TFC – Aplicació web per gestió de tornejos d'escacs amb J2EE

Alumne: *Xavier Benet i Saniger*
Enginyeria Tècnica en Informàtica de Gestió

Consultor: *Jose Juan Rodriguez*
Gener 2011

Aquest treball està subjecte -excepte que s'indiqui el contrari- en una llicència de Reconeixement – No Comercial – Sense Obra Derivada 2.5 Espanya de Creative Commons. Podeu copiar-lo, distribuir-los i transmetre'ls públicament sempre que citeu l'autor i l'obra, no es faci un ús comercial i no es faci còpia derivada. La llicència completa es pot consultar en http://creativecommons.org/licenses/by-nc-nd/2.5/es/deed.es.

Dedicatòria i agraïments

"Aprendre és caminar cap als nostres límits i amb aquest pas constant els anem empenyent, ampliant així el radi d'acció del nostre coneixement"

Això és el que he experimentat en alguns moments d'aquesta etapa acadèmica i crec que la qüestió és aprendre a saber trobar la manera d'anar avançant de forma que els propis límits poc a poc es vagin allunant del seu punt inicial.

Després de gairebé sis anys immers en aquests estudis, finalment ha arribat el moment de realitzar el Treball fi de carrera. El recorregut fins aquí no ha estat senzill i en alguns moments hauria tirat la tovallola, però amb prudència, paciència i perseverança he arribat fins aquí i ara toca fer la recta final fins a l'arribada d'aquesta cursa d'ultrafons.

Principalment, dedico aquest treball a la Marta, que en bona part és la culpable que em llencés a iniciar aquests estudis, i durant tots aquests anys ha estat al meu costat incondicionalment, animant-me en els moments més complicats i també compartint els meus èxits.

També el vull dedicar a tota la família i especialment a la Núria, en Xevi i la petita Maria per la paciència i l'interés que han anat mostrant tots aquests anys vers els meus estudis. També va per als que ja no són entre nosaltres, a l'avi Pepe, a l'àvia Jesusa i a l'avi Josep.

Durant aquest període a la UOC, també he tingut la sort de topar-me amb bons companys de batalla amb els que he hagut de fer alguna assignatura en grup. Dels que en guardo un bon record són: ***Angel Torrecillas, Sergi Martínez Millet, Alex Ullate, Oriol Bonany, Manu Prieto, Israel Díaz, Raul Garin, Xavier Bastida i Marc Garcia*.**

Resum

J2EE *(Java to Enterpise Edition)*, és la tecnologia actual per desenvolupar aplicacions web amb llenguatge Java, i que ha anat evolucionant tant durant els darrers anys, que actualment existeixen un gran ventall de *Frameworks* de desenvolupament ràpid que el converteixen en un paradigma molt potent.

En aquest cas, l'aplicació s'instal·la a un servidor *GlassFish*, contra una base de dades *MySql*, i s'ha realitzat amb els *Framworks Struts2* i *Hibernate*. Tot això configurat per poder treballar amb l'entorn de programació *NetBeans*.

Una part complicada i llarga ha estat la d'escollir les tecnologies amb les que treballar, bàsicament pel fet de no conèixer a priori res sobre J2EE. Per això, haver escollit aquesta àrea per fer el TFC, ha comportat un risc important, però tanmateix m'ha permès aprendre els procediments a seguir per construir una aplicació web empresarial, amb una connexió a una base de dades, amb aquesta tecnologia tan potent. La fase de desenvolupament també ha estat molt complicada, com a conseqüència de la manca d'exemples reals a la xarxa sobre els *Framworks* finalment escollits, però en qualsevol cas, la investigació i cerca de documentació també m'ha ajudat a formar-me una visió general de les tecnologies que existeixen, i quina tria en faria si hagués de tornar a fer aquest TFC, o si més endavant em toca construir una aplicació web amb J2EE.

L'aplicació consisteix en un programa web que des d'un navegador ens permet realitzar una sèrie de consultes i de gestions contra una base de dades de tornejos d'escacs. Les gestions i consultes variaran segons el rol d'usuari sigui administrador, gestor o usuari Internet.

Així, l'aplicació consta d'un servidor, una base de dades, unes classes entitat Java que treballaran a la capa de model per fer els canvis a la Base de dades, unes pàgines *html* i *jsp*, que estaran a la capa de vista, uns fitxers de configuració *.xml* i uns fitxers *.java* que actuaran a la capa de control. Els rols s'han implantat a nivell de servidor, amb una pàgina de *login*, i segons el rol escollit, es tindrà accés a una carpeta (zona) o altra.

Índex de continguts

1. Capítol 1: Introducció 8
   a. Justificació 9
   b. Objectius 10
   c. Enfocament i mètode seguit 12
   d. Planificació i temporització 12
   e. Productes obtinguts 14
   f. Breu descripció dels següents capítols 14
2. Capítol 2: Anàlisi i Disseny de l'aplicació 15
   a. Casos d'ús 15
   b. Diagrama de classes entitat 16
   c. Diagrama d'estats per torneig i ronda 17
   d. Esquema funcional 18
   e. Diagrama Entitat-Relació ER 18
   f. SQL per MySql 19
3. Capítol 3: Aprenentatge i tria de les tecnologies 21
4. Capítol 4: Disseny de l'arquitectura 22
5. Capítol 5: Preparació de l'entorn 24
6. Capítol 6: Fase de Desenvolupament 27
   6.1. Preparació del Servidor 27
      a) Sistema gestor de Bases de Dades *MySql 5.1.53* 27
      b) Base de J2EE i *JRE* i Servidor *GlassFish 3.0.1* 31
   6.2. Desenvolupament de l'aplicació 32
      a) Codi base de configuració i Entity Beans 32
      b) Afegint rols a l'aplicació 38
      c) Desplegament de l'aplicació 44
7. Capítol 7: Conclusions 45
8. Glossari 46
9. Bibliografia 49

Índex de figures

Figura 1 – Temporització del projecte 13

Figura 1 – Diagrama de GANTT 13

Figura 3 – Diagrama de Casos d'ús 15

Figura 4 – Diagrama UML de Classes 16

Figura 5 – Diagrama d'Estats 17

Figura 6 – Diagrama Funcional 18

Figura 7 – Diagrama Entitat Relació 18

Figura 8 – Esquema del model MVC 22

Figura 9 – Taules de la Base de Dades 24

Figura 10 – Nou Projecte amb NBeans 25

Figura 11 – Tria de Frameworks i connexió 25

Figura 12 – Descàrrega del J2EE SDK + GlassFish 31

Figura 13 – Pantalla inicial 37

Figura 14 – Pantalla Llistat Torneigs 38

Figura 15 – index.jsp 40

Figura 16 – Pantalla d'autenticació 41

Figura 17 – Pantalla 1 usuari Internet 41

Figura 18 – Pantalla 2 usuari Internet 42

Figura 19 – Pantalla 1 usuari Gestor 42

Figura 20 – Pantalla 2 usuari Gestor 42

Figura 21 – Pantalla 1 usuari Administrador 43

Figura 22 – Pantalla 2 usuari Administrador 43

Figura 23 – Creació d'usuaris a GlassFish 44

Capítol 1: **Introducció**

El Treball Final de Carrera és possiblement el moment més important durant tota la carrera d'un estudiant d'Enginyeria Tècnica, ja que aquí és on es tanca el cercle de tot el que s'ha anat aprenent durant aquests estudis, i a més encara cal d'aprendre i assimilar conceptes nous, que ens permetran finalment culminar en el desenvolupament d'una aplicació real des del principi amb la planificació i el disseny, fins a fer-ne el desenvolupament, implantació i desplegament.

Aquest document serà la memòria per el Treball Final de Carrera, que per una banda servirà com a dietari de tot el procés de realització i per l'altra, servirà per a especificar les passes de planificació, disseny i implementació que s'han anat seguint fins a arribar a l'obtenció del producte final.

D'aquesta manera s'aniran enregistrant totes les decisions preses durant el procés i els problemes amb què s'haurà anat topant i com s'han pogut superar.

a) Justificació

L'experiència adquirida en l'àmbit de la programació, ve de l'àmbit acadèmic durant el transcurs d'aquests anys d'estudi d'Enginyeria Tècnica en Informàtica de Gestió, essent nul·la en l'àmbit professional. Si bé als primeres cursos de programació, es va tocar el llenguatge C i l'Assemblador, des de que es va entrar en el paradigma de la POO (Programació Orientada a l'Objecte), el llenguatge sempre ha estat el mateix, el JAVA. Per ser aquest un llenguatge relativament nou, però amb una forta consolidació, i amb les infinites possibilitats que ens brinda, tan per la seva elasticitat, flexibilitat, portabilitat i la seva gran capacitat per desenvolupar aplicacions empresarials de grans dimensions, s'ha optat per escollir fer el Treball Final de Carrera amb la tecnologia J2EE (*Java to Enterprise Edition).*

Aquesta tecnologia és la continuació de la J2SE (*Java to Standard Edition)*, ja que si bé J2SE va enfocada a les aplicacions d'escriptori, J2EE està enfocada a les aplicacions web, i no cal dir la importància que això té en el món actual, on tantes són les empreses que encara s'han d'adaptar al 2.0 i tantes són les possibilitats que ens ofereix aquest paradigma de programació.

La tecnologia escollida no és la més senzilla ni planera, però vaig creure oportú seleccionar el J2EE per fer el meu treball, aprofitant la meva situació personal, que a priori em permetria disposar de prou temps per dedicar-me a l'aprenentatge de l'arquitectura del model a escollir dins de J2EE, i d'aquesta manera, a part d'aprendre a programar aplicacions web també n'obtindria l'aprenentatge de la preparació de tot l'entorn del servidor i de desenvolupament. A més, aquest és un paradigma de programari lliure, obert i multi plataforma, que ens permet construir aplicacions web, sense tenir en compte l'arquitectura del sistema operatiu al que anirà destinat, gràcies a la màquina virtual de *java* i a l'execució des de qualsevol navegador web.

El tema a escollir per a la realització del treball és lliure, i primerament es va haver de pensar en un tema que es pogués desenvolupar amb la tecnologia J2EE (*Java to Enterprise Edition*). De fet, essent un llenguatge orientat a l'objecte, imaginant alguna aplicació útil, s'haurien de poder aplicar les metodologies de disseny apreses a les assignatures: Enginyeria del Programari, Bases de Dades i Tècniques de Desenvolupament de Programari.

Per tant, es va escollir un tema conegut per mi, emmarcat dins el món dels escacs. Amb el poc temps disponible del semestre per a realitzar l'aplicació, es va haver de pensar en quelcom que pogués ser pràctic en el món real, i que fos assumible quant a volum de feina. Finalment es va pensar en una aplicació de gestió de tornejos d'escacs, que servís perquè els àrbitres puguin gestionar els aparellaments de les rondes dels tornejos que tinguin assignats. A més, l'aplicació podrà ser visitada per qualsevol usuari d'Internet, per consultar els aparellaments i resultats de cada ronda dels torneigs que figurin en el sistema.

