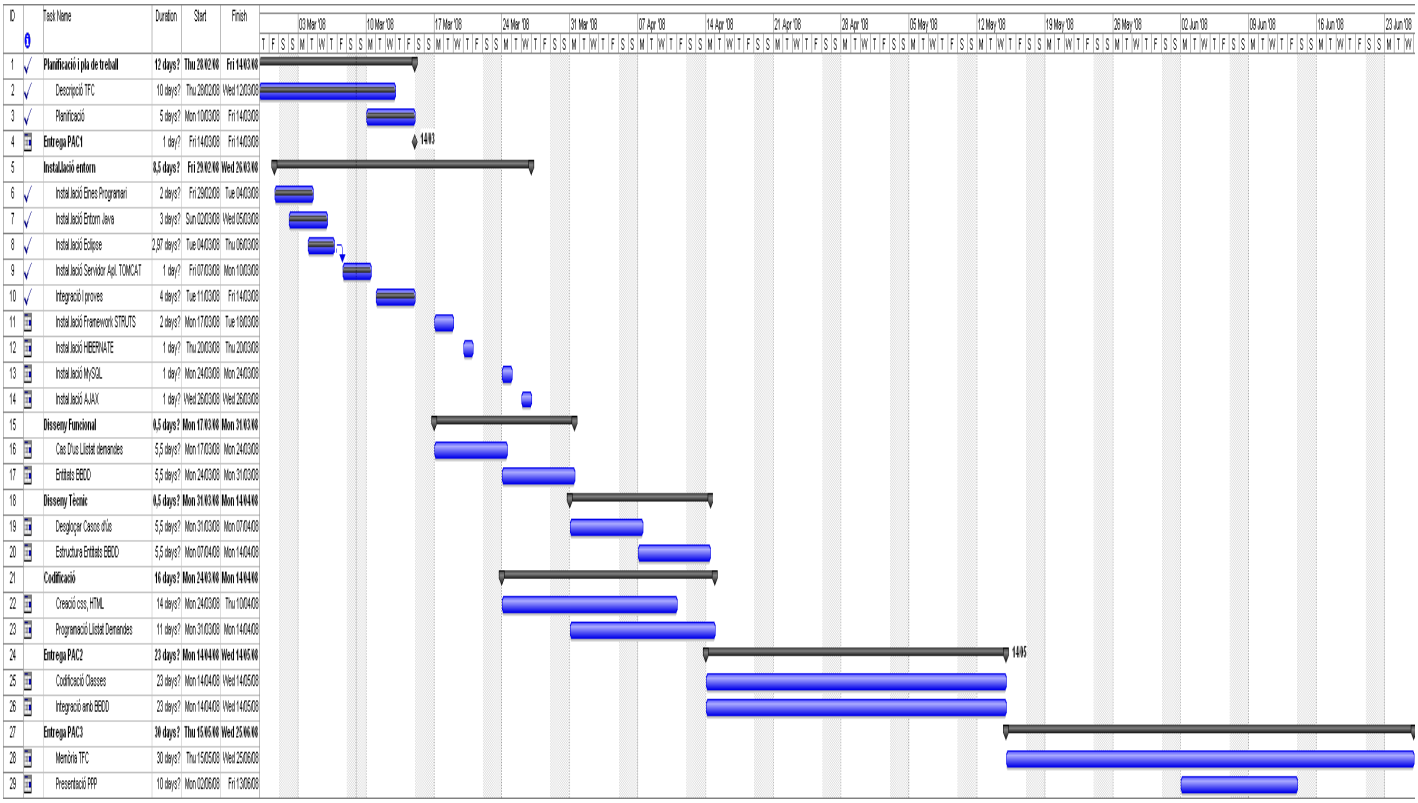
"El projecte consisteix en la creació d'una aplicació WEB construïda sota arquitectura J2EE, aquesta aplicació consistirà en mostrar ofertes laborals definides per empresa, categories, hores de treball i condicions contractuals amb un objectiu molt clar, el que totes les ofertes presentades és puguin realitzar des de casa.

L'objectiu serà crear mitjançant programari lliure una aplicació web a on es pugui posar d'acord gent que està buscant feina per treballar sense necessitat de desplaçar-se i empreses que els ofereixi aquest tipus de feina."

Per tant s'ha creat el pla de treball amb la **planificació**, els dissenys **Funcional** i **tècnic**, la **codificació**, el pla de **proves** i el document **d'instal·lació**.

3.1.3 Planificació del projecte

Planificació inicial:



Planificació final:
Les modificacions de correcció per cada PAC han estat:

Correcció de la PAC2:

- Els documents d'anàlisis funcional i disseny tècnic.
- En el document funcional s'han afegit les pantalles de l'aplicació que faltaven. S'ha definit un model més complert de classes.
- En el document tècnic s'han afegit els diagrames de classes beans, models, accions, forms , DAO i model ER de BBDD.
- En el disseny Tècnic s'ha explicat millor cada cas d'ús que és el que consistiria

Correcció de la PAC3:

- El document de disseny tècnic i la codificació.
- S'ha hagut de separar correctament les capes de Model i control.
- S'ha afegit la part de DAO per cada entitat i DAO abstracta.
- S'ha afegit el log4j per controlar els LOG's.
- S'ha controlat els errors que puguin venir derivats de peticions errònies.
- S'ha documentat millor el codi.

3.1.4 Productes obtinguts

A partir del pla de treball s'han generat els Documents d'anàlisis Funcional, Disseny el tècnic, la codificació i el document de proves i instal·lació. Passaré a descriure els punts mes rellevants cada document.

Document d'Anàlisis Funcional

En ell es reflecteix el producte en la seva versió global es defineixen:

- Els objectius del projecte. Tal com ja anomeno en aquest document. Punt **3.1.2 Objectius del TFC**
- El diagrama de casos d'ús i la definició de cadascun d'ells.

Mostro les interaccions dels actors amb els casos d'ús i també entre els mateixos casos d'ús les generalitzacions que hi ha. En el meu cas vaig optar per dividir en 2 casos d'ús per una millor comprensió del problema.
Diagrama de cas d'ús

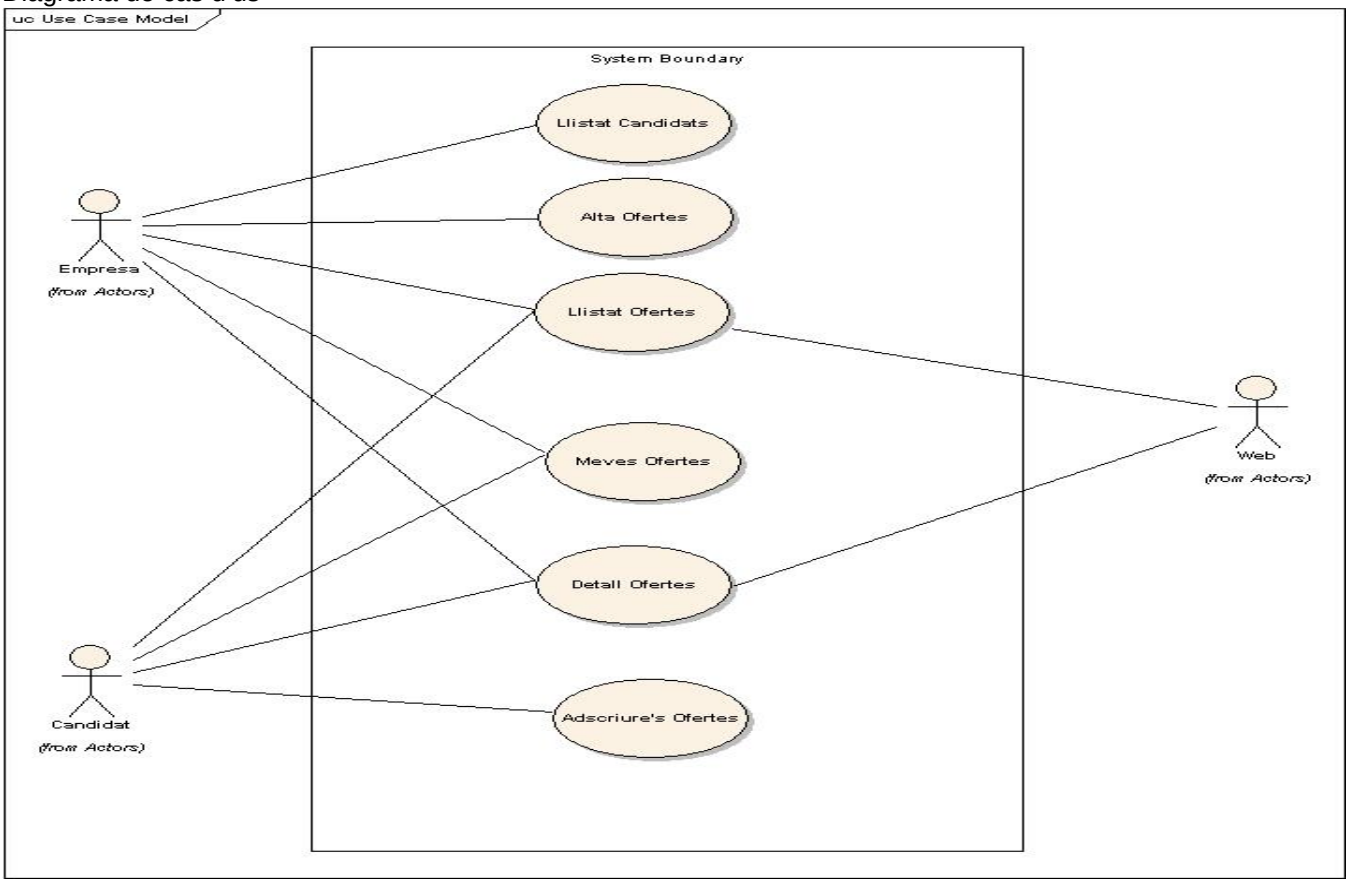


Figura 3.1. Diagrama de casos d'ús.

D'aquest diagrama de casos ens permetrà definir cada **cas d'ús** en particular.
Per exemple: el cas d'ús Adscriure's a una oferta:

He creat cada cas d'ús seguint el mateix criteri amb una descripció del problema(descripció), el que es necessita abans perquè es dugui a terme(pre-condició), com quedarà després d'executar-se(post-condició) i quin serà l'usuari que ho podrà duu a terme(actor-principal), en aquest cas d'ús en concret també hi ha un actor secundari(usuari empresa). El flux amb les accions mostra quins seran els passos que s'executaran perque s'arribi a la post-condició partint de la pre-condició

Cas d'us Adscriure's a una oferta

Descripció: Permet a l'usuari candidat adscriure's a una oferta.
Pre-condició: usuari restà registrat com a candidat.
Post-condició: usuari candidat s'ha adscrit a l'oferta .
Actor principal: Usuari candidat
Actor secundari: Usuari empresa

Flux:

1 Usuari empresa/candidat/web entra a l'aplicació wimhouse.com
2 Es registra com a usuari candidat
3 Usuari candidat accedeix al Llistat d'ofertes
4 Escull una oferta
5 Sistema mostra la pàgina Detall Oferta
6 Usuari candidat polsa el botó **adscriure**.
7 Sistema demana les dades obligatòries per adscriure's a l'oferta
8 Sistema adscriu a l'usuari candidat a l'oferta i en dona conformitat via missatge.

- El diagrama de classes més rellevant. En l'anàlissi Funcional es decideix mostrar el diagrama de classes Beans perquè són les entitats que podrem explicar a nivell de negoci..