Aquest tema, crec que s'adequa perfectament a l'àmbit i al que es demana, atès que per una banda hem de dissenyar una Base de Dades on anar desant totes les dades, i per l'altra, es tracta de dissenyar i desenvolupar el funcionament d'una aplicació orientada a l'objecte, que estigui disponible des de qualsevol navegador a Internet.

b) Objectius

Per aconseguir tenir clares les funcionalitats que ha de tenir l'aplicació, s'han visitat diferents webs de federacions d'escacs i s'ha llegit documentació sobre les normes en els torneig d'escacs. A més, s'ha utilitzat la pròpia experiència per imaginar i completar el disseny del funcionament general de l'aplicació.

El projecte ha d'implementar una aplicació web que permeti gestionar tornejos d'escacs. A Catalunya se celebren molts tornejos durant tot l'any i especialment a l'estiu és quan se'n concentren més. Molt clubs d'escacs utilitzen programes comercials per tal de gestionar tot el referent al torneig que han organitzat, però el problema és que per poder penjar els resultats de cada ronda a la seva pàgina web, han de transformar les dades de sortida del seu programa, a format web, amb la conseqüent pèrdua de temps que això comporta.

Actualment, tots volem tenir la informació disponible el més aviat possible i Internet ens ofereix aquest avantatge. Si poguéssim fer una aplicació dinàmica que des d'un navegador poguéssim fer totes les funcionalitats d'una aplicació de gestió de torneigs, la tasca del gestor del torneig seria força més simple. Així ell es dedicaria a generar un nou torneig, anar entrant els resultats per cada ronda, generar una nova ronda i tancament del torneig. Per altra banda, tots els jugadors podrien anar veient l'estat de cada ronda, les classificacions i el contrincant que els hi ha tocat per a la següent ronda. El paper de gestor d'un torneig queda reservat als àrbitres, però qualsevol usuari que es doni d'alta com a gestor, podrà donar d'alta un torneig i gestionar-lo.

Per això, hi haurà tres tipus d'usuaris de l'aplicació, amb tres rols diferenciats. Uns seran els administradors que podran crear usuaris gestors i jugadors, a més de totes les funcionalitats dels usuaris gestors. Després tindrem els usuaris gestors, que seran els que podran donar d'alta torneigs i entrar resultats, generar rondes i fer els tancaments de torneigs. L'altre tipus d'usuari serà qualsevol usuari d'Internet que vulgui fer consultes al web.

Els dos objectius bàsics són: en primer lloc, que qualsevol usuari d'Internet pugui veure els resultats dels torneigs que hi haurà a la web, i seguir dia a dia el desenvolupament dels que estiguin en joc. Així si el visitant és un participant d'algun d'aquests torneigs, podrà fer un seguiment dels resultats, aparellaments i classificacions de cada ronda, des de qualsevol punt amb connexió a Internet. El segon objectiu és que un àrbitre o usuari gestor es pugui donar d'alta per esdevenir gestor d'un torneig, i pugui crear i gestionar tornejos. Per aconseguir ser gestor, haurà e contactar amb l'administrador i aquest el donarà d'alta al servidor de l'aplicació i també a la base de dades mitjançant un formulari web.

A partir d'aquests dos objectius bàsics, a continuació es desgranaran el que serien els objectius més específics.

Quan es genera un nou torneig, l'aplicació demanarà les dades necessàries per tal de poder-lo crear correctament. Aquestes dades s'hauran d'entrar en un formulari dins d'un navegador d'Internet, i aniran des del nom del torneig, el nombre de rondes que es disputaran, data d'inici i data de finalització del torneig, fins al llistat de jugadors participants en el torneig.

De cada jugador volem saber el nom, l'ELO (força de joc del jugador que es calcula amb un algorisme segons va guanyant, perdent o empatant partides durant un cert període de temps i tenint en compte l'ELO dels seus rivals), club, edat, punts que porta actualment en el torneig...

Després hi haurà les rondes on a dins de cadascuna hi haurà les partides a disputar, que es farà amb un sorteig seguint el sistema suís. Aquest sorteig consisteix en dividir el llistat dels jugadors ordenats per puntuació i ELO i es fa jugar el 1r de la primera meitat amb el 1r de la primera meitat, el 2n amb el 2n, i així fins a arribar al final de la llista. Si el nombre de jugadors inicial és imparell el primer de la llista descansarà a la primera ronda i se li donarà un punt (Bye). En les següents rondes se li donarà el descans al jugador amb un ELO més alt del grup amb menys puntuació, entenent com a puntuació els punt que porta guanyats en el present torneig.

A la primera ronda, els jugadors imparells de la primera meitat de la llista jugaran amb blanques i els imparells d'aquesta primera meitat ho faran amb negres. Per exemple si tenim 40 jugadors a un torneig, els aparellaments quedarien així:

(1 – 21, 22 – 2, 3 – 23, 24 – 4, ..., 40 – 20)

Si tinguéssim 41 jugadors, el primer descansaria a la primera ronda i el segon passaria a ocupar el lloc del 1r. A partir de la segona ronda, el primer passaria a ocupar el seu lloc. Dos jugadors es poden enfrontar només una vegada en un mateix torneig.

Cada partida tindrà dos jugadors i un resultat, victòria del 1r jugador, taules o victòria del 2n jugador, així com la data, i alguna altra dada menys important. Quan un jugador guanyi una partida, se li sumarà un punt, en cas d'empat (taules) se li afegirà mig punt, i en cas de derrota, cero punts. El primer jugador jugarà amb blanques i el segon ho farà amb negres. En principi, cada jugador farà una partida amb blanques i a la següent ronda amb negres. Així cada jugador haurà de jugar amb un nombre de partides amb cada color de peces de forma equilibrada, sempre i quan això sigui possible.

Al finalitzar una ronda l'usuari creador del torneig entrarà els resultats i seguidament generarà el sorteig per la següent ronda. Qualsevol usuari que entri a la web del torneig ha de poder consultar com ha anat cada ronda. També ha de poder consultar el sorteig per a la següent ronda, la classificació a cada ronda, o la classificació final, que serà la classificació després de la darrera ronda.

c) Enfocament i mètode seguit

És bàsic fer una bona planificació de totes les tasques que caldrà anar fent durant el transcurs del semestre, incloses les fases d'aprenentatge i cerca de documentació, proves, etc. Abans però, caldrà seguir les fases habituals de creació d'una aplicació, anàlisi i disseny, i per acabar amb el desenvolupament de i finalment ambles proves de l'aplicació en una màquina des de zero, fent totes les instal·lacions i desplegaments necessaris.

d) Planificació i temporització

Una vegada que s'ha definit l'àmbit, els objectius i les funcionalitats del projecte, toca fer-ne la planificació i temporització, tenint en compte el temps disponible per realitzar-ne tot el procés.

Per tal de dur a terme el desenvolupament de l'aplicació i poder assolir els objectius marcats, en aquest apartat es farà un llistat de les tasques a realitzar i de la temporització de cadascuna d'elles.

- Diagrama de casos d'ús
- Diagrama de classes
- Diagrama de seqüència
- Disseny gràfic de les pantalles
- Aprenentatge Tecnologia J2EE
- Dissenyar esquema E/R
- Crear Base de Dades
- Disseny de l'arquitectura J2EE
- Disseny de la pantalla de formulari d'alta d'usuaris gestors
- Disseny de la pantalla de gestió de tornejos
- Disseny de la pantalla de gestió de jugadors
- Disseny de la pantalla de visualització d'aparellaments i resultats
- Programar les classes entitat
- Programar els fitxers de configuració
- Programar les pàgines de capa vista
- Programar el funcionament dels rols
- Retocar funcionaments generals

Temporització amb dates:

	Inici	Fi	Duració (setmanes)
Fase de Disseny	04/10/2010	11/11/2010	5.5
Diagrama casos ús	04/10/2010	10/10/2010	1
Diagrama classes	04/10/2010	10/10/2010	1
Diagrama seqüència	04/10/2010	10/10/2010	1
E/R i SQL	11/10/2010	17/10/2010	1
Disseny de pantalles	11/10/2010	17/10/2010	1
Aprenentatge tecnològic	11/10/2010	11/11/2010	4.5
Disseny Arquitectura	18/10/2010	11/11/2010	3.5
Prova cas ús bàsic	25/10/2010	11/11/2010	2.5
F. Desenvolupament	12/11/2010	15/01/2011	9
Gestió usuaris	12/11/2010	21/11/2010	1.2
Gestió tornejos	22/11/2010	30/12/2010	5
Gestió jugadors	29/11/2010	04/12/2010	1
Gestió ronda	06/12/2010	11/12/2010	1
Vistes rondes	27/12/2010	01/01/2011	1
Navegació i Rols	12/11/2010	15/01/2011	10.5
Retocs generals	11/112010	15/01/2011	9
Memòria i presentació	04/10/2010	15/01/2011	15

Figura 1 - Temporització del projecte

Diagrama de Gantt

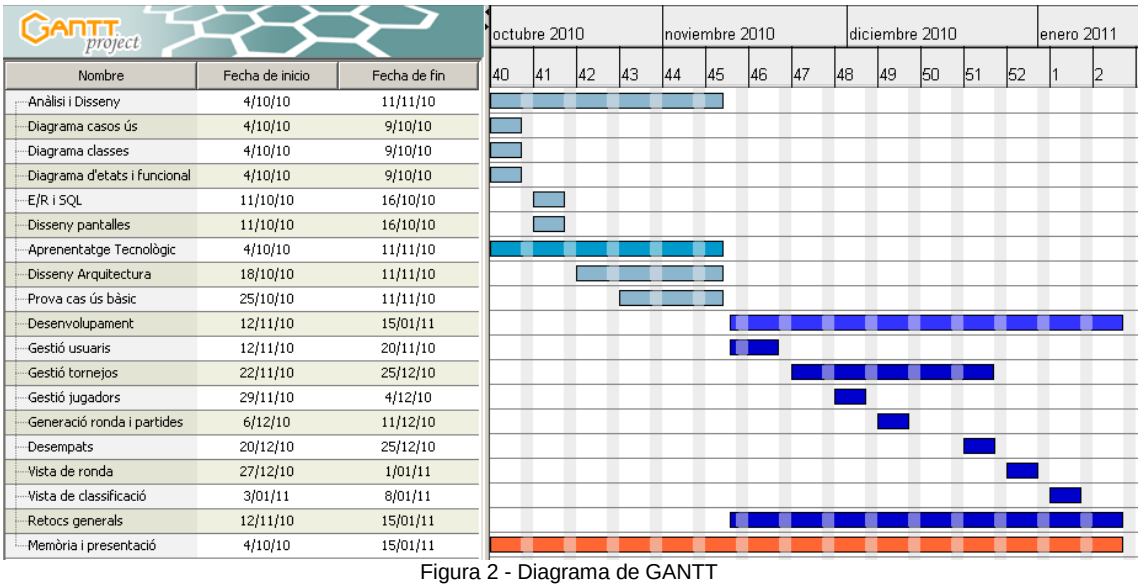

Figura 2 - Diagrama de GANTT

e) Productes obtinguts

Al finalitzar aquest Treball Final de Carrera, haurem d'obtenir els productes següents:

- Una jerarquia de carpetes amb tots els fitxers de l'aplicació inclosos, i el fitxers .war i *.ear* per poder fer el desplegament a qualsevol altra màquina. La jerarquia de carpetes estarà formada per les classes, un fitxer de creació de la Base de Dades i inserció de registres de prova, els fitxers .html i .jsp de visualització i la resta de fitxers necessaris pel funcionament.
- Una memòria (aquest document) on hi tindrem l'explicació de tot el procés de creació i desenvolupament de l'aplicació, així com les instruccions d'instal·lació i funcionament,
- Una presentació que sintetitzi aquest memòria

f) Breu descripció dels següents apartats

En el **capítol 2** hi ha tot l'anàlisi i disseny de l'aplicació, des dels casos d'ús, passant pel diagrama de classes, l'esquema Entitat-Relació de la Base de Dades i el codi SQL de creació, fins al disseny de la vista de les pantalles. És una fase molt important, que a vegades cal anar modificant a mesura que estem avançant ja en el desenvolupament, atès que quan realment estem codificant és quan se'ns obren nous dubtes que no havíem previst quan fèiem l'anàlisi. Per això, pot ser que aparegui alguna petita modificació, respecte a l'entrega de la feina feta en aquesta fase.

El **capítol 3** és on es descriu en què es basa per escollir unes tecnologies i no unes altres, i quin procés se segueix per aprendre-les.

En J2EE, existeixen una sèrie de patrons arquitectònics que segons el resultat buscat, n'hi haurà alguns que encaixaran millor que d'altres. En aquest cas hem escollit el model vista controlador MVC. En el **capítol 4**, es farà una descripció d'aquest model i com es fusionarà amb les tecnologies escollides i amb les peces de l'aplicació.

La preparació de l'entorn també és una tasca amb un pes important, ja que comporta les seves complexitats i característiques segons la tria que haguem fet de les tecnologies escollides. Això ho veurem en detall al **capítol 5**.

El **capítol 6** fa referència a tot el desenvolupament de l'aplicació, amb la descripció dels diferents fitxers de l' aplicació, la creació de la base de dades, les pantalles i el desenvolupament dels rols a nivell de servidor i de fitxers. Aquí també s'hi descriu com efectuar el desplegament a un equip des de zero, a partir dels fitxers *.war* i *ear*.

Els capítols restants fan referència a les conclusions, glossari, bibliografia i annexos.

Capítol 2: **Anàlisi i disseny**

El disseny de l'aplicació s'ha fet seguint les tècniques apreses a altres assignatures de la carrera i utilitzant diferents eines de disseny que hi ha al mercat. El disseny dels diagrames de casos d'ús, d'estats i de classes s'ha realitzat amb l'aplicació de disseny *Power Designer 12*, pel diagrama de persistència s'ha utilitzat el programa de disseny gràfic *Macromedia Fireworks MX*, pel diagrama funcional, s'ha utilitzat el *Office Word*