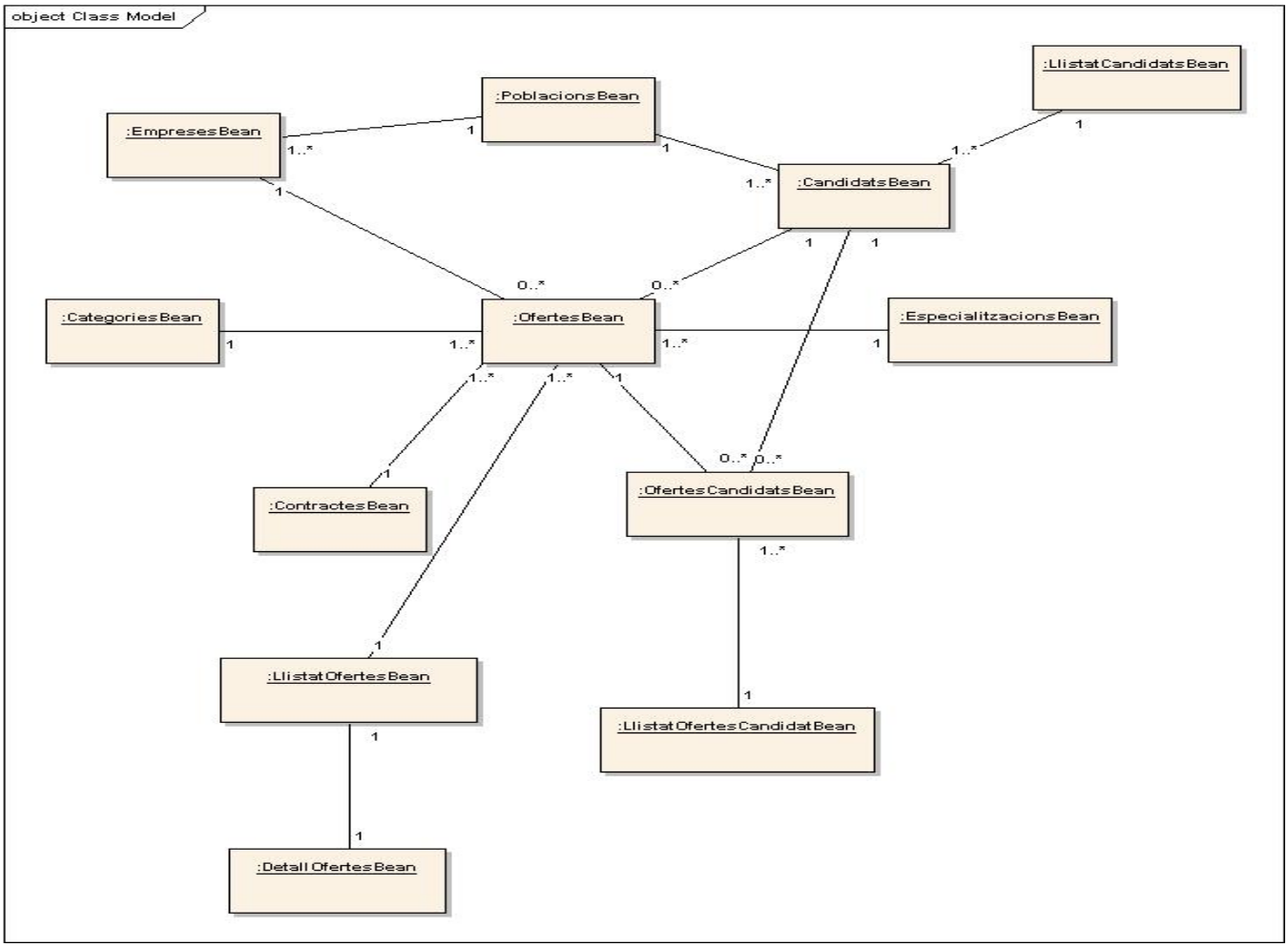


Figura 3.2. Diagrama de Classes

- Diagrama de seqüència per al cas ‘d’ascriure’s a una oferta’

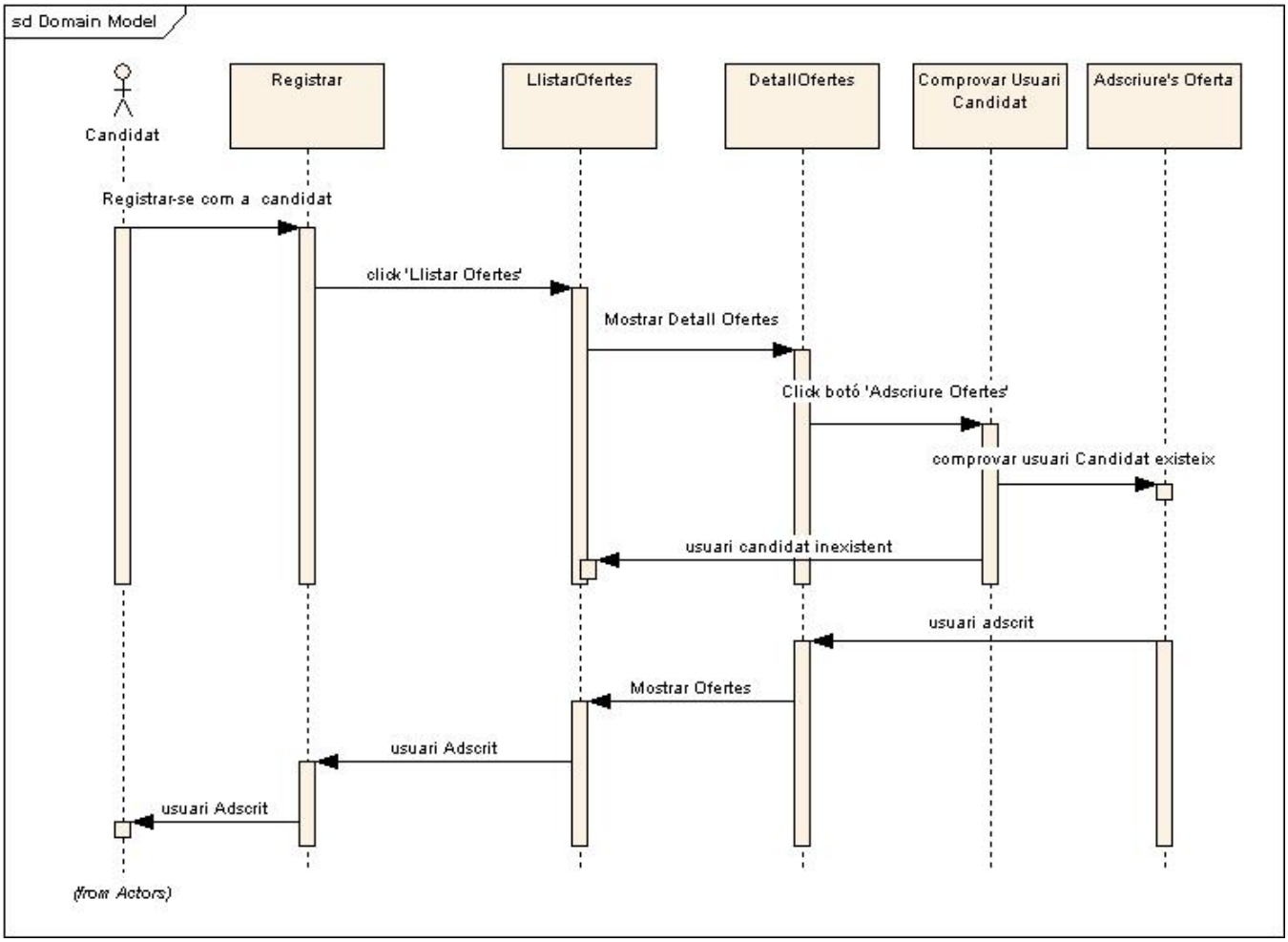


Figura 3.3. Diagrama de Seqüència

- l’Enumeració de les pantalles de l’aplicació i que farà cadascuna d’elles. S’explica en un mode global, més a nivell de negoci que a nivell tècnic(aquesta part ja la tractarem en el DT).

Es mostra el producte final perquè es pugui tenir una idea de com es l’aplicació (el frontend). En ella ja es poden veure aspectes de disseny (el logotip, la divisió de la pantalla, els menús). Per exemple la pantalla ‘d’adscriure’s a una oferta’ tindria la següent visió:

Figura 3.4. Pantalla Adscriure's a una oferta.

El document de d'anàlisi funcional es troba en el fitxer adjunt a aquest document **AF_jordi_fonoll.doc.**

Document de disseny Tècnic

S'explica :

(1) El perquè es decideix treballar en aquesta arquitectura.
(2) L'elecció de cada component que forma part de l'arquitectura i el perquè: Java, Struts, Hibernate, AJAX, Jsp, MySQL.
(3) Els diagrames D'UML. Quins he decidit explicar i perquè.
(4) Exemples de flux de l'aplicació.
(5) Detall de cada cas d'ús.
(6) S' explica el format de la pantalla i les parts de que consta.
(7) Model de dades ER
(8) Programari Utilitzat. Quines llibreries externes s'han escollit i perquè.

(1) El perquè es decideix treballar en aquesta arquitectura

Un dels aspectes que es demana en la PAC2 es perquè decidim treballar en arquitectura J2EE.

La justificació del perquè es treballa en J2EE la he descrit ja en el punt **3.1.6 Justificació del TFC** d'aquesta memòria.

(2) L'elecció de cada component que forma part de l'arquitectura i el perquè: Java, Struts, Hibernate, AJAX, Jsp, MySQL.

STRUTS

El framework era la decisió difícil alhora d'escollir quin utilitzar. He escollit Struts:

Permetia facilitar el model MVC

M'ajudava alhora de deixar el codi net i estructurat aplicant tota la nomenclatura d'accions, de configuració(struts-config) i em facilitava la feina alhora de 'pintar' les pàgines .jsp.
Per ser un dels més estàndards, dels quals hi ha més documentació i per tant la integritat amb eines com els frameworks de persistència o els servidors d'aplicacions son ben estables.
Em vaig decantar per la versió 1.3.8, al ser la release més avançada en la seva versió Struts1. Hi ha la versió Struts 2 , però no la escollir perquè al ser una versió recent podia tenia problemes de consistència en eines com hibernate , editors,....
En el DT vaig crear un diagrama molt útil per entendre que fa el Struts-Config.xml , quines accions són les que s'estan cridant, i quines pàgines es mostren.

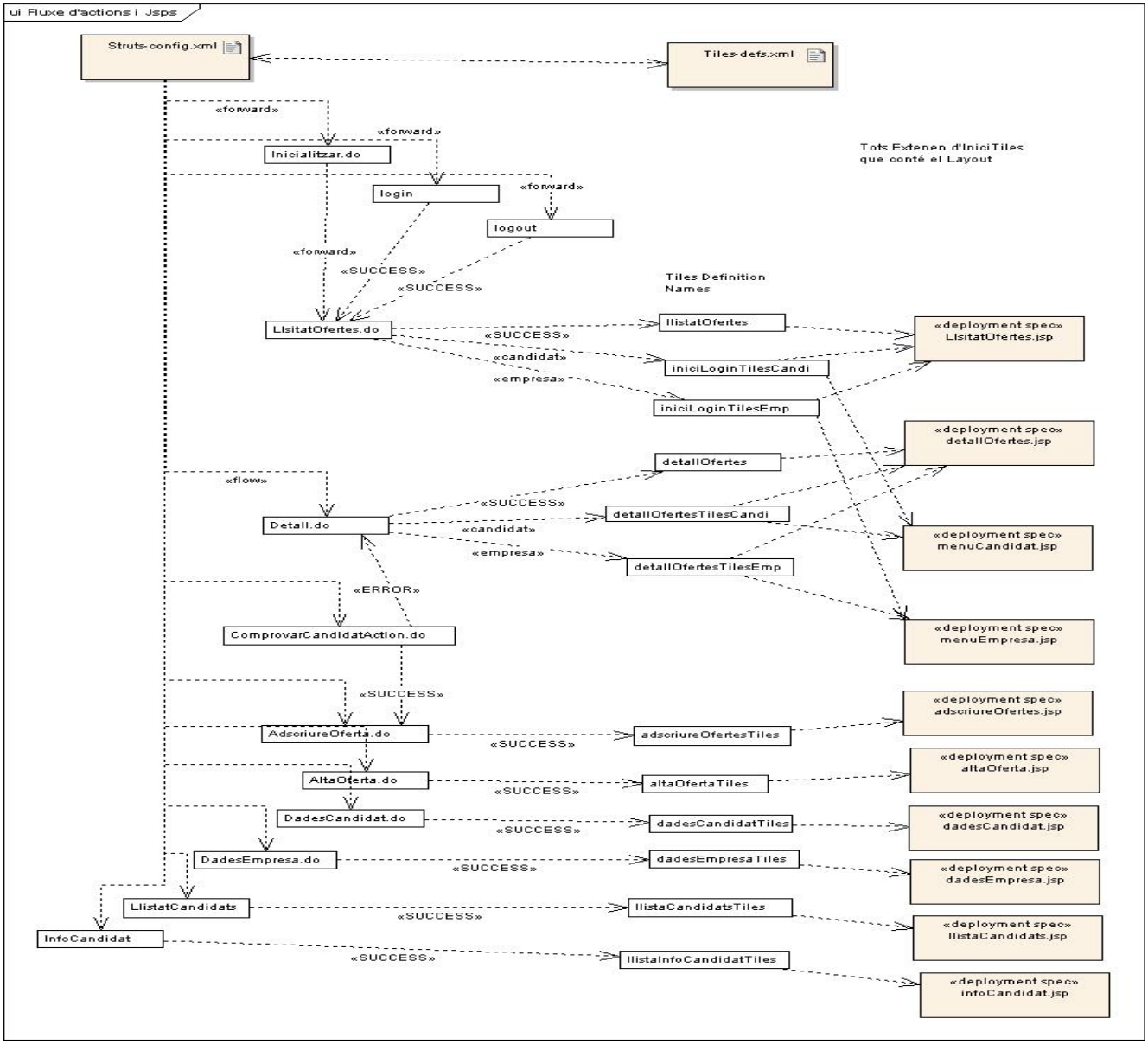


Figura 3.5. Diagrama d'Actions d'struts

Aquest diagrama mostra des del principi la primera acció que es duu a terme en l'aplicació inicialitzar.do i tot el flux d'accions que es van cridant.
Per exemple : Si volem veure el llistat d'ofertes. Què faria internament l'aplicació? Com s'accediria a la pàgina **llistatOfertes.jsp?**

Primer pas : entrar en el navegador i accedir a l'aplicació (entenem que es troba en el servidor localhost i que tenim el Tomcat en servei)

http://localhost:8080/WimHouse.com

El servidor d'aplicacions llegirà el **Web.xml** que detectarà que estem estenent els servlets d'struts, aquest té com a pàgina principal index.jsp que ens derivarà cap a **inicialitzar.do.**

El que fa aquesta acció es inicialitzar totes les variables de sessió que necessitem, instància el sessionFactory d'hibernate per tal d'utilitzar la connexió amb BBDD .