a) Diagrama de casos d'ús

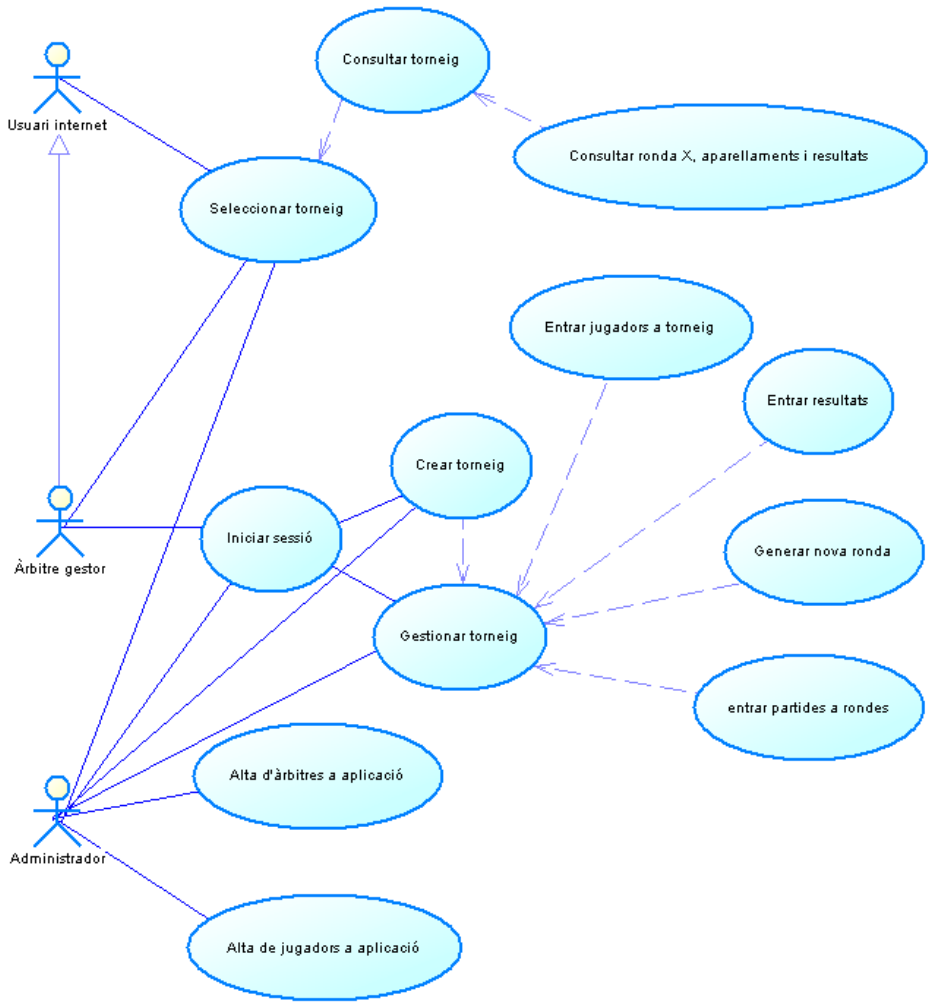

Figura 3 - Diagrama de Casos d'ús

b) Diagrama de classes entitat

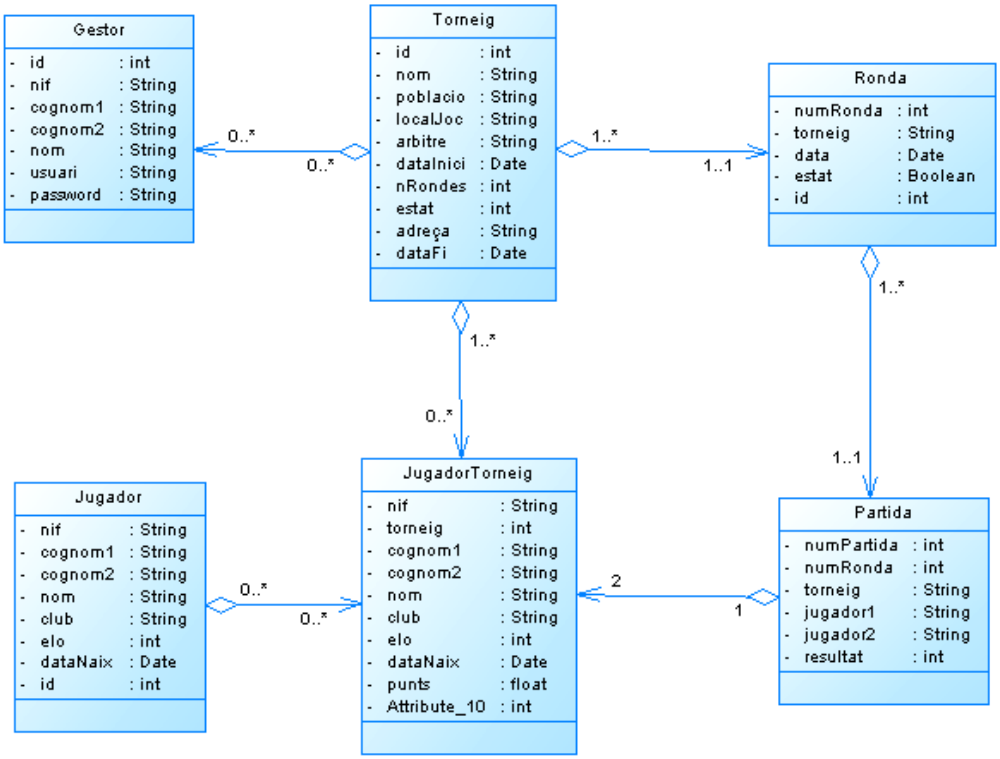

Figura 4 - Diagrama UML de Classes

c) Diagrama d'estats

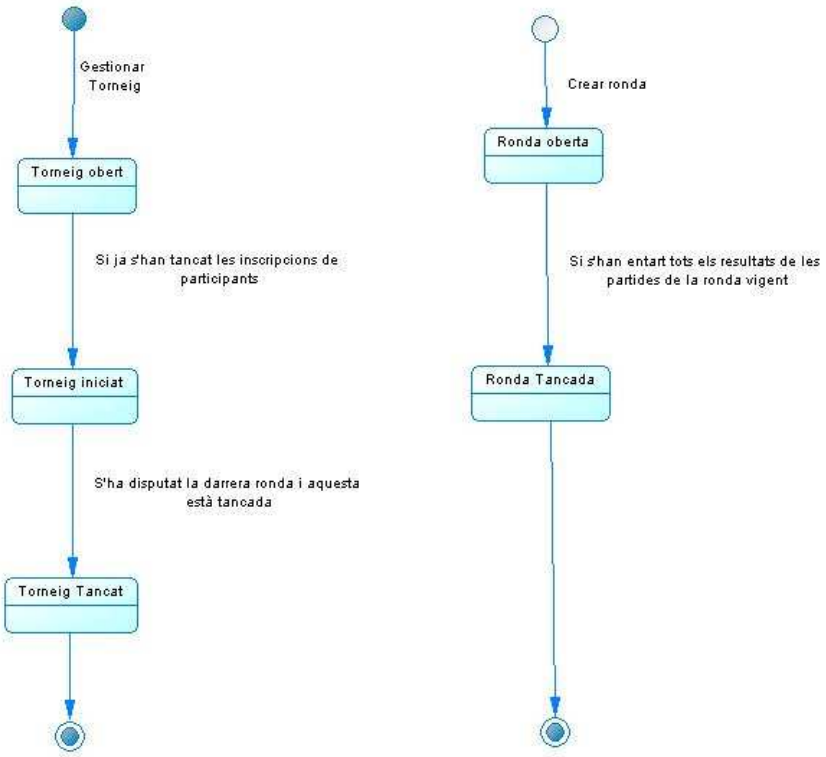

Figura 5 - Diagrama d'Estats

d) Esquema funcional

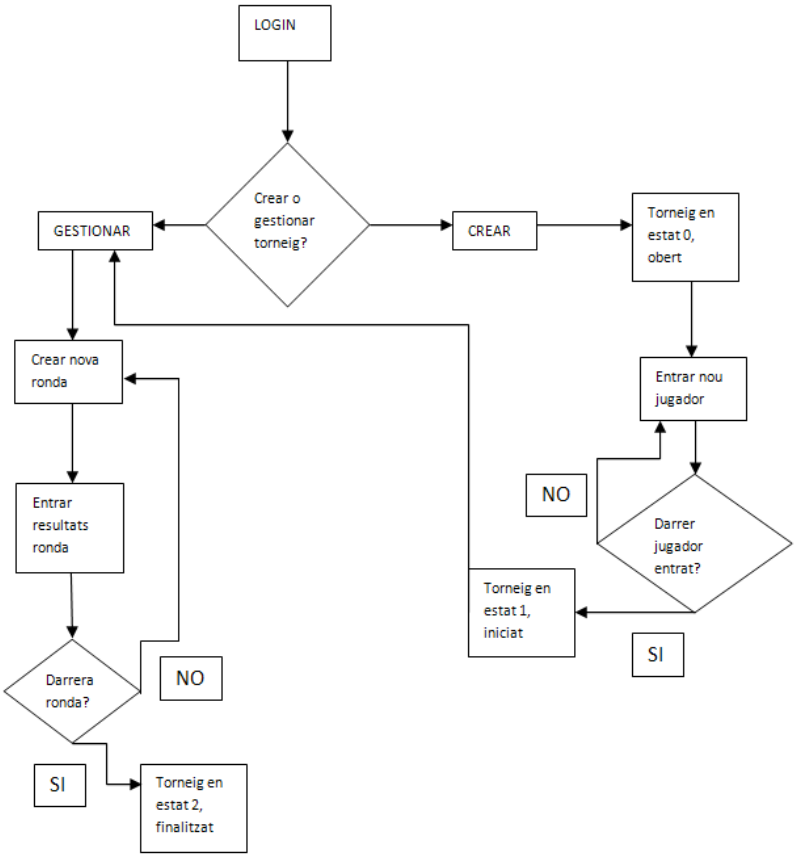

Figura 6 - Diagrama Funcional

e) Diagrama de persistència ER (Entitat Relació)

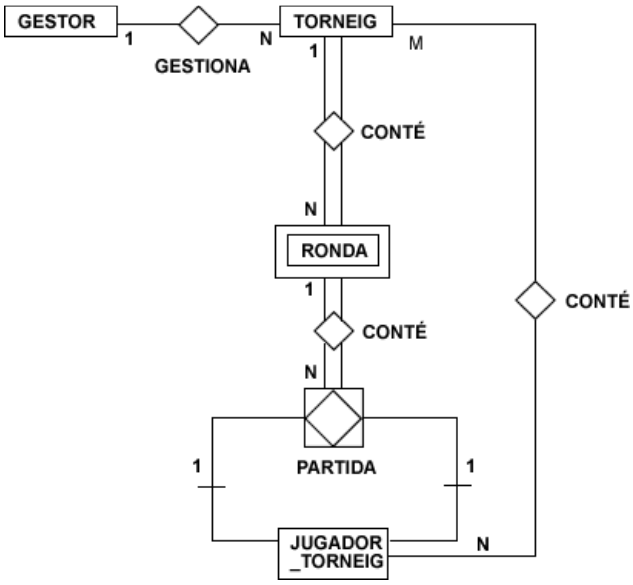

Figura 7 - Diagrama Entitat Relació

f) SQL per a MySql

Aquí es mostra el codi SQL de creació de la Base de Dades, per MySql, que es podrà carregar directament al SGBD. S'ha pogut escriure a partir de l'esquema ER dissenyat prèviament.

```
CREATE DATABASE IF NOT EXISTS chess;
CREATE TABLE gestor (
id INT AUTO_INCREMENT,
nif varchar(10),
cognom1 varchar(30),
cognom2 varchar(30),
nom varchar(30),
usuari varchar(10),
password varchar(10),
PRIMARY KEY (id)
);
CREATE TABLE torneig (
id INT AUTO_INCREMENT,
nom varchar(50),
poblacio varchar(30),
adreca varchar(50),
arbitre INT NOT NULL,
d_inici date,
d_fi date,
n_rondes int,
estat Integer DEFAULT 0,
PRIMARY KEY (id),
FOREIGN KEY (arbitre) REFERENCES gestor(id)
);
CREATE TABLE ronda (
id INT AUTO_INCREMENT,
n_ronda int NOT NULL,
torneig int NOT NULL,
data date,
estat int DEFAULT 0,
PRIMARY KEY (id),
FOREIGN KEY (torneig) REFERENCES torneig(id)
);
CREATE TABLE jugador(
id INT AUTO_INCREMENT,
nif varchar(10),
cognom1 varchar(30),
cognom2 varchar(30),
nom varchar(30),
club varchar(30),
elo int,
data_naixement date,
PRIMARY KEY (id)
);
CREATE TABLE jugador_torneig(
id INT AUTO_INCREMENT ,
torneig int NOT NULL,
jugador int NOT NULL,
punts float DEFAULT 0 NOT NULL,
PRIMARY KEY (id),
FOREIGN KEY (torneig) REFERENCES torneig(id),
FOREIGN KEY (jugador) REFERENCES jugador(id)
);
CREATE TABLE partida(
id INT AUTO_INCREMENT,
n_partida int NOT NULL,
ronda int NOT NULL,
jugador1 int NOT NULL,
```

```
jugador2 int NOT NULL,
resultat int DEFAULT 0,
PRIMARY KEY (id),
FOREIGN KEY (ronda) REFERENCES ronda(id),
FOREIGN KEY (jugador1) REFERENCES jugador_torneig(id),
FOREIGN KEY (jugador2) REFERENCES jugador_torneig(id));
```

Capítol 3: **Aprenentatge i tria de les tecnologies**

La fase d'aprenentatge ha estat força llarga i he hagut de cercar i llegir molta documentació per poder arribar a fer una selecció de les tecnologies més adequades a les característiques del projecte.

El primer pas ha estat decidir si treballar amb algun entorn de programació IDE o no, i si és el cas de la primera opció, estudiar quines possibilitats hi ha. Els dos entorns més coneguts per treballar amb JAVA, i que ja havia tocat durant el transcurs de l'enginyeria són *l'Eclipse* i el *NetBeans*. El primer el vaig tocar més cap al principi amb POO i Estructura de la Informació, i el segon, durant el semestre anterior al present amb l'assignatura de Tècniques de Desenvolupament de Programari per J2SE. Així, havent après recentment algunes de les particularitats del *NetBeans*, finalment he optat per a aquest producte.

Per tal de començar a treballar amb J2EE, es configura l'entorn dins de l'equip amb el que treballarem. Això ens porta a visitar la web de www.sun.com i cercar la versió més recent de l'entorn *JDK, JRE* i *J2EE*.

http://www.oracle.com/technetwork/java/javase/downloads/index.html#need

Aquí trobem una versió amb l'entorn J2EE integrada amb l'IDE NetBeans i no dubtem ni un moment a començar-ne la seva descàrrega.

http://netbeans.org/downloads/index.html?intcmp=3513

Una vegada que hem instal·lat la base on començar a treballar, tocarà configurar les variables d'entorn, com ja hem après a fer en tantes altres ocasions. Simplement tenim d'avisar al sistema operatiu d'on es troben les subcarpetes \bin i \lib del *java*, on hi tenim les llibreries i els executables per poder compilar i executar els programes fets amb JAVA.

Quan ja tenim això instal·lat i configurat, podem generar un nou projecte des del NBeans i veurem com tenim un ventall d'opcions a seleccionar quant a la tria de *Frameworks* i *Servidors*. En aquest punt ens hem d'aturar a fer un exercici de documentació molt important, i demanar consell si cal a persones més experimentades en aquest àmbit.

Capítol 4: Disseny de l'arquitectura

Llegint de diverses fonts, s'arriba a la conclusió que el patró més utilitzat actualment per fer aquest tipus d'aplicacions, és el MVC (Model Vista Controlador), que està format per diverses capes, on cadascuna té la seva funció. S'ha realitzat un esquema visual per tal de tenir una idea més clara i senzilla del que és aquest patró:

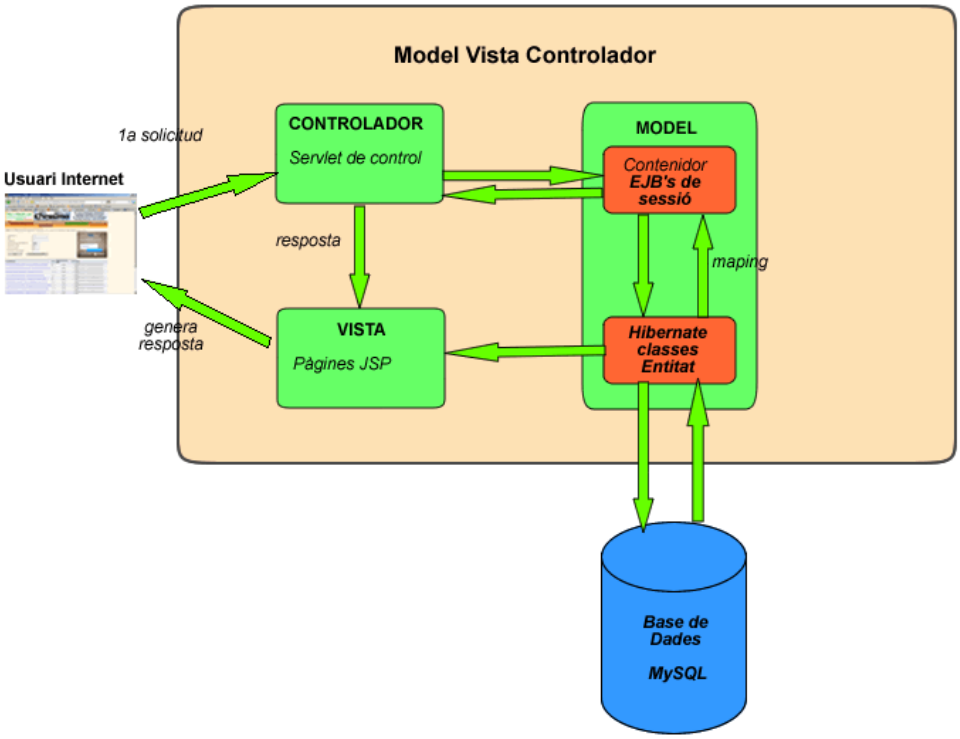

Figura 8 - Esquema del model MVC (Model Vista Controlador)

L'usuari es connectarà a l'aplicació via Internet, mitjançant un navegador web que treballarà al servidor. És a dir, tota la feina es fa des del Servidor, i la màquina client simplement enviarà i rebrà les dades a visualitzar pel navegador. Això fa que ens estalviem de fer la instal·lació de la part client a cada màquina client, ja que amb la simple traducció dels processos que es generen al servidor, a codi HTML, XML o JSP, en tindrem prou per tal d'utilitzar aquest tipus d'aplicacions.

El navegador enviarà les ordres a través d'un *Servlet* controlador i aquest podrà enviar aquestes ordres al model amb els contenidors EJB, que per la seva banda es comunicaran amb la base de dades i el contingut serà enviat cap a unes pàgines JSP que seran les encarregades de mostrar la informació al navegador d'Internet perquè l'usuari pugui anar interactuant amb l'aplicació.

Els principals avantatge que ens ofereix aquest model, són la separació de l'aplicació per capes, que ens permetrà construir i actualitzar el codi font, d'una forma molt ordenada, i saber

on anar a trobar els errors; per altra banda, el fet de no treballar directament amb les dades de la base de dades, sinó que seran les Entitats de classe que modificaran les dades, aporta certa millora quant a seguretat i consistència de dades.

Capítol 5: Preparació de l'entorn

Per tal de donar forma a tot aquest projecte, existeixen uns quants *Frameworks* de desenvolupament ràpid d'aplicacions, i n'hi ha una gran varietat en el mercat. Això provoca que inicialment no sabrem massa per on tirar a l'hora de fer la selecció del que necessitarem, i haurem de fer moltes proves abans de prendre una direcció o una altra.

Quan obrim el *NetBeans* per primera vegada, aquest ens ofereix una guia d'auto aprenentatge que molt complerta, i aquest és el camí a iniciar per tal d'aprendre sobre la tecnologia, i poc a poc anem entenent i fent-nos un dibuix mental de tot l'entorn amb el que haurem de treballar.

I per posar un punt de partida, comencem amb el servidor *GlassFish Server 3*, amb el *Framework* pel model MVC *Struts2* i per a les vistes farem servir les pàgines *JSP*. Pel que fa a la persistència, es dubta entre el *PostgreSql* i el *MySql*. Inicialment s'instal·la el *PostgreSql* i se'n configura el connector JDBC i més tard es connecta amb el NBeans amb èxit. Però quan s'aconsegueix fer la primera mini aplicació que mostri les dades d'una taula de la base de dades per pantalla des del navegador, ens topem amb un escull que no ens deixa avançar, i com que ens adonem que per la xarxa hi existeix més informació per *MySql*, finalment es decideix optar per a aquest SGBD (Sistema Gestor de Bases de Dades).

Així finalment el punt de partida serà:

JSP + GlassFish + Struts2 + Hiberate + MySql

Finalment amb l'ajuda d'aquests enllaços s'aconsegueix un punt de partida prou bo: des d'un navegador es poden donar d'alta nous àrbitres, eliminar-los i llistar els existents a la taula *gestor*. I això es fa mitjançant la classe *Gestor* amb els *Beans* Entitat, com a intermediària entre la Base de Dades i la capa vista, i tot això també amb l'ajuda d'*Hibernate*.

Per fer-ho s'han seguit aquestes passes:

1. Crear la Base de dades amb MySql mitjançant l'Script de creació de base de dades.

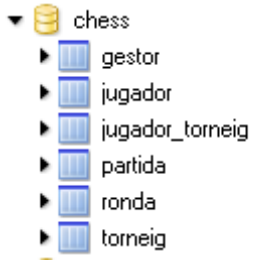

Figura 9 - Taules de la Base de Dades

2. Crear l'aplicació amb NetBeans amb Struts2 + Hibernate

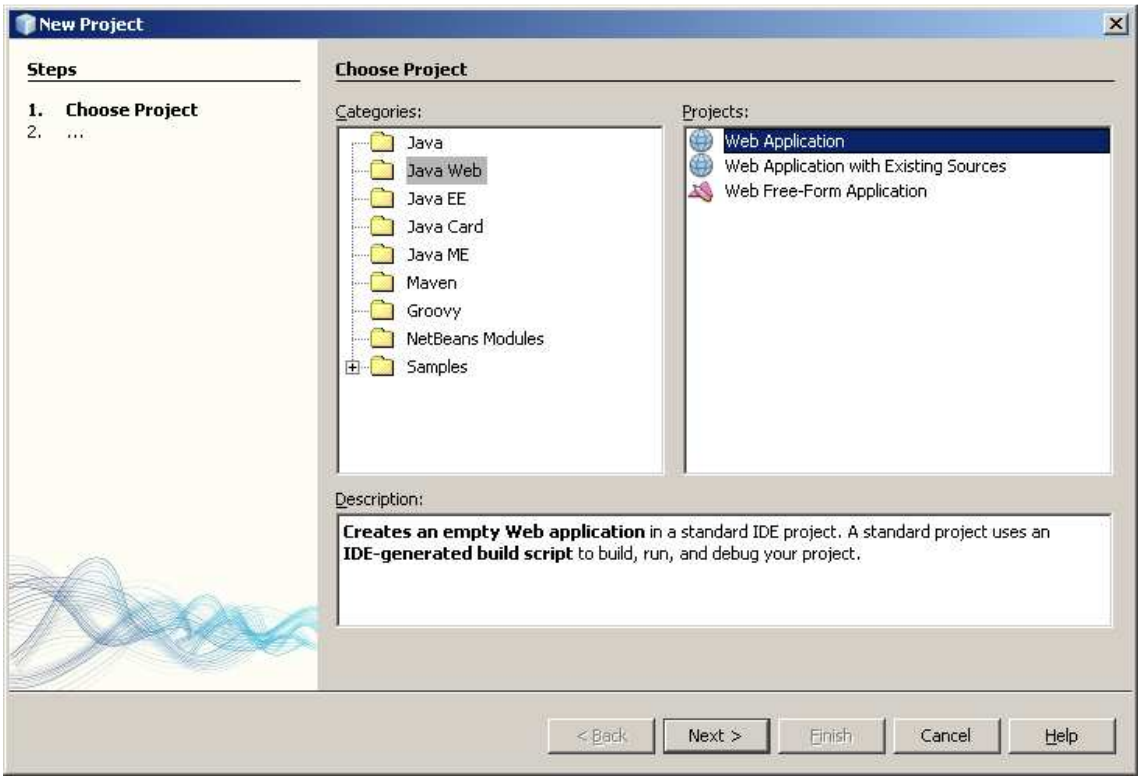

Figura 10 - Nou projecte amb NBeans

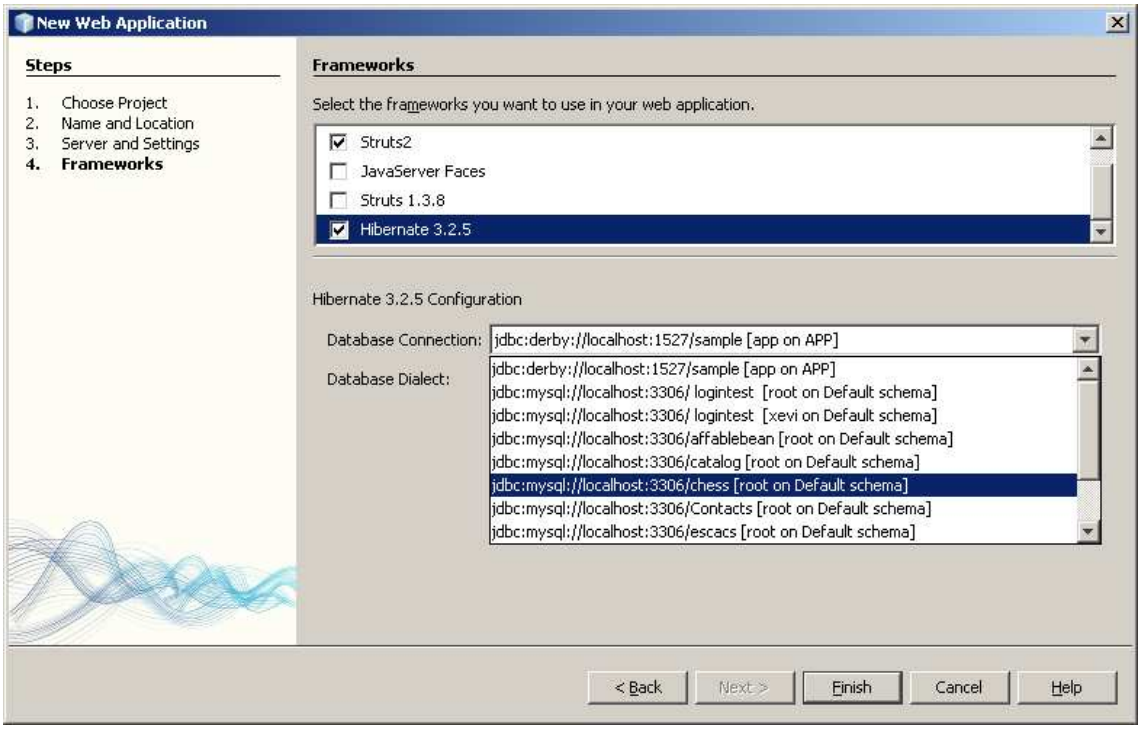

Figura 11 - Tria de *Frameworks* i connexió

Amb això ja tindrem l'entorn de programació a punt, per començar a programar:

- Entorn de desenvolupament *NetBeans*
- Un servidor *GlassFish*
- Un SGBD *MySql* amb la BD creada
- *Struts2* + *Hibernate*

Capítol 6: Fase de Desenvolupament

6.1. Preparació del Servidor

Tenint en compte el Model Vista Controlador i les tecnologies utilitzades per fer el desenvolupament, en aquest document es donaran les pautes a seguir per tal de poder executar l'aplicació en un PC, en local. El sistema operatiu utilitzat ha estat un *Microsoft Windows XP*, i s'ha utilitzat l'entorn de programació NetBeans.
La prova pilot s'ha fet amb un *Windows XP* professional 32 bits, instal·lat des de zero a una VMWare Virtual.

El principal avantatge d'utilitzar aquest IDE ha estat que descarregant la versió adequada, amb ella ja hi anaven incorporats l'entorn J2EE i diferents servidors i *Frameworks* a escollir, però també podem instal·lar separadament cada component. Es pot realitzar la descàrrega de tot plegat integrat amb el NetBeans, des d'aquí:

http://netbeans.org/downloads/index.html?intcmp=3513

Tanmateix, en els següents apartats, donarem les passes a seguir per tal de tenir el sistema a punt, sense necessitat d'instal·lar cap IDE.

En aquest segon cas, els instal·ladors i fitxers que necessitem són:

- Mysql-5.1.53-win32
- Mysql-gui-tools-5.0-r17-win32
- Mysql-connector-java-5.1.13
- Jpiinstall (eines d'administració de *MySql*)
- Jdk-6u22-windows-i586
- Java_ee_sdk-6u1-web-windows

a) Sistema Gestor de Bases de Dades *MySql 5.1.53 i connector JDBC*

Aquí s'exposen els passos a seguir per preparar la Base de Dades del sistema:

- Descarregar i instal·lar la versió 5.1.53: http://dev.mysql.com/downloads/
- Incloem una excepció al tallafocs pel port 3306.
- Afegim el directori BIN al *Path* del *Windows* (Variable d'entorn)
- En el nostre cas configurem la connexió amb un usuari administrador "*root*" i el seu mot de pas "*1234*".