L'acció inicialitzar.do mitjançant el mètode **Mapping**.**findForward** de l'objecte **ActionMapping** (d'struts) enviarà una constant **SUCCESS** que rebrà el fitxer struts-config.xml. Aquest enviarà una nova petició a l'acció que té en el seu path : **llistatOfertes.do** , per tant ara s'executarà aquest action d'struts. Un cop finalitzat retornarà el control un altre cop a **struts-config.xml** enviant una constant que marcarà la següent petició CANDIDAT, EMPRESA o SUCCESS.

Segons sigui la constant en el fitxer config-struts tenim el **forward** que ens cridarà al **path** corresponent, per exemple 'success' crida a detallOfertes, 'candidat' crida a detallOfertesTilesCandi i 'empresa' crida a **detallOfertesTilesEmp**. Aquests 3 noms es troben com a 'definition names' dintre del fitxer **tiles-defs.xml.**

En aquest fitxer es troba el format de les pantalles que hereten totes de layout.jsp, d'aquesta manera només haurem de canviar el nom del jsp y la seva ubicació en pantalla dependrà del layou.jsp. En el nostre cas si el forward és candidat, empresa i success el .jsp mostrat serà **llistatOfertes.jsp** i el menú dependrà, del forward candidat es crida a **menuCandidat.jsp**, forward empresa es crida a **menuEmpresa.jsp** i forward success es crida a **menus.jsp**

Per tant tots aquests passos són els que es realitzen per arribar a la pantalla **llistatOfertes.jsp**

L'eleccio d'HIBERNATE

Permet treballar amb objectes al desenvolupar sobre les dades de BBDD. L' Sql s'encarrega hibernate de crear-lo. D'aquesta manera continuem treballant amb objectes i abstraient tota la informació.

Els fitxers Hibernates de mapping s'han creat per cada objecte de BBDD, Taules i vistes.
He considerat que d'aquesta manera és molt més fàcil de mostrar la informació ja que accedim directament al objecte i ens oblidem de HSQL o utilitzacions de criteria amb joins , outher joins, que sempre són més complicades i difícils de seguir.

Per tant tindrem objectes per les taules: categories, empreses, especialitzacions, contractes, candidats, ofertes, ofertesCandidats i poblacions. I per les vistes: LlistatOfertes, LlistatCandidats, LlistatOfertesCandidats.

Per cada entitat s'ha creat 1 fitxer hibernate amb extensió .hbm.xml per tal de enllaçar els camps de les entitats de la BBDD amb les propietats del Bean corresponent.

Exemple de Fitxer Hibernate de l'entitat Candidats

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
```

```
<!-- Generated 24-mar-2008 21:46:51 by Hibernate Tools 3.2.0.CR1 -->
<hibernate-mapping>
    <class name="beans.CandidatsBean" table="candidats" catalog="projecte">
        <id name="idCandidat" type="java.lang.Integer">
            <column name="idCandidat" />
            <generator class="identity" />
        </id>
        <property name="nom" type="string">
            <column name="nom" length="45" not-null="true" />
        </property>
        <property name="usuari" type="string">
            <column name="usuari" length="45" not-null="true"/>
        </property>
        <property name="contrasenya" type="string">
            <column name="contrasenya" length="45" not-null="true"/>
        </property>
        <property name="nif" type="string">
            <column name="nif" length="10" not-null="true" />
        </property>
         <property name="dataNaixement" type="java.util.Date">
            <column name="dataNaixement"  />
        </property>
        <property name="adressa" type="string">
            <column name="adressa" length="45" />
        </property>
        <property name="idPoblacio" type="java.lang.Integer">
            <column name="idPoblacio" />
        </property>
        <property name="mail" type="string">
            <column name="mail" length="45" not-null="true"/>
        </property>
        <property name="curriculum" type="java.sql.Blob">
            <column name="curriculum"  />
        </property>

    </class>
</hibernate-mapping>
```

l'elecció d' AJAX

Es necessari dotar a l'aplicació d'un cercador per filtrar la informació. Per tant es decideix filtrar per paraula clau sobre la descripció d'oferta, per categoria, per especialització i contracte.

Com que les tres entitats categoria, contracte i especialització es carreguen de dades prominents de la BBDD haurem de dotar d'un sistema ràpid de carrega sense que hi hagi la necessitat de fer-ho de tota la pàgina, per tant s'escolleix treballar en AJAX. Ajax és ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* com que manté una comunicació asíncrona amb el servidor és poden fer canvis en la mateixa pàgina sense la necessitat de recargarla. Al enviar la petició el servidor carrega les dades de les tres entitats de BBDD y les envia al client mitjançant un fitxer XML, aquest ho detecta i carrega les llistes de valors amb les dades rebudes.

Per tant al no fer la recarrega de tota la pàgina jsp, es guanya en rapidesa de carrega.

l'elecció de MySql

No havia treballat en MySql , si que ho havia fet en Oracle, Sql server y Access. Però vaig elegir des d'un principi treballar en programari lliure, com que Mysql m'oferia aquesta possibilitat vaig optar per ell.

S'han hagut d'instal·lar aplicacions lliures per controlar les connexions i per facilitar la creació de taules , vistes i tractament de dades.

(3) Els diagrames D'UML.

He dividit els diagrames d'UML segons funcionalitats: classes Beans, Model de negoci, DAO per cada entitat, actions, Form's i Utils.

Exemple de diagrama de DAO

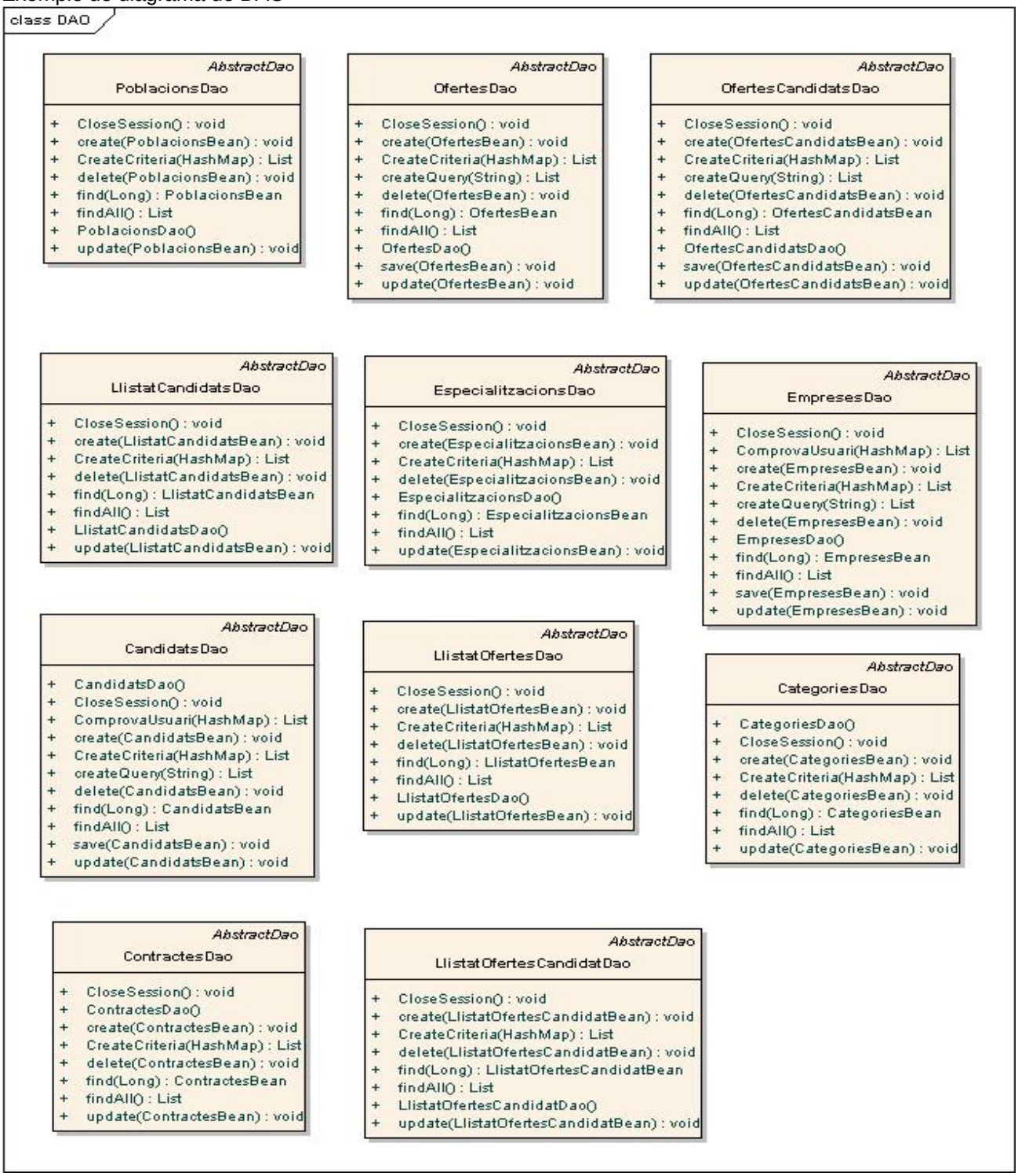


Figura 3.6. Diagrama de Classes DAO.

(4) Exemples de flux de l'aplicació (cas concret).
Explico el comportament en format diagrama del que fa l'aplicació des de la seva crida fins que mostra una pàgina .jsp.

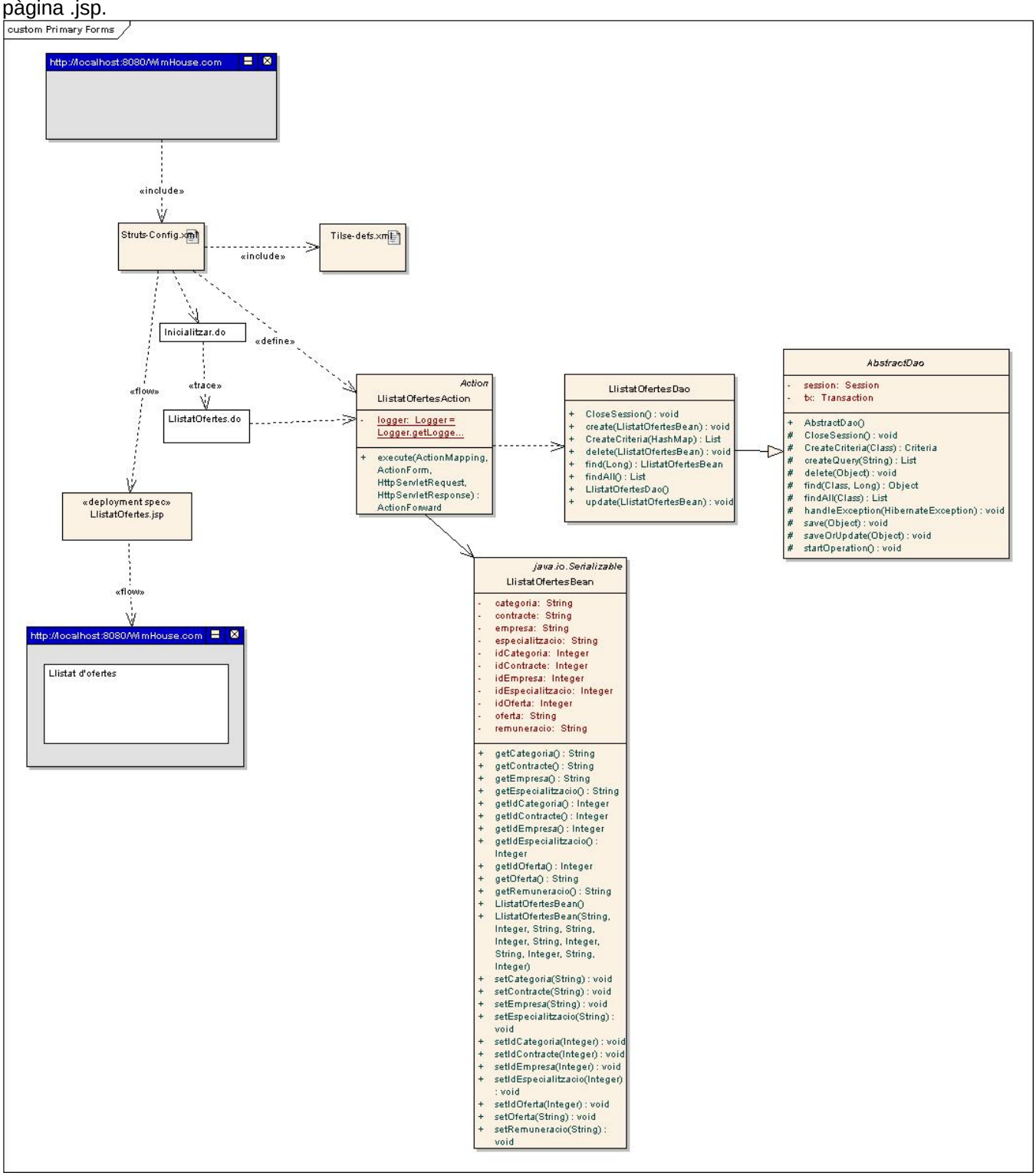


Figura 3.7. Diagrama de Flux

(5) Detall de cada cas d'ús.

Explico el detall tècnic de cada cas d'ús que conté el document d'anàlisi funcional

- Detallat totes les accions que es portaran a terme, quines dades es mostraran i quines dades són d'entrada.
- També quines classes estan implicades en cada cas.
- Quins passos s'haurien de seguir durant la construcció per tal de que es compleixi el MVC, tenint en compte totes les divisions de classes especificades en els diagrames UML i els fitxers hibernate.

Exemple del cas d'us **d'alta d'ofertes** en el DT

Camps d'Entrada Dades
Descripció
Descripció de l'oferta
Camp obligatori
Camp associat a l'atribut de l'ActionForm **descripció**
Màxima longitud de 45 caràcters.
Tipus Contracte
Contracte que ofereix l'empresa
Camp obligatori
S'escull de la classe **ContractesBean**, Es mostren totes les dades de contractes.
Camp associat a l'atribut de l'ActionForm **idContracte**
Per defecte primer contracte ordenat de forma ascendent
Remuneració
Remuneració que s'ofereix en l'oferta
Camp obligatori
Camp associat a l'atribut de l'ActionForm **remuneració**
Màxima longitud de 45 caràcters.
Especialització
Branca laboral a qui va destinada l'oferta
S'escull de la classe **EspecialitzacionsBean**, Es mostren totes les dades d'especialitzacions.
Camp obligatori
Per defecte primera especialització ordenada de forma ascendent
Categoria
Quin lloc de treball s'ofereix
S'escull de la classe **CategoriesBean**, Es mostren totes les dades de categories.
Camp obligatori
Per defecte primera categoria ordenada de forma ascendent
Accions
Boto gravar
Valida informació. Per validar informació es necessari utilitzar **ActionErrors** d'struts.
Assigna identificador a l'oferta i grava informació a la BBDD en l'entitat **Oferta**.
Mostra Missatge de conformitat en cas de que hagi anat bé el procés d'alta.
Mostra missatge de validació o error en cas de que no es compleixin les validacions o en cas de que s'hagi produït un error al gravar.

Botó Cancel·lar
Neteja tots els camps de la pàgina i els deixa amb els valors que tenia inicialment durant la càrrega

Funcionament:

Es cridarà des del menú :
menuEmpresa.jsp
opció **'Alta Oferta'** mitjançant AltaOferta.do cridarem a
AltaOfertaAction.java

Es crea el controlador **AltaOfertaAction** té dos maneres de comportar-se depenent si l'acció es gravar o si l'acció és consultar, aquest comportament el marcarà el paràmetre gravar.
La acció gravar='gravar' només s'executarà quan s'hagi fet click al botó gravar, per tant quan es ve del menú empresa l'acció gravar = '' consultarà les dades.

Es crea una classe **AltaOfertaForm** que hereta d'ActionForm associada al formulari, amb aquesta classe ens serà molt més fàcil consultar i gravar les dades llegint dels mètodes get i set de cada

propietat. Per tant aquesta classe ha de tenir Get's i Set's de tots els camps que es mostren a pantalla tal com informem en la secció Entrada de Dades d'aquest cas d'ús.

La classe **AltaOfertaAction** en el mode *consulta* (paràmetre gravar = '') ha de llegir les dades de la classe AltaOfertaForm i passar-les com a propietats a la pàgina AltaOferta.jsp.

La classe **AltaOfertaAction** en el mode *gravar* (paràmetre gravar = 'gravar') ha de llegir les dades de AltaOferta.jsp mitjançant la classe AlertaOfertaForm associada i passar-les al Bean AltaOfertaBean perquè es pugui enllaçar amb hibernate i gravar a l'entitat Ofertes.

La càrrega de les llistes CategoriesBean, EspecialitzacionsBean i ContractesBean es realitza accedint als Models CategoriesModel, EspecialitzacionsModel i ContractesModel que bolquen les dades sobre els Beans i mitjançant una llista que s'associarà a una colecció de jsp per tal de que es puguin mostrar.

El retorn de l'Action sempre és SUCCESS.

En cas de que hi hagi validacions es posarà dintre del ActionErrors li poden ser els següents errors:
Errors.empresa.required
Errors.descripcioOferta.required
Errors.descripcio.long
Errors.remuneracio.required
Errors.remuneració.long

En el cas de que hi hagi un error al gravar:
Errors.gravar

Els errors es mostraran mitjançant el tag d'struts en la capçalera de la pàgina amb el Tag <html:errors/>

Flux de pantalles

http://localhost:8080/WimHouse.com → Menú Empresa → Alta Oferta → altaOferta.jsp

Diagrama MVC del Cas d'ús:

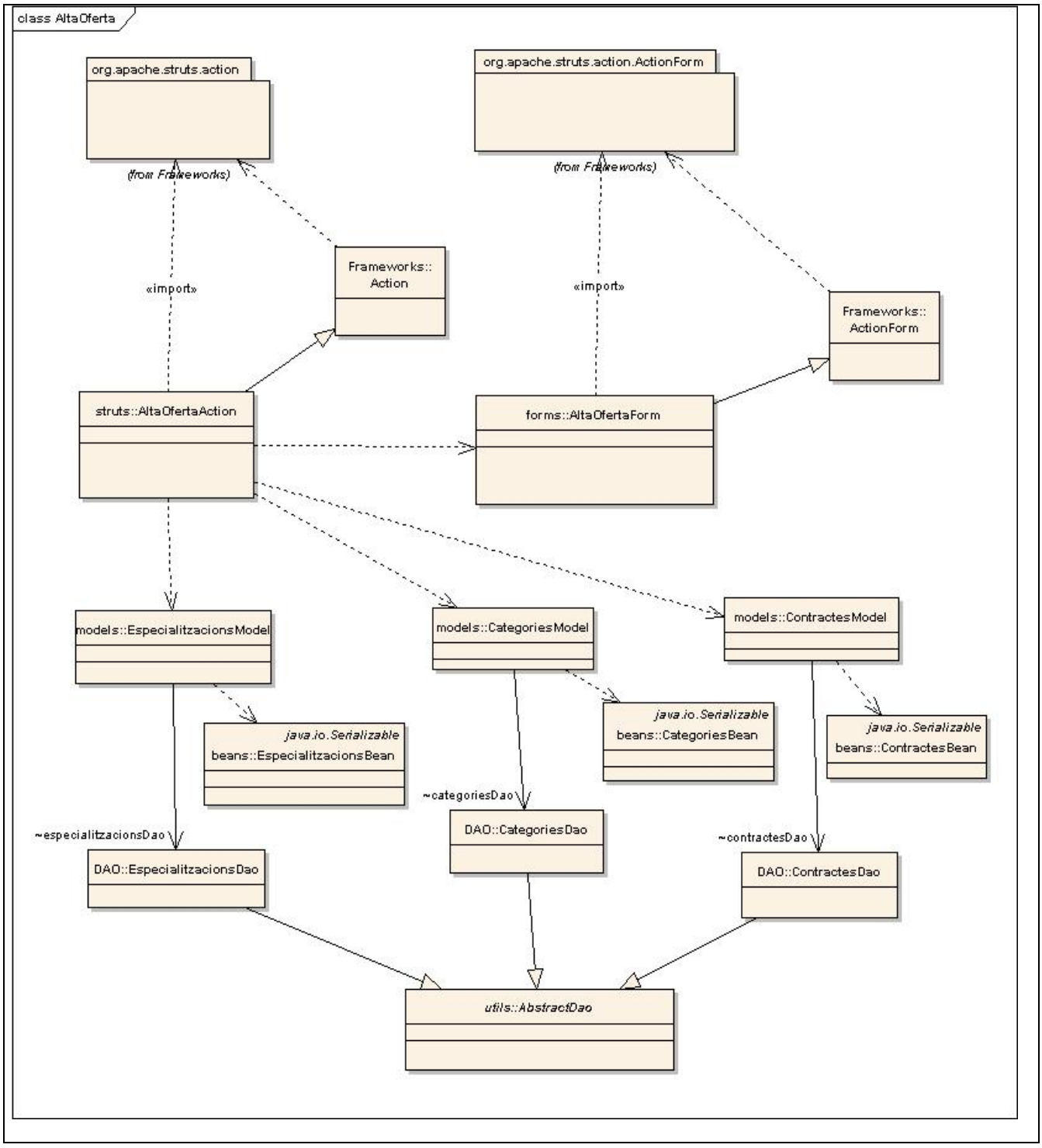


Figura 3.8. Diagrama de Classes. Cas d'ús Alta Oferta.

(6) S' explica el format de la pantalla i les parts de que consta

Es divideix la pantalla en 5 parts: logotip,cercador, zona menús , part principal i enllaços.

Exemple de la part cercador:

Figura 3.9. Diagrama de Format de Pantalla.

(7) Model ER, model de dades

Es detallen totes les entitats de l'aplicació i les vistes es mostra el scripd d'sql

Exemple de l'script de creació de l'entitat empreses