- Descarreguem i instal·lem les eines d'administració visuals: http://dev.mysql.com/get/Downloads/MySQLGUITools/mysql-gui-tools-5.0-r17-win32.msi/from/pick
- Li donem la connexió a *localhost* i com a esquema *mysql*
- També és necessari descarregar i instal·lar el connector JDBC que ens connectarà la base de dades amb la nostra aplicació JAVA. Ens el podem descarregar des de: http://dev.mysql.com/downloads/connector/j/ i el podem posar en una carpeta juntament on hi tenim el MySql, així el localitzarem amb més facilitat. Però més endavant cal situar-lo a una carpeta del servidor web.

Creem un nou esquema amb el *Qwery Brownser* del *Mysql* i li donem el nom 'chess' i després passem a executar l'Script de creació de les taules que s'ha mostrat més amunt (L'script també es pot trobar a la carpeta del projecte, dins la subcarpeta BD)

Seguidament ja podem fer algunes insercions de prova, per més tard poder llistar registres de les diferents taules pel navegador (Aquest *Script* també el trobarem a la carpeta del projecte, dins la subcarpeta BD):

```
-- INSERTAR GESTORS--
INSERT INTO gestor (nif, cognom1, cognom2, nom, usuari, password) VALUES
('33333333','Benet', 'Saniger', 'Xavier','xbenet','1234');
INSERT INTO gestor (nif, cognom1, cognom2, nom, usuari, password) VALUES
('11111111','Bosch', 'Casacuberta', 'Lluís', 'urena','34567');
INSERT INTO gestor (nif, cognom1, cognom2, nom, usuari, password) VALUES
('22222222','Jordà', 'Solà', 'Maria','jsola','2258');
INSERT INTO gestor (nif, cognom1, cognom2, nom, usuari, password) VALUES
('44444444','Vilardell', 'Roca', 'Toni', 'tvila','34567');

-- INSERTAR TORNEJOS--
INSERT INTO torneig ( nom, poblacio, adreca, arbitre, d_inici, d_fi, n_rondes)VALUES
('Open Calldetenes', 'Calldetenes','Plaça

Major',1,'2010-07-15','2010-10-15',8);
INSERT INTO torneig ( nom, poblacio, adreca, arbitre, d_inici, d_fi, n_rondes)VALUES (
'Open de Gurb', 'Gurb','Plaça Menor', 1,

'2010-11-11','2011-12-11',9);
INSERT INTO torneig ( nom, poblacio, adreca, arbitre, d_inici, d_fi, n_rondes)VALUES (
'Open de Vic', 'Vic','Casino', 2,

'2010-11-15','2010-11-23',8);
INSERT INTO torneig ( nom, poblacio, adreca, arbitre, d_inici, d_fi, n_rondes)VALUES (
'Open de Taradell', 'Taradell','H.Rosa', 4,

'2011-01-15','2011-01-23',8);

-- INSERTAR JUGADORS--
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('00000000','Solà', 'Lluch', 'Miquel','Vic', 2100,

'1950-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('22222222','Homs', 'Piqué', 'Josep','Calldetenes', 2000,

'1975-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('00000000','Font', 'Torras', 'Joan','Taradell', 1900,

'1950-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('22222222','Ferrer', 'Ferrer', 'Esteve','Roda', 1754,
```

```
'1975-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('00000000','Moreta', 'Casas', 'Carles','Vic', 2130,

'1950-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('22222222','Casas', 'Piqué', 'Josep','Calldetenes', 2000,

'1975-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('00000000','Verdaguer', 'Mas', 'Xevi','Vic', 2100,

'1950-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('22222222','Saborit', 'Parareda', 'Pep','St.Joan', 2000,

'1975-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('00000000','Pladevall', 'Parramon', 'Enric','Vic', 2100,

'1950-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('22222222','Andreu', 'Costa', 'Gerard','Vic', 2000,

'1975-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('00000000','Sanchez', 'Fernandez', 'Roger','Vic', 2100,

'1950-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('22222222','Piñol', 'Rico', 'Genis','Calldetenes', 2000,

'1975-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('00000000','Pujol', 'Benet', 'Jordi','Calldetenes', 2100,

'1950-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('22222222','Roscalleda', 'Capdevila', 'Laia',

'Calldetenes', 2000, '1975-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('00000000','Roure', 'Arderiu', 'Miquel','Vic', 2100,

'1950-02-01');
INSERT INTO jugador (nif, cognom1, cognom2, nom, club, elo, data_naixement) VALUES
('22222222','Llop', 'Buenafuente', 'Josep','Taradell',
2000, '1975-02-01');

-- INSERTAR JUGADORS TORNEIGS ---
insert into jugador_torneig (torneig, jugador) VALUES(1,1);
insert into jugador_torneig (torneig, jugador) VALUES(1,2);
insert into jugador_torneig (torneig, jugador) VALUES(1,3);
insert into jugador_torneig (torneig, jugador) VALUES(1,4);
insert into jugador_torneig (torneig, jugador) VALUES(1,5);
insert into jugador_torneig (torneig, jugador) VALUES(1,6);
insert into jugador_torneig (torneig, jugador) VALUES(1,7);
insert into jugador_torneig (torneig, jugador) VALUES(1,8);
insert into jugador_torneig (torneig, jugador) VALUES(2,1);
insert into jugador_torneig (torneig, jugador) VALUES(2,2);
insert into jugador_torneig (torneig, jugador) VALUES(2,3);
insert into jugador_torneig (torneig, jugador) VALUES(2,4);
insert into jugador_torneig (torneig, jugador) VALUES(3,1);
insert into jugador_torneig (torneig, jugador) VALUES(3,2);
insert into jugador_torneig (torneig, jugador) VALUES(3,3);
insert into jugador_torneig (torneig, jugador) VALUES(3,4);
insert into jugador_torneig (torneig, jugador) VALUES(3,5);
```

```
insert into jugador_torneig (torneig, jugador) VALUES(3,6);
insert into jugador_torneig (torneig, jugador) VALUES(3,7);
insert into jugador_torneig (torneig, jugador) VALUES(3,8);

-- INSERTAR RONDES ---
insert into ronda (torneig, n_ronda, data) VALUES(1, 1, '2010-10-16');
insert into ronda (torneig, n_ronda, data) VALUES(1, 2, '2010-10-17');
insert into ronda (torneig, n_ronda, data) VALUES(1, 3, '2010-10-18');
insert into ronda (torneig, n_ronda, data) VALUES(1, 4, '2010-10-19');
insert into ronda (torneig, n_ronda, data) VALUES(1, 5, '2010-10-20');
insert into ronda (torneig, n_ronda, data) VALUES(1, 6, '2010-10-21');
insert into ronda (torneig, n_ronda, data) VALUES(1, 7, '2010-10-22');
insert into ronda (torneig, n_ronda, data) VALUES(1, 8, '2010-10-23');
insert into ronda (torneig, n_ronda, data) VALUES(2, 1, '2010-10-16');
insert into ronda (torneig, n_ronda, data) VALUES(2, 2, '2010-10-17');
insert into ronda (torneig, n_ronda, data) VALUES(2, 3, '2010-10-18');
insert into ronda (torneig, n_ronda, data) VALUES(2, 4, '2010-10-19');
insert into ronda (torneig, n_ronda, data) VALUES(2, 5, '2010-10-20');
insert into ronda (torneig, n_ronda, data) VALUES(2, 6, '2010-10-21');
insert into ronda (torneig, n_ronda, data) VALUES(2, 7, '2010-10-22');
insert into ronda (torneig, n_ronda, data) VALUES(2, 8, '2010-10-23');

-- INSERTAR PARTIDES ---
insert into partida (n_partida, ronda, jugador1, jugador2) VALUES(1,1,1,2);
insert into partida (n_partida, ronda, jugador1, jugador2) VALUES(2,1,3,4);
insert into partida (n_partida, ronda, jugador1, jugador2) VALUES(3,1,5,6);
insert into partida (n_partida, ronda, jugador1, jugador2) VALUES(4,1,7,8);
insert into partida (n_partida, ronda, jugador1, jugador2) VALUES(1,2,1,2);
insert into partida (n_partida, ronda, jugador1, jugador2) VALUES(2,2,3,4);
insert into partida (n_partida, ronda, jugador1, jugador2) VALUES(3,2,5,6);
insert into partida (n_partida, ronda, jugador1, jugador2) VALUES(4,2,7,8);
```

b) Base de J2EE i Servidor *GlassFish 3.0.1*

Necessitem l'entorn d'execució de *JAVA*, que el podem descarregar des de:
http://java.com/es/download/index.jsp

Seguidament hem de descarregar i instal·lar el JDK per J2SE:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Des d'aquest enllaç ens podem descarregar una versió del J2EE SDK que incorpora el servidor GlassFish 3.0.1 web: http://www.oracle.com/technetwork/java/javaee/downloads/index.html

GlassFish and Java EE 6 SDKs Update 1 *(Last Updated: June 17,2010*					
Components	**Java EE 6 Web Profile SDK**	**Java EE 6 SDK**	**NetBeans IDE 6.9**	**GlassFish Tools Bundle for Eclipse ***	**Java EE 5 SDK**
Description	A free integrated development kit used to build, test, and deploy Java EE 6 web applications.	A free integrated development kit used to build, test, and deploy Java EE 6 applications.	Integrated support for GlassFish Server Open Source Edition 3.0.1 and the entire Java EE 6 spec, including JSF 2.0, Java Persistence 2.0, and EJB 3.1.	Rapid iterative development of Java EE 6 applications with the Eclipse IDE and GlassFish v3.	Java EE 5 SDK is a free integrated development kit used to build, test, and deploy Java EE 5-based applications. Multilingual support is available.
	Download	**Download**	**Download**	**Download**	**Download**
GlassFish 3.0.1		•	•		
GlassFish 3.0.1 Web Profile	•				
GlassFish v2.1.1					•
Java EE 6 Samples	•	•	•		
Java EE 5 Samples					•
API Documentation	•	•			
Tutorial	•	•			

Figura 12 - Descàrrega J2EE SDK + GlassFish

Aquí ja ens hi ve incorporada la base de J2ee amb el SDK de desenvolupament Java. Una vegada descarregat el fitxer l'executem i seguim les passes.

Seguidament haurem de configurar el servidor. Per fer-ho seguim les passes següents:

- Executem el fitxer *start-serv.bat que es troba* a *C:\glassfishv3\glassfish\lib*, per iniciar el servidor
- A una finestra del nostre navegador accedim a http://localhost:4848/login.jsf

6.2. Desenvolupament de l'aplicació

a) Codi base de configuració i Entity Beans

Ha arribat el moment de posar fil a l'agulla i començar a donar forma a aquest treball, amb la fase de desenvolupament. Estudiant una mica més el model MVC i els *Frameworks Struts2 + Hibernate*, i consultant i provant alguns exemples que trobem, finalment decidim començar per provar de crear una Entitat *Gestor*, que correspondrà a la fila *gestor* de la base de dades.