```
CREATE TABLE  projecte.empreses (
  IdEmpresa             int(10) unsigned NOT NULL auto_increment,
  nomEmpresa  varchar(45) default NULL,
  NIF                   varchar(10) NOT NULL,
  descripcio            varchar(100) NOT NULL,
  idPoblacio            int(10) unsigned default NULL,
  mail                  varchar(45) NOT NULL,
  usuari                varchar(10) NOT NULL,
  contrasenya           varchar(10) NOT NULL,
  PRIMARY KEY  USING BTREE (`IdEmpresa`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=latin1
COMMENT='empreses que ofereixen demandes';
```

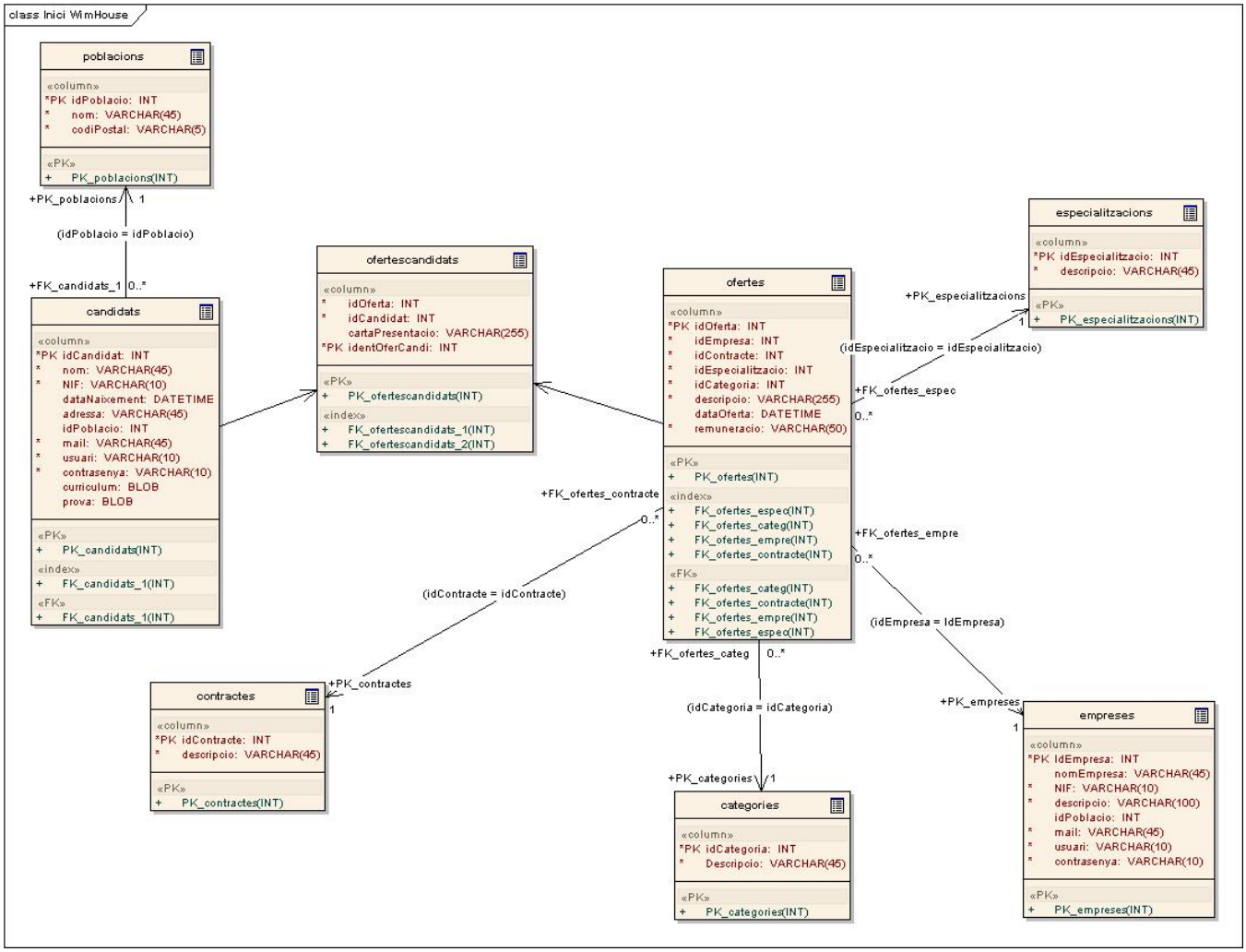


Figura 3.10. Model Entitat Relació

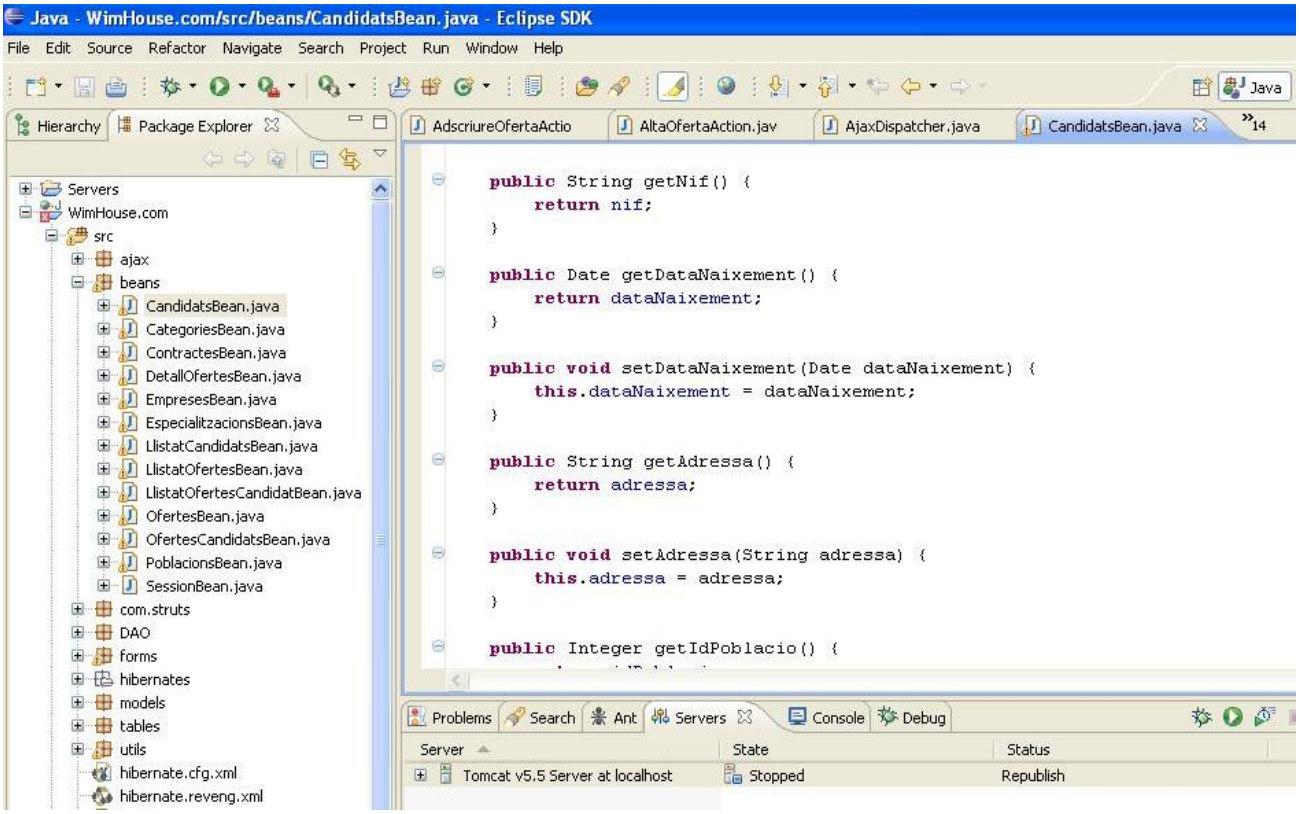
(8) Programari Utilitzat. Quines llibreries externes s'han escollit i perquè.

En el document de disseny tècnic s'explica quines versions de programari i llibreries utilitzades, fent menció a les externes (displayTag) per poder paginar i ordenar les llistes de taules de manera automàtica.

El document de Disseny Tècnic es troba en el fitxer adjunt **DT_jordi_fonoll.doc**

CODIFICACIO

He decidit treballar amb editor Eclipse perquè em permet integrar tota l'arquitectura en un sol programari.



Com es mostra la divisió de classes es per objectes en el directori **src** es troba: beans, dao, actions, forms , fitxers d'hibernate, models de negoci(models) i utilitats (utils).

Les pàgines .jsp es troben en el directori **pages**.

Seguidament passo a descriure un exemple de cada tipus de classe que he hagut de realitzar

Exemple de la classe Bean

Es crea una classe .java que conté com a propietats els camps de l'entitat candidats de BBDD. I com a mètodes els get's i set's de cada propietat. Per tant quan treballem sobre qualsevol entitat de BBDD des de programa el seu contingut sempre el posarem en aquestes classes. Exemple de la classe Bean de Candidats:

```
package beans;

import java.util.Date;
import java.sql.Blob;
// Generated 24-mar-2008 21:46:50 by Hibernate Tools 3.2.0.CR1

/**
 * Empreses generated by hbm2java
 */
public class CandidatsBean implements java.io.Serializable {

     private Integer idCandidat;
```

```java
private String nom;
private String usuari;
private String contrasenya;
private String nif;
private Date dataNaixement;
private String adressa;
private Integer idPoblacio;
private String mail;
private Blob curriculum;

public String getNif() {
	return nif;
}

public Date getDataNaixement() {
	return dataNaixement;
}

public void setDataNaixement(Date dataNaixement) {
	this.dataNaixement = dataNaixement;
}

public String getAdressa() {
	return adressa;
}

public void setAdressa(String adressa) {
	this.adressa = adressa;
}

public Integer getIdPoblacio() {
	return idPoblacio;
}

public void setIdPoblacio(Integer idPoblacio) {
	this.idPoblacio = idPoblacio;
}

public String getMail() {
	return mail;
}

public void setMail(String mail) {
	this.mail = mail;
}

public Blob getCurriculum() {
	return curriculum;
}

public void setCurriculum(Blob curriculum) {
	this.curriculum = curriculum;
}

public void setNif(String nif) {
	this.nif = nif;
}

public CandidatsBean() {
}
```

```
    public CandidatsBean(Integer idCandidat,String nom,String usuari,String
    contrasenya) {
       this.idCandidat = idCandidat;
       this.nom = nom;
       this.usuari = usuari;
       this.contrasenya = contrasenya;
       }

public Integer getIdCandidat() {
       return idCandidat;
}

public void setIdCandidat(Integer idCandidat) {
       this.idCandidat = idCandidat;
}

public String getNom() {
       return nom;
}

public void setNom(String nom) {
       this.nom = nom;
}

public String getUsuari() {
       return usuari;
}

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

public String getContrasenya() {
       return contrasenya;
}

public void setContrasenya(String contrasenya) {
       this.contrasenya = contrasenya;
}
       }
```

Exemple de classe Model

Les classes que s'encarregaran de cridar a les DAO cada entitat, contenint les accions que es duran a terme sobre les taules de BBDD seran les classes model.

Per exemple el següent codi es de la classe model de candidats, en ella es poden veure els mètodes utilitzats per accedir a la classe DAO de candidats:

FindAll -> ens trobaria tots els candidats que hi ha en l'entitat de BBDD (accedint primer al DAO).
Find -> ens retornaria el candidat passat com a paràmetre id.
Insert -> permet insertar un candidat a la BBDD
Update -> permet modificar el candidat passat com a paràmetre
Comprovar Usuari -> ens diu si aquell usuari esta o no registrat com a usuari candidat en la nostra BBDD

CreateQuery -> ens permet crear una sentència SQL dintre del nostre DAO candidats.

Per tant les classes model estaran ja dintre de la nostra capa de negoci però sense tenir cap vinculació amb hibernate, d'aquesta manera s'abstrau qualsevol dependència amb qualsevol eina hibernate/spring...o altres.

```
package models;

/**
 * @author Jordi F
 * Classe de model de Candidats
 */

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import beans.CandidatsBean;
import DAO.CandidatsDao;

public class CandidatsModel {

   CandidatsDao candidatsDao;

   /**
    * @author Jordi F
//  *    Classe de model de negoci de Candidats
    */


   public CandidatsModel(){
         super();
         candidatsDao = new CandidatsDao();

   }


   /**
    * @param pHash
    * @return
    */
   public List ConsultaPerCriteria(HashMap pHash){

         List data = new ArrayList();

         data = candidatsDao.CreateCriteria(pHash);
         return data;
   }

   /**
    * @return
    */
   public List findAll(){

         List data = new ArrayList();

         data = candidatsDao.findAll();
         return data;
   }

   /**
    * @param id
```

```
        * @return
        */

       public CandidatsBean find(Integer id){

             CandidatsBean  mCandidatsBean  = new CandidatsBean();

             mCandidatsBean = candidatsDao.find(new Long(id.longValue()));
             return mCandidatsBean;
       }


     /**
      * @param mCandidatsBean
      */
     public void update(CandidatsBean mCandidatsBean )
     {
       candidatsDao.update(mCandidatsBean);
     }

     /**
      * @param mCandidatsBean
      */
     public void insert(CandidatsBean mCandidatsBean )
     {
       candidatsDao.save(mCandidatsBean);
     }

     /**
      * @param pQuery
      * @return
      */
     public List createQuery(String pQuery)
     {
       return candidatsDao.createQuery(pQuery);
     }

       /**
        * @param pHash
        * @return
        */
       public List ComprovaUsuari(HashMap pHash){

             List data = new ArrayList();

             data = candidatsDao.ComprovaUsuari(pHash);
             return data;
       }

}
```

Exemple de classe DAO

Les classes DAO seran creades per cada entitat model que tinguem i per cada entitat de BBDD, així dons per la entitat de CandidatsModel anterior tindrem la CandidatsDAO. Totes les classes DAO extenen d'una classe AbstractaDAO que té tots els imports d'hibernate necessaris. Així doncs els mètodes que tenim en aquestes classes DAO extenen de la classe DAO abstracta i són els mètodes que proporciona hibernate per consultar i afegir, modificar i esborrar registres de BBDD.

Create –> inserta un registre a la entitat corresponent, en aquest cas a candidats

Find -> busca el candidat passat com a Bean a l'entitat candidats
Delete -> esborra el registre passat com a paràmetre Bean de l'entitat candidats
findAll -> busca tots els candidats de la BBDD
update -> modifica el contingut de candidats de la BBDD del candidat passat com a Bean
createCriteria -> Crea un criteri de selecció de registres.

```
package DAO;

import org.hibernate.Criteria;
import org.hibernate.criterion.Expression;
import utils.AbstractDao;
import utils.DataAccessLayerException;
import beans.CandidatsBean;
import java.util.HashMap;
import java.util.List;


public class CandidatsDao extends AbstractDao {
    public CandidatsDao() {
        super();
    }

    /**
     * Inserta Candidat a la BBDD.
     * @param Candidat
     */
    public void create(CandidatsBean mCandidatsBean ) throws
DataAccessLayerException {
        super.saveOrUpdate(mCandidatsBean);
    }


    /**
     * Esborra Candidat de la BBDD.
     * @param Candidat
     */
    public void delete(CandidatsBean mCandidatsBean) throws
DataAccessLayerException {
        super.delete(mCandidatsBean);
    }

    /**
     * TRoba un candidat a la BBDD.
     * @param id
     * @return
     */
    public CandidatsBean find(Long id) throws DataAccessLayerException {
        return (CandidatsBean) super.find(CandidatsBean.class, id);
    }

    /**
     * Modifica un candidat de la BBDD.
     *
     * @param Candidat
     */
   public void update(CandidatsBean mCandidatsBean) throws
DataAccessLayerException {
        super.saveOrUpdate(mCandidatsBean);
    }
```

```
    public void save(CandidatsBean mCandidatsBean) throws
DataAccessLayerException {
        super.save(mCandidatsBean);
    }

    /**
     * Troba tots els Candidats de la BBDD.
     * @return
     */
    public List findAll() throws DataAccessLayerException{
        return super.findAll(CandidatsBean.class);
    }

    public List CreateCriteria(HashMap pHash) throws DataAccessLayerException{
      List data = null;
      Criteria c = super.CreateCriteria(CandidatsBean.class);

        if (!pHash.isEmpty()){

            if (!pHash.get("idCandidat").equals(new Integer(0))){
                  c.add(Expression.eq("idCandidat",(Integer)
pHash.get("idCandidat")));
            }
            data = c.list();
        }
        else
        {
            data = findAll();
        }

        super.CloseSession();

        return data;
    }

    public List ComprovaUsuari(HashMap pHash) throws DataAccessLayerException{
      List data = null;
      Criteria c = super.CreateCriteria(CandidatsBean.class);

        if (!pHash.isEmpty()){
            c.add(Expression.eq("usuari", pHash.get("usuari")));
            c.add(Expression.eq("contrasenya", pHash.get("contrasenya")));
            c.setMaxResults(1);
            data = c.list();
        }
        else
        {
            data = findAll();
        }

        super.CloseSession();

        return data;
    }


    public void CloseSession() throws DataAccessLayerException{
        super.CloseSession();

    }
```

```
    public List createQuery(String pQuery)
    {
            return super.createQuery(pQuery);
    }
}
```

Exemple de classe Action

Posaré com exemple de classe Action la classe que ens permet gravar les dades de candidats

Primer hem d'importar totes les llibreries que necessitem per dependències de classes

```
  package com.struts;
  import java.util.ArrayList;
  import java.util.HashMap;
  import java.util.Iterator;
  import java.util.List;
  import javax.servlet.http.HttpServletRequest;
  import javax.servlet.http.HttpServletResponse;
  import models.CandidatsModel;
  import models.PoblacionsModel;


import org.apache.log4j.Logger;
import org.apache.struts.action.*;
import org.apache.struts.upload.FormFile;
import org.apache.struts.util.LabelValueBean;
import org.hibernate.*;
import forms.DadesCandidatForm;
import utils.constants;
import beans.CandidatsBean;
import beans.PoblacionsBean;
import beans.SessionBean;


Passem a crear la classe, que extèn de la classe Action d'struts
Import de la llibreria (import org.apache.struts.action.*)
/**
 * @author Jordi F
 * Classe que controla el formulari de dades del candidat
 */
public class DadesCandidatAction extends Action
{
      private static Logger logger =

Paràmetre que ens permet insertar valors en el fitxer de LOG's de log4J. d'aquí
l'import a aquesta llibreria (import org.apache.log4j.Logger;)

Logger.getLogger(DadesCandidatAction.class);

      public ActionForward execute (
                  ActionMapping mapping,
                  ActionForm form,
                  HttpServletRequest request,
                  HttpServletResponse response) {

            logger.info("Entra a DadesCandidatAction - Execute()");

            try{
```

Tal com indico recuperem totes les variables de sessió que necessitarem.
També l'acció si es gravar o consultar. (Si ens han clickat el botó gravar o no)

```
                // primer llegim variables de sessió i averiguem si estem
                // gravant o consultant

                DadesCandidatForm loginForm = (DadesCandidatForm) form;
                String mTipusAccio = request.getParameter("tipusAccio");

                SessionBean mSessionBean = new SessionBean();
                mSessionBean.getSessionVar(request,mSessionBean);
                String mIdCandidat = mSessionBean.getIdCandidat().toString();

                request.getSession().setAttribute("informacio","");
                request.setAttribute("veureLink", "");

                //en cas de que gravem
                if (constants.GRAVAR.equals(mTipusAccio)){
```

En cas de que sigui gravar necessitarem comprovar que tota la informació que ens estan enviant des del client sigui correcta, per això definim els errors que són una variable D'actionErrors. Comprovarem la longitut dels camps i que siguin obligatoris.

```
                        //comprovo errors
                        ActionErrors errors = new ActionErrors();
                        if ((loginForm.getNif() == null) ||
(loginForm.getNif().length() < 1)){
                                errors.add (constants.NIF,
                                new ActionMessage("errors.nif.required"));
                        }
                        if (loginForm.getNif().length() != 8){
                                errors.add (constants.NIF,
                                new ActionMessage("errors.nif.longitud"));
                        }

                        //comprovar nif
                        if ((loginForm.getNom() == null) ||
(loginForm.getNom().length() < 1)){
                                errors.add(constants.NOM,
                                new ActionMessage("errors.nom.required"));
                        }
                        if (loginForm.getNom().length() > 45){
                                errors.add(constants.NOM,
                                new ActionMessage("errors.nom.long"));
                        }

                        if ((loginForm.getMail() == null) ||
(loginForm.getMail().length() < 1)){
                                errors.add(constants.MAIL,
                                new ActionMessage("errors.mail.required"));
                        }
                        if (loginForm.getMail().length() > 45){
                                errors.add(constants.MAIL,
                                new ActionMessage("errors.mail.long"));
                        }
                        if ((loginForm.getUsuari() == null) ||
(loginForm.getUsuari().length() < 1)){
```

```
                              errors.add(constants.USUARI,
                              new ActionMessage("errors.usuari.required"));
                        }
                        if (loginForm.getUsuari().length() > 10){
                              errors.add(constants.USUARI,
                              new ActionMessage("errors.usuari.long"));
                        }
                        if ((loginForm.getContrasenya() == null) ||
(loginForm.getContrasenya().length() < 1)){
                              errors.add(constants.CONTRASENYA,
                              new ActionMessage("errors.contrasenya.required"));
                        }
                        if (loginForm.getContrasenya().length() > 10){
                              errors.add(constants.CONTRASENYA,
                              new ActionMessage("errors.contrasenya.long"));
                        }
                        if (loginForm.getIdPoblacio() == null) {
                              errors.add(constants.DESCRIPCIO,
                              new ActionMessage("errors.poblacio.required"));
                        }
                        if (errors.size() > 0){
                              saveErrors(request, errors);
                              return mapping.findForward(constants.SUCCESS);
                        }

                        try{

                        Comprovarem si existeixen els usuaris a la BBDD, es per
                        saber si hem de fer un alta o una modificació.
                        //comprovar si existeix el candidat a BBDD
                        CandidatsModel mCandidatsModel= new CandidatsModel();

                        String mAltaoModif = constants.MODIF;
                        if    ("0".equals(mIdCandidat)){

                              try{
                                    String SQL_QUERY = "select max(idCandidat)+1
                                    from CandidatsBean";
                                    List list =
                                    mCandidatsModel.createQuery(SQL_QUERY);
                                    if (list.get(0)==null){
                                      mIdCandidat = "1";
                                    }else{
                                      mIdCandidat = list.get(0).toString();
                                    }
                                    mAltaoModif = constants.ALTA;
                              }
                             catch(Exception e){
                                  System.out.println(e.getMessage());
                                }
                        }


                        Comencem a carregar el Bean que gravarem amb les dades
                        que corresponen i tractem la data per gravar-la
                        correctament

                              CandidatsBean mCandidat = new CandidatsBean();
                              mCandidat.setIdCandidat(new Integer(mIdCandidat));
                              mCandidat.setNom(loginForm.getNom());
                              mCandidat.setNif(loginForm.getNif());
```

```
mCandidat.setAdressa(loginForm.getAdressa());
mCandidat.setIdPoblacio(loginForm.getIdPoblacio());

if (!"".equals(loginForm.getDataNaixement())){
      try{
            java.text.SimpleDateFormat sdf =
            new java.text.SimpleDateFormat("dd-MM-
      yyyy"); //Las M en mayúsculas o interpretará
      minutos!!

      mCandidat.setDataNaixement(sdf.parse(loginFor
      m.getDataNaixement()));
      }catch(Throwable t){
            t.printStackTrace();
            errors.add ("data",

            new
            ActionMessage("errors.data.incorrecta")
            );
            saveErrors(request, errors);

            return
            mapping.findForward(constants.SUCCESS);

      }
}
mCandidat.setUsuari(loginForm.getUsuari());

mCandidat.setContrasenya(loginForm.getContrasenya()
);
mCandidat.setMail(loginForm.getMail());

Tractament del currículum primer recuperem la
informació que tenim en el form associat a la jsp
//gravem curriculem en format fitxer
FormFile provaAux = loginForm.getCurriculum();

byte[] fileData = null;
if (provaAux != null){

       fileData = provaAux.getFileData();

mCandidat.setCurriculum(Hibernate.createBlob(fileDa
ta));
}

if (mCandidat.getCurriculum() != null){
      request.setAttribute("veureLink", "Curriculum
      " + mCandidat.getNom());
}

Gravem segons si és alta o modificació
//si es una modificació
if (mAltaoModif == constants.MODIF){
      mCandidatsModel.update(mCandidat);
//si es una alta
}else{
      mCandidatsModel.insert(mCandidat);
      if ("1".equals(mCandidat)){
            String SQL_QUERY = "select
            max(idCandidat) from CandidatsBean";
```

```
                                        List list =
                                        mCandidatsModel.createQuery(SQL_QUERY);

                                    mIdCandidat = list.get(0).toString();
                                }

                            request.getSession().setAttribute("idCandidat",mIdC
                            andidat);


                            request.getSession().setAttribute("candidat",mCandi
                            dat.getUsuari());


                            request.getSession().setAttribute("candidatPwd",mCa
                            ndidat.getContrasenya());


                            request.getSession().setAttribute("nomCandidat",mCa
                            ndidat.getNom());
                            }

                            Informem de que s'ha gravat correctament
                            request.getSession().setAttribute("informacio","S'h
                            an gravat les dades correctament");

                            return mapping.findForward(constants.SUCCESS);
                        }

                        En cas de que hi hagi error retornarem la la classe
                        errors amb contingut i no gravarem
                        //error
                        catch(Exception e){
                               errors.add ("errorgravar",   new
ActionMessage("errors.gravar"));
                               saveErrors(request, errors);
                           System.out.println(e.getMessage());
                           return mapping.findForward(constants.SUCCESS);
                       }

                    }else{

                    En cas de que siguin dades a consultar
                          //LLegir les dades del idCandidat
                    Omplim les llistes de valors de Població.
                          ArrayList mLlistaPoblacions = new ArrayList();
                          List mListaPoblacions = new ArrayList();

                          PoblacionsModel mPoblacionsModel = new PoblacionsModel();
                          mListaPoblacions = mPoblacionsModel.findAll();

                    Iterator it = mListaPoblacions.iterator();
                    while (it.hasNext())
                    {
                          PoblacionsBean vr = (PoblacionsBean) it.next();
                                               mLlistaPoblacions.add(new
                          LabelValueBean(vr.getNom(),vr.getIdPoblacio().toString())
                          );
                    }
```

```
                request.getSession().setAttribute("mLlistaPoblacions",mLlistaPo
                blacions );

                        if    ("0".equals(mIdCandidat)){
                              //ALTA
                              loginForm.setNom("");
                              loginForm.setNif("");
                              loginForm.setDataNaixement("");
                              loginForm.setAdressa("");
                              loginForm.setIdPoblacio(null);
                              loginForm.setMail("");
                              loginForm.setUsuari("");
                              loginForm.setContrasenya("");
                              loginForm.setCurriculum(null);
                        }else{
                        Busquem el candidat a la BBDD i el posem en la variable
                        loginForm, que serà la associada al formulari
                        d'ActionForm d'struts
                        //MODIFICACIO
                        //comprovar si existeix el candidat a BBDD
                        HashMap pHash = new HashMap();
                        pHash.put("idCandidat",new Integer(mIdCandidat));
                        CandidatsModel mCandidatsModel= new CandidatsModel();
                        List users  = mCandidatsModel.ConsultaPerCriteria(pHash);

                        CandidatsBean mCandidatsBean = (CandidatsBean)
users.get(0);
                        loginForm.setNom(mCandidatsBean.getNom());
                        loginForm.setNif(mCandidatsBean.getNif());

                        try{
                              java.text.SimpleDateFormat sdf = new
java.text.SimpleDateFormat("dd-MM-yyyy");

      loginForm.setDataNaixement(sdf.format(mCandidatsBean.getDataNaixement()));
                        }catch(Throwable t){
                              t.printStackTrace();
                        }
                        loginForm.setAdressa(mCandidatsBean.getAdressa());
                        loginForm.setIdPoblacio(mCandidatsBean.getIdPoblacio());

                        loginForm.setMail(mCandidatsBean.getMail());
                        loginForm.setUsuari(mCandidatsBean.getUsuari());

      loginForm.setContrasenya(mCandidatsBean.getContrasenya());

                        if (mCandidatsBean.getCurriculum() != null){
                              request.setAttribute("veureLink", "Curriculum " +
mCandidatsBean.getNom());
                        }

                        request.getSession().setAttribute("idCandidat",
mCandidatsBean.getIdCandidat().toString());


                        }
                        Retornem succes i errors a null en cas de que tot estigui
                        OK
                        return mapping.findForward(constants.SUCCESS);
                }
```

```
            }catch(Exception e){
               Retornem succes i errors diferent de null en cas de que hi hagi un
            error

               //en cas d'error gravem els logs
               logger.error("Causa: " + e.toString());
               logger.error("Causa StackTrace:\n");