L'aplicació en aquest punt ens ha de permetre crear, eliminar i llistar els gestors de la base de dades (àrbitres).

Resumint, el que s'ha de fer perquè aquesta entitat doni la volta per totes les capes del circuit de l'entorn, és seguir aquests 9 passos:

1. Crear el model vista controlador MVC, mitjançant paquets (*packages*)
2. Crear una pàgina *gestors.jsp* on visualitzar els resultats i interactuar amb aquesta entitat
3. Configurar el fitxer *hibernate.cfg.xml*
4. Crear el fitxer *HibernateUtil.java* dins el paquet Util
5. Crear la classe *Gestor.java* dins el paquet *Model*
6. Crear la classe controladora *GestorManger.java* en el paquet *Controlador*
7. Configurar fitxer *web.xml* per apuntar a *gestor.jsp*
8. Configurar fitxer *Struts.xml* amb les accions interceptores
9. Crear classe *GestorAction.java* al paquet *Vista*

Crear el model MVC

Es crea la jerarquia de paquets, d'aquesta manera:

- tfc.escacs.controlador
- tfc.escacs.model
- tfc.escacs.util
- tfc.escacs.vista

Tots dins el *Source Packages* i es manté el <Default>, que és on hi anirà l'*struts.xml* i l'*hiberbate.cfg.xml*

Crear *gestors.jsp* dins de la carpeta *Web Pages*

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>

<html>
<head>
          <title>gestió àrbitres</title>
</head>
<body>

    <center>
        <img alt="titol"  src="img/titol.jpg" width="790" height="51">
        <table>
            <td><a href ="index.html">Pàgina d'inici</a></td>
            <td><a href ="tornejos.jsp">Gestió de tornejos</a></td>
        </table>
        <br>
<h3>Nou àrbitre</h3>

     <s:actionerror/>

        <s:form action="addGestor" method="post">

            <s:textfield name="gestor.cognom1" label="1r cognom"/>
            <s:textfield name="gestor.cognom2" label="2n cognom"/>
            <s:textfield name="gestor.nom" label="nom"/>
            <s:textfield name="gestor.usuari" label="usuari"/>
            <s:textfield name="gestor.nif" label="nif"/>
            <s:textfield name="gestor.password" label="password"/>

            <s:submit value="Afegir àrbitre" align="center"/>
        </s:form>

<h2>Llistat d'àrbitres</h2>
<table>
<tr>    <th><div align="left">id</div></th>
        <th><div align="left">    -    </div></th>
          <th><div align="left">nom</div></th>
        <th><div align="left">    -    </div></th>
          <th><div align="left">cognom1</div></th>
        <th><div align="left">    -    </div></th>
          <th><div align="left">cognom2</div></th>
        <th><div align="left">    -    </div></th>
          <th><div align="left">usuari</div></th>
        <th><div align="left">    -    </div></th>
        <th><div align="left">NIF</div></th>
        <th><div align="left">    -    </div></th>
          <th><div align="left">Delete</div></th>
</tr>
<s:iterator value="gestorList" var="gestor">
          <tr>
            <td><s:property value="id"/></td>
            <td>    |    </td>
            <td><s:property value="nom"/></td>
            <td>    |    </td>
            <td><s:property value="cognom1"/></td>
            <td>    |    </td>
            <td><s:property value="cognom2"/></td>
            <td>    |    </td>
            <td><s:property value="usuari"/></td>
            <td>    |    </td>
            <td><s:property value="nif"/></td>
            <td>    |    </td>
            <td><a href="deleteGestor?id=<s:property value="id"/>">delete</a></td>
          </tr>
</s:iterator>
</table>
    </center>
</body>
</html>
```

Configurar *hibernate.cfg.xml* dins el paquet *<Default>*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE     hibernate-configuration     PUBLIC     "-//Hibernate/Hibernate     Configuration     DTD     3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">
                              com.mysql.jdbc.Driver
                    </property>
    <property name="connection.url">jdbc:mysql://localhost:3306/chess</property>
    <property name="connection.username">root</property>
```

```
    <property name="connection.password">1234</property>
    <property name="connection.pool_size">1</property>
    <property name="dialect">
                            org.hibernate.dialect.MySQLDialect
                    </property>
    <property name="current_session_context_class">thread</property>
    <property name="cache.provider_class">
                            org.hibernate.cache.NoCacheProvider
                    </property>
    <property name="show_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping class="tfc.escacs.model.Gestor"/>
  </session-factory>
</hibernate-configuration>
```

Crear el fitxer *HibernateUtil.java* dins el paquet *Util*

```
package tfc.escacs.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateUtil {

          private static final SessionFactory sessionFactory = buildSessionFactory();

          private static SessionFactory buildSessionFactory() {
                    try {
                              // Create the SessionFactory from hibernate.cfg.xml
                              return new AnnotationConfiguration().configure()
                                                  .buildSessionFactory();
                    } catch (Throwable ex) {
                              System.err.println("Initial SessionFactory creation failed." + ex);
                              throw new ExceptionInInitializerError(ex);
                    }
          }

          public static SessionFactory getSessionFactory() {
                    return sessionFactory;
          }
}
```

Crear el fitxer *Gestor.java* dins el paquet *Model*

```
package tfc.escacs.model;

import java.io.Serializable;
import java.sql.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="gestor")
public class Gestor implements Serializable{

          private static final long serialVersionUID = -8767337896773261247L;

          private Long id;
          private String nom;
          private String cognom1;
          private String cognom2;
          private String usuari;
          private String password;
        private String nif;

    public String getNif() {
        return nif;
    }

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

        //private Date created;

          @Id
          @GeneratedValue( strategy =  GenerationType.AUTO)
```

```
        @Column(name="id")
        public Long getId() {
                return id;
        }
        @Column(name="nom")
        public String getNom() {
                return nom;
        }
        @Column(name="cognom1")
        public String getCognom1() {
                return cognom1;
        }
        @Column(name="cognom2")
        public String getCognom2() {
                return cognom2;
        }
        @Column(name="usuari")
        public String getUsuari() {
                return usuari;
        }
        @Column(name="password")
        public String getPassword() {
                return password;
        }
        public void setId(Long id) {
                this.id = id;
        }
        public void setNom(String nom) {
                this.nom = nom;
        }
        public void setCognom1(String cognom1) {
                this.cognom1 = cognom1;
        }
        public void setCognom2(String cognom2) {
                this.cognom2 = cognom2;
        }
        public void setUsuari(String usuari) {
                this.usuari = usuari;
        }
        public void setPassword(String password) {
                this.password = password;
        }
}
```

Crear el fitxer *GestorManager.java* dins el paquet *Controlador*

```
package tfc.escacs.controlador;

import java.util.List;

import org.hibernate.HibernateException;
import org.hibernate.classic.Session;

import tfc.escacs.model.Gestor;
import tfc.escacs.util.HibernateUtil;

public class GestorManager extends HibernateUtil {

        public Gestor add(Gestor gestor) {
                Session session = HibernateUtil.getSessionFactory().getCurrentSession();
                session.beginTransaction();
                session.save(gestor);
                session.getTransaction().commit();
                return gestor;
        }
        public Gestor delete(Long id) {
                Session session = HibernateUtil.getSessionFactory().getCurrentSession();
                session.beginTransaction();
                Gestor gestor = (Gestor) session.load(Gestor.class, id);
                if(null != gestor) {
                        session.delete(gestor);
                }
                session.getTransaction().commit();
                return gestor;
        }

        public List<Gestor> list() {

                Session session = HibernateUtil.getSessionFactory().getCurrentSession();
                session.beginTransaction();
                List<Gestor> gestors = null;
                try {

                        gestors = (List<Gestor>)session.createQuery("from Gestor").list();

                } catch (HibernateException e) {
                        e.printStackTrace();
```

```
                                session.getTransaction().rollback();
                        }
                        session.getTransaction().commit();
                        return gestors;
                }
}
```

Configurar *web.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <filter>
        <filter-name>struts2</filter-name>
        <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>gestors.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Configurar *struts.xml*

```
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
        <constant name="struts.enable.DynamicMethodInvocation"
                    value="false" />
          <constant name="struts.devMode" value="false" />

          <package name="default" extends="struts-default" namespace="/">

                    <action name="addGestor"
                              class="tfc.escacs.vista.GestorAction" method="add">
                              <result name="success" type="chain">indexGestor</result>
                              <result name="input" type="chain">indexGestor</result>
                    </action>
                    <action name="deleteGestor"
                              class="tfc.escacs.vista.GestorAction" method="delete">
                              <result name="success" type="chain">indexGestor</result>
                    </action>

                 <action name="indexGestor"
                              class="tfc.escacs.vista.GestorAction">
                              <result name="success">gestors.jsp</result>
                    </action>

          </package>
</struts>
```

Crear el fitxer *GestorAction.java* dins el paquet *Vista*

```
package tfc.escacs.vista;
import java.util.List;
import tfc.escacs.controlador.GestorManager;
import tfc.escacs.model.Gestor;
import com.opensymphony.xwork2.ActionSupport;

public class GestorAction extends ActionSupport {

          private static final long serialVersionUID = 9149826260758390091L;
          private Gestor gestor;
          private List<Gestor> gestorList;
          private Long id;

          private GestorManager linkController;

          public GestorAction() {
                    linkController = new GestorManager();
          }
    @Override
```

```
        public String execute() {
                this.gestorList = linkController.list();
                System.out.println(gestorList);
                System.out.println(gestorList.size());
                return SUCCESS;
        }
        public String add() {
                System.out.println(getGestor());
                try {
                linkController.add(getGestor());
                }catch(Exception e) {
                }
                return SUCCESS;
        }
        public String delete() {
                linkController.delete(getId());
                return SUCCESS;
        }
        public Gestor getGestor() {
                return gestor;
        }
        public List<Gestor> getGestorList() {
                return gestorList;
        }
        public void setGestor(Gestor gestor) {
                this.gestor = gestor;
        }
        public void setGestorList(List<Gestor> gestorsList) {
                this.gestorList = gestorsList;
        }
        public Long getId() {
                return id;
        }
        public void setId(Long id) {
                this.id = id;
        }
}
```

A partir d'aquest punt, fem el mateix amb la resta d'entitats que han de conformar l'aplicació final, que coincideixen amb les taules de la Base de Dades, i ja tenim part de l'aplicació en funcionament. Ara tocarà anar polint i afinant totes les funcionalitats.

També hem de crear una pagina d'índex (index.html) amb els menús d'opcions, i que ens serveixi de porta d'entrada cap a la resta de funcionalitats.