               ActionErrors errors = new ActionErrors();
               errors.add("generic", new ActionMessage("errors.generic"));
               saveErrors(request, errors);

                   return mapping.findForward(constants.SUCCESS);
            }
      }


}
```

Exemple de classe Form

Continuem amb l'exemple de candidats i ara mostrarem el Bean del formulari struts associat

Com hem vist en l'exemple anterior, s'utilitza una variable loginForm de tipus `(DadesCandidatForm) form.` Per tant l'action anterior tenia associat una classe Form que és la que ens permet interactuar amb la pàgina jsp, aquesta classe form tindrà una propietat per cada valor de la pàgina jsp que vulguem mostrar o gravar. Aquestes propietats haurien de ser les mateixes que el Bean associat de la classe CandidatBean, per tant la classe hauria de ser

```
   package forms;

   import javax.servlet.http.HttpServletRequest;
   import org.apache.struts.action.ActionForm;
   import org.apache.struts.action.ActionMapping;
   import org.apache.struts.upload.FormFile;


   /**
    * @author Jordi
    * Classe bean que conte les dades que es mostren en
    * el formulari
    */
   public class DadesCandidatForm extends ActionForm{
      private String nom;
      private String nif;
      private String usuari;
      private String contrasenya;
      private String dataNaixement;
      private String adressa;
      private Integer idPoblacio;
      private String mail;
      private FormFile curriculum;

      public String getNif() {
            return nif;
      }
      public void setNif(String nif) {
            this.nif = nif;
      }
      public String getNom() {
            return nom;
```

```
      }
      public void setNom(String nom) {
            this.nom = nom;
      }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
      nom = null;
      nif = null;
    }
      public String getUsuari() {
            return usuari;
      }
      public void setUsuari(String usuari) {
            this.usuari = usuari;
      }
      public String getContrasenya() {
            return contrasenya;
      }
      public void setContrasenya(String contrasenya) {
            this.contrasenya = contrasenya;
      }
      public String getDataNaixement() {
            return dataNaixement;
      }
      public void setDataNaixement(String dataNaixement) {
            this.dataNaixement = dataNaixement;
      }
      public String getAdressa() {
            return adressa;
      }
      public void setAdressa(String adressa) {
            this.adressa = adressa;
      }
      public Integer getIdPoblacio() {
            return idPoblacio;
      }
      public void setIdPoblacio(Integer idPoblacio) {
            this.idPoblacio = idPoblacio;
      }
      public String getMail() {
            return mail;
      }
      public void setMail(String mail) {
            this.mail = mail;
      }
      public FormFile getCurriculum() {
            return curriculum;
      }
      public void setCurriculum(FormFile curriculum) {
            this.curriculum = curriculum;
      }


}
```

He posat aquest exemple perquè es veu que totes les propietats són del mateix tipus que el Bean i per tant que la BBDD menys la dataNaixement que necessita un tractament(que ja hem vist que és l'action el que s'encarrega de fer-ho) i la propietat currículum.

Be en la propietat currículum s'ha hagut de crear un tipus FormFile per llegir-ho de pantalla i carregar-ho directament, però hi ha d'haver un tractament perquè en BBDD es guarda en format BLOB, per tant l'action conté el mètode static proporcionat per hibernate anomenat `Hibernate.createBlob` que permet transformar un fitxer en format BLOB. alhora de gravar-lo.

Però per llegir el currículum s'ha hagut de fer mitjançant una petició de servlet anomenat Download. S'ha extès d' HttpServlet i s'ha llençat mitjançant window.location, amb la particulariat de mostrar una pantalla per adjuntar fitxers: `response.setHeader("Content-disposition", "attachment; filename=Curri`

Per tant el codi que ens permet adjuntar el fitxer de curriculum es trobarà dintre de la classe Download

```
package ajax;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;

import org.apache.log4j.Logger;

import models.CandidatsModel;

import beans.CandidatsBean;

//import org.apache.log4j.Logger;

import java.io.IOException;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.List;


/**
 * @author Jordi F.
 * Ompla els combos del cercador, mitjançant crida ajax
 */
public class DownLoad extends HttpServlet{

    private static final long       serialVersionUID = 2L;

    private static Logger logger = Logger.getLogger(AjaxDispatcher.class);


    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException{
        doGet(request,response);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException{


         try {


               String mIdCandidat =
request.getSession().getAttribute("idCandidat").toString();

               HashMap pHash = new HashMap();
               pHash.put("idCandidat",new Integer(mIdCandidat));
```

```
                    CandidatsModel mCandidatsModel= new CandidatsModel();
                    List users  = mCandidatsModel.ConsultaPerCriteria(pHash);

                    CandidatsBean mCandidatsBean = (CandidatsBean) users.get(0);

                byte[] salida = null;

                salida =  mCandidatsBean.getCurriculum().getBytes(1, (int)
  mCandidatsBean.getCurriculum().length());

                Com podem veure sempre espera aplicació Word per defecte
              OutputStream outServlet = response.getOutputStream();
              response.setContentType("application/msword");

              al posar attachment ens obra una finestra per adjuntar el fitxer
              response.setHeader("Content-disposition",
                                              "attachment; filename=Curri_" +
              mCandidatsBean.getNom() );

              outServlet.write(salida);
              outServlet.close();

          }catch(Throwable t){

              logger.error("Causa: " + t.getCause().getMessage());
              logger.error("Causa StackTrace:\n");

            }

      }

}
```

El Fitxer JSP

Posarem el fitxer d'exemple de dades de candidats per comprovar que les propietats que es mostren en la jsp han de ser les mateixes que tenim en el ActionForm associat(DadesCandidatForm) i com des de javascript es crida al servlet download.

Les llibreries a importar per struts

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

  <link rel="stylesheet" href="estilos.css" TYPE="text/css"/>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>

<script type="text/javascript">
Fem la crida del servlet Download
   function downloadFitxer(idCandidat){
         var url = "/WimHouse.com/DownLoad?idCandidat=" + idCandidat;
         window.location=url;

   }
</script>

L'action executarà l'action d'struts DadesCandidat.do
<html:form action="/DadesCandidat.do"  enctype="multipart/form-data">
   <input type="hidden" name="tipusAccio" value=""/>
   <div id="divBody" style="position:absolute; left: 160px; top: 157px;
width:500px;"
   class="divs2">

   <table class="taula2" >
         <tr>
               En cas de que hi hagin errors s'escriurien aquí
               <html:errors/>
   <bean:write name="informacio"/>
         </tr>
         <tr>
               <th align = left >
                     Zona Candidat
               </th>

         </tr>
         <tr>
               <td align = left >Nom(*)</td>
               <td>
                     Les propietats del form han de tenir el mateix nom que
                     les que vinguin en el jsp
                     <html:text property="nom" size="45"/>

               </td>
         </tr>
         <tr>
               <td align = left >DNI(*)</td>
               <td>
                     <html:text property="nif" size="10"/>
               </td>
         </tr>
         <tr>
               <td align = left >
                     Data Naixement (dd-mm-yyyy)
               </td>
               <td>
                     <html:text property="dataNaixement" size="10"/>
               </td>
         </tr>
         <tr>
```

```
                  <td align = left >
                        Adreça
                  </td>
                  <td>
                        <html:text property="adressa" size="45"/>
                  </td>
            </tr>

            <tr>
                  <td align = left >
                        Població(*)
                  </td>
                  <td>
                        <html:select property="idPoblacio">
                         <html:options collection="mLlistaPoblacions"
property="value" labelProperty="label"/>
                        </html:select>
                  </td>
            </tr>
            <tr>
                  <td align = left >
                        e-mail(*)
                  </td>
                  <td>
                        <html:text property="mail" size="45"/>
                  </td>
            </tr>
            <tr>
                  <td align = left >
                        Usuari(*)
                  </td>
                  <td>
                        <html:text property="usuari" size="10"/>
                  </td>
            </tr>
            <tr>
                  <td align = left >
                        Contrasenya(*)
                  </td>
                  <td>
                        <html:text property="contrasenya" size="10"/>
                  </td>
            </tr>
            <tr>
                  <td align = left >
                        Curriculum(Adjuntar fitxer)
                  </td>
                  <td>
                        El currículum el llegim com un fitxer
                        <html:file property="curriculum" />
                  </td>
            </tr>
            <tr>
                  <td>
                        Només mostrem el l'enllaç en cas de que sigui usuari
                        candidat
                        <a href="javascript: downloadFitxer('<%=
request.getSession().getAttribute("idCandidat") %>')">
                              <bean:write name="veureLink"/>
                        </a>
                  </td>
```

```
            </tr>
            <tr>
                  <td>
                  <button id= "gravar" class="botonEnviar"
                  onclick="javascript:document.forms['dadesCandidatForm'].tipusAc
                  cio.value
                  = 'gravar';document.dadesCandidatForm.submit();"
                  onmouseover="getElementById('gravar').style.cursor='hand';
                  getElementById('gravar').style.color = 'red'; "
                  onmouseout="getElementById('gravar').style.color =
                  '#333333';getElementById('gravar').style.cursor='normal';">
                        Gravar
                  </button>
                  <button id= "cancelar" class="botonEnviar"
                  onclick="javascript:document.forms['dadesCandidatForm'].tipusAc
                  cio.value = 'cancelar';document.dadesCandidatForm.submit();"
                  onmouseover="getElementById('cancelar').style.cursor='hand';
                  getElementById('cancelar').style.color = 'red'; "
                  onmouseout="getElementById('cancelar').style.color =
                  '#333333';getElementById('cancelar').style.cursor='normal';">
                        Cancel.lar
                  </button>
                  </td>
            </tr>

      </table>

</div>
   </html:form>
```

L’AJAX

En la creació d’AJAX es necessita de funcions creades en javascript. Es troben en el fitxer Ajax.js següent:

```
var xmlHttp;

function createXMLHttpRequest() {
    var ajaxsupports = true;
    if (window.XMLHttpRequest) {
          xmlHttp = new XMLHttpRequest();
          if (xmlHttp.overrideMimeType) {
                xmlHttp.overrideMimeType('text/xml');
          }
    } else if (window.ActiveXObject) {
          try {
                xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
          } catch (e) {
                try {
                      xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
                } catch (e) {
                      ajaxsupports=false;
                }
          }
    }

    return ajaxsupports;
}


function sendXMLHttpRequest(url) {
    if (createXMLHttpRequest()) {
          xmlHttp.onreadystatechange = processXMLHttpResponse;
          xmlHttp.open("GET", url, true);
          //Ya s'han enviat les dades en mètode OPEN
          xmlHttp.send(null);

          return (true);
    }
    return (false);
}

function processXMLHttpResponse() {
    //Estado de la conexion 4: Completado
    if (xmlHttp.readyState == 4) {

          //Codi de resposta del servidor al que ens conectem:
          //    404 (not found),500 (internal error), 200(ok),etc..
          if (xmlHttp.status == 200 && xmlHttp.responseText != "") {
                //retornem la resposta en format XML

                callback(xmlHttp.responseXML);
          }
    }
}
```

La crida des de la jsp es fa a la funció:

- SendXMLHttpRequest. Aquest crearà l'objecte XMLhttpRequest i obrirà una petició amb la url passada com a paràmetre. S'executarà el servlet passat com a paràmetre AjaxDispatcher que contindrà un paràmetre 1, 2 o 3.
    - Si es 1 es carregaran les especialitzacions de BBDD,
    - si es un 2 es carregaran les categories
    - si es un 3 els contractes.

- El contingut és retornarà al client mitjançant la funció Callback i en format XML

DISPLAYTAG

Per la paginació i ordenació de les llistes d'ofertes/candidats necessitava d'una llibreria externa. Vaig optar per DisplayTag per la facilitat tant de codi com de instal·lació. Es per això que només insertant les llibreries de DisplayTag en vaig tenir prou i el codi és molt senzill, semblant al iterator d'struts. En poso un exemple. Llistat Ofertes.jsp, en la que les propietats són del Bean de llistatOfertes:

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<%@ taglib uri="http://displaytag.sf.net" prefix="display" %>

  <link rel="stylesheet" href="estilos.css" TYPE="text/css"/>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
<div>
<body onLoad="cridaAjax(1,null);">
<form name="formLlistatOfertes" method="POST" action="../Detall.do">
<input type="hidden" name="idOferta" value="2"/>
<input type="hidden" name="categoria" value=""/>
<input type="hidden" name="contracte" value=""/>
<input type="hidden" name="especialitzacio" value=""/>
<div id="divBody" style="position:absolute; left: 160px; top: 157px;
width:500px;"
  class="divs2">
<table class="taula2"><tr><td class="camp"><%=
request.getSession().getAttribute("titol") %></td></tr></table>
<a class="camp2"><%= request.getSession().getAttribute("nomCandidat") %></a>
<a class="camp2"><%= request.getSession().getAttribute("nomEmpresa") %></a>
<html:errors/>
</br>
<display:table name="sessionScope.testList" pagesize="10" class="taula2"
requestURI="../LlistatOfertes.do" decorator="tables.MyListDecorator">

  Es necessària una modificació de les propietats del displaytag per poder
modificar els missatges per defecte que té displaytag
        <display:setProperty name="basic.msg.empty_list" value="No hi ha
Ofertes"> </display:setProperty>
        <display:setProperty name="paging.banner.items_name" value="ofertes"/>
        <display:setProperty name="paging.banner.one_item_found" >
              <span class="pagebanner">
                    Una oferta trobada.
              </span>
        </display:setProperty>
        <display:setProperty name="paging.banner.all_items_found" >
              <span class="pagebanner">{0} {1} trobades. Mostrant totes les
{2}.</span>
        </display:setProperty>
        <display:setProperty name="paging.banner.some_items_found" >
              <span class="pagebanner">{0} {1} trobades. Mostrant totes les
{2}.</span>
        </display:setProperty>
        <display:setProperty name="paging.banner.first" >
              <span class="pagelinks">
                    [Primer/Previ] {0}
                    [<a href="{3}">Proper</a>
                    /<a href="{4}">Últim</a>]
              </span>
        </display:setProperty>
```

```
            <display:setProperty name="paging.banner.full" >
                  <span class="pagelinks">
[<a href="{1}">Primer</a>
                        /<a href="{2}">Previ</a>]
                        {0}
                        [<a href="{3}">Proper</a>
                        /<a href="{4}">Últim</a>]
                  </span>
            </display:setProperty>
            <display:setProperty name="paging.banner.last" >
                  <span class="pagelinks">
                        [<a href="{1}">Primer</a>
                        /<a href="{2}">Previ</a>]
                        {0} [Proper/Últim]
                  </span>
            </display:setProperty>


Es mostren les columnes amb el mateix nom de les propietats de la taula
      <display:column property="oferta" sortable="true" title="Ofertes de Feina"
href="document.formLlistatOfertes.submit()"/>
      <display:column property="empresa" title="Empresa" />
      <display:column property="contracte"  title="Contracte" />
      <display:column property="remuneracio"  title="Remuneracio" />
      <display:column property="categoria"  title="Categoría" />

</display:table>
</body>
    </div>
```

ALTRES

Vaig generar el .WAR per a la codificació , en ell també incorporo els fitxers .java que formen part del codi font.

Tota la codificació per tant es troba en el fitxer adjunt .WAR **WimHouse.com.war**

PROVES I INSTAL.LACIÓ

Proves

(1) S'ha generat un fitxer de proves per tal de seguir el comportament correcte de l'aplicació tant si les accions es duen a terme com si surt un error inesperat.
(2) S'han creat 2 bases de dades una en blanc, o sigui amb l'estructura però sense dades i l'altra amb dades, per tal de facilitar les proves.

Les proves que s'han realitzat son les següents:

1. Entrar a l'aplicació automàticament mostrarà el Llistat d'Ofertes:

   http://localhost:8080/WimHouse.com hauria de mostrar la pantalla de llistatOfertes.

   Resultat esperat:
   Ens mostrarà el llistat d'ofertes.
   Si no te cap oferta sortirà un missatge dient que no hi ha dades a mostrar.
   Resultat en casos d'error o validacions
   Ens mostrarà un missatge en la part superior de la pantalla.
   S'ha produït un error al llistar les ofertes.

2. Detall d'una Oferta

Des de la pantalla Llistat Ofertes cliquem sobre el link que es mostra en la descripció de l'oferta.
Resultat esperat:
Ens mostrarà el detall de l'oferta

3. Donar d'alta un **Empresa**

Primer pas: entrar a la zona de candidats
En el menú **Zona Empresa** click sobre **Registrar Aquí** (sense informar ni el camp de **Empresa** ni el camp de **Contrasenya**).

Resultat esperat: Ens mostrarà el menú de **Menú Empresa**.

Segon pas: gravar les dades personals
En el **Menú Empresa** click sobre **Dades Empresa.** Entrar totes les dades obligatòries (camps amb *) i click a **gravar**.

Resultat esperat: Ens mostrarà un missatge en la part superior de la pantalla
S'han gravat les dades correctament
Resultat en casos d'error o validacions, ens mostrarà un missatge en la part superior de la pantalla:
S'ha produït un error al gravar

4. Donar d'Alta una Oferta

Des del **Menú Empresa** click sobre **Alta Oferta.**
Entrar totes les dades obligatòries que ens demana (camps amb *) i click botó **gravar**
Resultat esperat: Ens mostrarà un missatge en la part superior de la pantalla
S'han gravat les dades correctament
Resultat en casos d'error o validacions, ens mostrarà un missatge en la part superior de la pantalla*:*
S'ha produït un error al gravar

5. Llistar les Meves ofertes(Menú Empresa)

Des del **Menú Empresa** clickar sobre **Meves Ofertes.**

Resultat esperat: Ens mostrarà una llista de les ofertes que hem donat d'alta amb el nostre usuari Empresa

6. Llistar els candidats **Adscrits a Ofertes**

Des del **Menú Empresa** click sobre **Adscrits a Ofertes.**

Resultat esperat: Ens mostrarà una llista dels candidats que estiguin adscrits a les ofertes donades d'alta des de el nostre usuari Empresa.

7. Donar d'alta un **Candidat**

Primer pas: entrar a la zona de candidats
En el menú **Zona Candidats** click sobre **Registrar Aquí** (sense informar ni el camp de **Candidat** ni el camp de **Contrasenya**).

Resultat esperat: Ens mostrarà el menú de **Zona Candidats**.

Segon pas: gravar les dades personals

En el menú **Zona Candidats** click sobre **Dades Personals.** Entrar totes les dades obligatòries (camps amb *) i click **gravar**.

Resultat esperat: Ens mostrarà un missatge en la part superior de la pantalla
S'han gravat les dades correctament
Resultat en casos d'error o validacions, ens mostrarà un missatge en la part superior de la pantalla:
S'ha produït un error al gravar

8. Adscriure's a Ofertes(Menú Candidat).

Realitzar els passos 1 i 2 d'aquestes proves. Es a dir Llistar una oferta i escollir una oferta mitjançant la descripció.
Es mostrarà el detall de l'oferta.

Click sobre el botó **Adscriure's.** Ens mostrarà la carta de presentació que haurem d'omplir i click sobre botó Acceptar.

Resultat esperat: Ens mostrarà un missatge en la part superior de la pantalla
S'han gravat les dades correctament
Resultat en casos d'error o validacions, ens mostrarà un missatge en la part superior de la pantalla:
S'ha produït un error al gravar

IMPORTANT: NOMES PODEM ADSCIRUE'S A L'OFERTA EN CAS DE QUE ESTIGUEM REGISTRATS COM A USUARIS **CANDIDATS.** EN CAS CONTRARI NO ENS DEIXARÀ L'APLICACIÓ(NO ES VEURÀ EL BOTO ADSCRIURE'S)

9. Meves Ofertes(Menú Candidat).
En el menú de candidat click sobre **Meves Ofertes**

Resultat esperat: Ens mostrarà una llista de les ofertes a la que ens hem adscrit com a usuari Candidat

10. Tanca Sessió(Menú Candidat i Menú Empreses).
Des de **Menú Candidat** o M**enú Empreses** click sobre **Tancar Sessió.**

Resultat esperat: Ens mostrarà el menú per registrar-nos com a usuari o empresa i la llista d'ofertes en la part principal.

11. Registrar-se com a Usuari Candidat o Usuari Empresa
Des de **la Zona Candidat** o **Zona Empreses** del menú principal. Introduir l'usuari/contrasenya i click sobre l'enllaç **Registrar-se aquí**

Resultat esperat: Ens mostrarà el menú Usuari/empresa i La llista d'ofertes com a dades per defecte

Resultat validat: Si usuari/contrasenya es incorrecte, ens mostrarà el menú principal i en la part superior de 'la llista d'ofertes' ens mostrarà un missatge **Usuari registrat no existeix com a Usuari Candidat, si us plau Registreu-vos com a usuari Candidat.**

12. Modificar dades Personals d'Usuari Candidat
Des de Menú principal Registrar-se com a usuari Candidat.
Resultat esperat: Estem registrats i ens mostra el **Menú Candidats**

Accedir a **Dades Personal**, podem modificar totes les dades i gravar.

Resultat esperat: Ens mostrarà un missatge en la part superior de la pantalla
S'han gravat les dades correctament
Resultat en casos d'error o validacions, ens mostrarà un missatge en la part superior de la pantalla:
S'ha produït un error al gravar

13. Modificar dades Empresa d'Usuari Empresa

Des de Menú principal Registrar-se com a usuari Empresa.
Resultat esperat: Estem registrats i ens mostra el **Menú Empreses**

Accedir a **Dades Empresa**, podem modificar totes les dades i gravar.

Resultat esperat: Ens mostrarà un missatge en la part superior de la pantalla
S'han gravat les dades correctament
Resultat en casos d'error o validacions, ens mostrarà un missatge en la part superior de la pantalla:
S'ha produït un error al gravar

Instal.lació:

(1) Per a l'instal.lació del servidor serà necessari tenir un Pentium 4 o superior amb 2 gb deRAM mínim, Navegador FireFox(versio 2) o explorer(6 o superior), Servidor d'aplicacions TOMCAT (versió 5.5), servidor MySQL versió 5.0, j2re1.6.
(2) Per al client : Navegador FireFox(versio 2) o explorer(6 o superior), conexió amb el servidor d'aplicacions, jdbc de mysql amb el datasource corresponent, java versió runtime jre1.6.

El document de proves i instal·lació es troba en el fitxer adjunt **install_jordi_fonoll.doc**

3.1.5 Breu descripció dels altres capítols de la memòria

Consideracions sobre les propostes de modificació enviades pel consultor:

He modificat tant els documents d'anàlisi Funcional com el Disseny Tècnic sobre les consideracions de la PAC2.

- S'ha afegit els diagrames UML en la part de disseny tècnic: Beans, Models, d'actions, flux de l'aplicació, d'hibernate i els d' DAO.
- He explicat cada cas d'ús més detalladament i com hauria de ser el seu comportament amb les classes associades.
- En el model ER he afegit les vistes.

En la PAC3 vaig haver de modificar la part de codificació:

S'ha modificat:

- L'estructura de directoris quedant dividits tal com mostro en la part de codificació.
- S'ha afegit tota la part de DAO, i entitat DAO per cada entitat de BBDD. S'ha creat una classe abstracta DAO.
- S'ha creat un model per cada entitat de BBDD i que està vinculat amb el seu DAO corresponent.
- Els literals de displayTages mostren en català i estan personalitzats.
- S'ha creat un fitxer de LOG's per poder seguir les traçes en cas d'error. Aquest fitxer es troba en la C:\.

- S'ha controlat el tema de les sessions d'hibernate. Ara ja es tanquen correctament. S'ha creat una nova classe HibernateFactory que controla tots aquests aspectes.
- S'han controlat les peticions de sistema no esperades.

3.2 Capítol últim amb les conclusions

Tal com he explicat al principi de la memòria, l'aposta pel projecte era arriscada en tant que no havia treballat ni desenvolupat mai en arquitectura J2EE, però com un repte més, m'ha permès conèixer eines que em serviran molt de cara a la meva professió.

Els conceptes també m'ajudaran per treballar en altres editors, servidors d'aplicacions i llenguatges de programació.

Les eines de diagramació UML m'han servit per entendre la construcció des d'un punt de vista analític tècnic tant d'alt com de baix nivell.

4 Glossari

Les entitats de negoci de l'aplicació:

- Candidats — Usuari registrat que busca ofertes i es pot adscriure a elles.
- Empreses — Usuari registrat que pot donar d'alta ofertes.
- Ofertes — Oferta laboral que s'ha donat d'alta.
- Contractes — Contracte vinculat a l'oferta.
- Categories — Categoria a qui va destinada l'oferta.
- Especialitzacions — Especialització a qui va destinada l'oferta.

Altres:

- DisplayTag — Llibreria d'open Source per Tags de presentació en arquitectures MVC.
- BBDD — Servidor de Base de dades.
- MySql — Sistema de gestió de Base de dades relacional i multiusuari.
- J2ee — Plataforma de programació—part de la plataforma Java – per desenvolupar i executar software d'aplicacions en Llenguatge de programació Java amb arquitectura de N nivells distribuïda.
- AJAX — Asynchronous JavaScript And XML.
- FrameWork — És una estructura de suport definida en la qual un altre projecte de programari pot ser organitzat y desenvolupat.
- Bean — Component de programari reutilitzable .
- DAO — Patró d'accés a dades.
- Struts — Eina de suport per al desenvolupament d'aplicacions Web sota el patró MVC
- MVC — Patró d'arquitectura de software que separa les capes d'interfície, dades i lògica de negoci.
- UML — Llenguatge unificat de modelatge.
- Hibernate — Eina de mapeig objecte-relacional que facilita el mapeig d'atributs entre una Base de dades relacional tradicional i el model d'objectes d'una aplicació.
- Model ER — Eina de modelatge de dades d'un sistema d'informació.

5 Bibliografia

www.wikipedia.com

www.google.com

www.struts.apache.org

www.tomcat.apache.org

www.Displaytag.sourceforge.net

www.hibernate.org

www.eclipse.org

www.java.sun.com

6 Annexos

Envio els següents documents

Document d'anàlisis funcional: AF_Jordi_Fonoll.doc

Document de Disseny Tècnic DT_JordiFonoll.doc

Codificació: WimHouse.com.war

Document d'instal·lació i proves: install_Jordi_Fonoll.doc