El resultat obtingut després d'executar l'aplicació en aquest punt, és el que es mostra a continuació en un navegador web:

Figura 13 – Pantalla inicial

Torneig	Població	Adreça	Àrbitre	Data inici	Data fi	Rondes	Estat	Esborra
Open Calldetenes	Calldetenes	Plaça Major	1	15/07/10	15/10/10	8	0	Esborra
Open de Gurb	Gurb	Plaça Menor	1	11/11/10	11/12/11	9	0	Esborra
Open de Vic	Vic	Casino	2	15/11/10	23/11/10	8	0	Esborra
Open de Taradell	Taradell	H.Rosa	4	15/01/11	23/01/11	8	0	Esborra
www	654	ggg	17	15/07/73	20/04/11	10	0	Esborra
POPOPOPO	ggg	yyy	1	20/02/11	20/04/11	10	0	Esborra

Figura 14 – Pantalla Llistat Torneigs

A dalt hi tenim una barra de navegació amb la que podrem navegar per les diferents opcions de l'aplicació. Dins de cada opció, hi tenim un botó per llistar les dades relacionades amb la opció. A més, a cada registre hi tenim un enllaç que ens esborra el registre en qüestió. Al mateix temps podem trobar un formulari a cada opció, per anar introduint les dades.

b) Afegint rols a l'aplicació

La seguretat és una part molt important de les aplicacions web i en aquest cas s'ha volgut diferenciar entre tres perfils d'usuari. D'aquesta manera hi haurà tres zones ben diferenciades a dins de l'aplicació, que es separaran amb carpetes. Aquests rols són:

Administrador: serà l'encarregat de gestionar tot el que es pot gestionar dins de l'aplicació, i podrà fer tot el que poden fer els usuaris àrbitres i els usuaris navegants d'Internet.

Àrbitre: aquests usuaris podran gestionar totes les opcions de gestió, excepte les referents a la gestió d'àrbitres i a la gestió de jugadors en el sistema.

Usuari Internet: finalment aquests usuaris podran navegar per les diferents opcions de consulta, com si fos un àrbitre, però amb la diferència que no podran ni crear ni eliminar registres.

Per implementar la seguretat amb els rols, el primer que haurem de fer serà definir els usuaris i/o grups d'usuaris al servidor GlassFish. A continuació es definiran els rols en l'aplicació i finalment s'hauran d'associar cada usuari amb cada rol de l'aplicació.

Per fer el primer pas, hem d'obrir la consola d'administració del servidor *GlassFish* i obrir la pestanya *Seguretat*. A dintre haurem d'entrar a la carpeta *file* i des d'aquí ja podrem crear els usuaris necessaris amb els corresponents rols. Aquí s'ha creat un usuari Administrador de nom

admin i contrasenya ***admin***, i un usuari àrbitre de nom ***user*** i contrasenya ***user***. Aquests dos usuaris serviran per fer les proves pilot dels diferents rols. Crearem una pàgina d'inici nova, amb els enllaços per a cada zona segons tipus d'usuari. A més també necessitem una crear pàgina informativa per quan es produeixi un error.

Després haurem de modificar el fitxer *web.xml*, per tal de definir els rols, enllaçar cadascun amb la seva zona, definir el tipus d'autenticació (en aquest cas amb formulari), etc. El codi del fitxer quedarà així:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app             version="3.0"             xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <filter>
        <filter-name>struts2</filter-name>
        <filter-
class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-
class>
    </filter>
    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <security-constraint>
        <display-name>AdminConstraint</display-name>
        <web-resource-collection>
            <web-resource-name>Admin</web-resource-name>
            <description/>
            <url-pattern>/SeguretatAdmin/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <description>admin</description>
            <role-name>admin</role-name>
        </auth-constraint>
    </security-constraint>
    <security-constraint>
        <display-name>UserConstraint</display-name>
        <web-resource-collection>
            <web-resource-name>User</web-resource-name>
            <description/>
            <url-pattern>/seguretatUser/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <description>user</description>
            <role-name>user</role-name>
        </auth-constraint>
    </security-constraint>
    <login-config>
        <auth-method>FORM</auth-method>
        <realm-name>file</realm-name>
        <form-login-config>
            <form-login-page>/login.jsp</form-login-page>
            <form-error-page>/loginError.html</form-error-page>
        </form-login-config>
    </login-config>
```

```
    <security-role>
        <description/>
        <role-name>Admin</role-name>
    </security-role>
    <security-role>
        <description/>
        <role-name>User</role-name>
    </security-role>
</web-app>
```

També cal modificar la pàgina *sun-web.xml* afegint-hi els rols creats:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD GlassFish Application Server 3.0 Servlet 3.0//EN" "http://www.sun.com/software/appserver/dtds/sun-web-app_3_0-0.dtd">
<sun-web-app error-url="">
  <context-root>/Escacs_TFC</context-root>
  <security-role-mapping>
    <role-name>admin</role-name>
    <principal-name>admin</principal-name>
  </security-role-mapping>
  <security-role-mapping>
    <role-name>user</role-name>
    <principal-name>user</principal-name>
  </security-role-mapping>
  <security-role-mapping>
    <role-name>Admin</role-name>
    <principal-name>admin</principal-name>
  </security-role-mapping>
  <security-role-mapping>
    <role-name>User</role-name>
    <principal-name>user</principal-name>
  </security-role-mapping>
  <class-loader delegate="true"/>
  <jsp-config>
    <property name="keepgenerated" value="true">
      <description>Keep a copy of the generated servlet class' java code.</description>
    </property>
  </jsp-config>
</sun-web-app>
```

Ara la pàgina d'inici serà *index*.jsp i ens mostrarà tres opcions diferents, cadascuna corresponent a un d'aquests rols:

Entrar com a visitant aquí!

Entrar com a àrbitre aquí!

Entrar com a Administrador aquí!

Figura 15 – index.jsp

Seleccionant l'opció d'usuari Internet, anirem a parar just a la pàgina amb el menú per aquests usuaris. En canvi per qualsevol de les altres dues opcions, se'ns obrirà el formulari de validació següent:

Figura 16 – Pantalla d'autenticació

Ara, si entrem algun usuari inexistent o cometem un error d'escriptura de nom d'usuari o contrasenya, el sistema ens redirigirà a una pàgina d'error, mostrant-nos el missatge:

Error d'autenticació!!, si us plau comprova que l'usuari i contrasenya siguin correctes...

Si l'opció escollida és la de visitant, el sistema ens enviarà directament a la pàgina d'inici per aquest rol, amb les opcions pertinents. Si es tracta d'algun dels altres dos rols, haurem d'omplir un formulari de validació que verificarà amb el servidor si l'usuari existeix o no. Seguidament, segons el rol escollit i si l'usuari en qüestió existeix en el sistema, l'aplicació ens durà al menú d'opcions segons sigui el rol, i en cas d'error d'autenticació, ens enviarà a una pàgina d'error.

Per tal de separar els diferents rols, s'han creat tres carpetes a l'arrel *Web Pages*, una pels usuaris web, *webUser*, una altra pels usuaris gestors *seguretatUser*, una tercera pels usuaris administradors *seguretatAdmin*. D'aquesta manera, la navegació permesa per cada tipus d'usuari es farà a dins de la seva carpeta.

Així finalment, per a l'usuari d'Internet tindrem les pantalles:

Figura 17 – Pantalla 1 usuari Internet

GESTIÓ DE TORNEJOS D'ESCACS

Inici | Tornejos | Jugadors Torneig | Rondes | Partides

Llistar Tornejos

Torneig	Àrbitre	Població	Adreça	Data inici	Data fi	Rondes	Estat
Open Calldetenes	1	Calldetenes	Plaça Major	15/07/10	15/10/10	8	0
Open de Gurb	1	Gurb	Plaça Menor	11/11/10	11/12/11	9	0
Open de Vic	2	Vic	Casino	15/11/10	23/11/10	8	0
Open de Taradell	4	Taradell	H.Rosa	15/01/11	23/01/11	8	0
Open Sants	1	Andorra	ggg	15/07/73	20/04/11	10	0
www	3	Llagosta	La vella	20/02/10	20/07/73	8	0
u	1	u	u	20/02/10	20/07/73	8	0

Figura 18 – Pantalla 2 usuari Internet

Les pantalles per a l'usuari Gestor seran:

GESTIÓ DE TORNEJOS D'ESCACS

Opcions

Gestionar tornejos

Gestionar jugadors per torneig

Gestionar rondes

Gestionar partides

Un usuari gestor podrà llistar, afegir o eliminar registres de tornejos, jugadors a tornejos, rondes i partides.

Figura 19 – Pantalla 1 usuari Gestor

GESTIÓ DE TORNEJOS D'ESCACS

Inici | Tornejos | Jugadors Torneig | Rondes | Partides |

Llistar Tornejos

Torneig	Població	Adreça	Àrbitre	Data inici	Data fi	Rondes	Estat	Esborra
Open Calldetenes	1	Calldetenes	Plaça Major	15/07/10	15/10/10	8	0	Esborra
Open de Gurb	1	Gurb	Plaça Menor	11/11/10	11/12/11	9	0	Esborra
Open de Vic	2	Vic	Casino	15/11/10	23/11/10	8	0	Esborra
Open de Taradell	4	Taradell	H.Rosa	15/01/11	23/01/11	8	0	Esborra
www	17	654	ggg	15/07/73	20/04/11	10	0	Esborra
POPOPOPO	1	ggg	yyy	20/02/11	20/04/11	10	0	Esborra

Torneig:
Població:
Adreça:
Àrbitre:
Data inici:
Data fi:
Rondes:

Afegir torneig

Figura 20 – Pantalla 2 usuari Gestor

Les pantalles per a l'usuari Administrador seran:

Figura 21 – Pantalla 1 usuari Administrador

Torneig	Població	Adreça	Àrbitre	Data inici	Data fi	Rondes	Estat	Esborra
Open Calldetenes	1	Calldetenes	Plaça Major	15/07/10	15/10/10	8	0	Esborra
Open de Gurb	1	Gurb	Plaça Menor	11/11/10	11/12/11	9	0	Esborra
Open de Vic	2	Vic	Casino	15/11/10	23/11/10	8	0	Esborra
Open de Taradell	4	Taradell	H.Rosa	15/01/11	23/01/11	8	0	Esborra
www	17	654	ggg	15/07/73	20/04/11	10	0	Esborra
POPOPOPO	1	ggg	yyy	20/02/11	20/04/11	10	0	Esborra

Figura 22 – Pantalla 2 usuari Administrador

c) Desplegament de l'aplicació

Possiblement hàgim de fer una aplicació per a un tercer, i aquest necessiti instal·lar l'aplicació a un servidor propi. En aquest cas li podem passar la jerarquia de carpetes i les passes a seguir per la instal·lació i desplegament d'una aplicació, a partir d'aquesta jerarquia més un fitxer *.EAR*, són les següents:

1. Copiar la carpeta comprimida *Escacs.TFC.rar* i descomprimir-lo a C:\glassfishv3\glassfish\domains\domain1\applications
2. Crear usuaris al Servidor, anant a *http://localhost:4848* i a dins del servidor accedim a /Seguretat/Dominis/file/Administrar usuaris. Aquí hem d'afegir un usuari ***admin*** amb contrasenya ***admin*** i al grup ***admin***; i un altre usuari ***user*** amb contrasenya ***user*** i al grup ***user***.

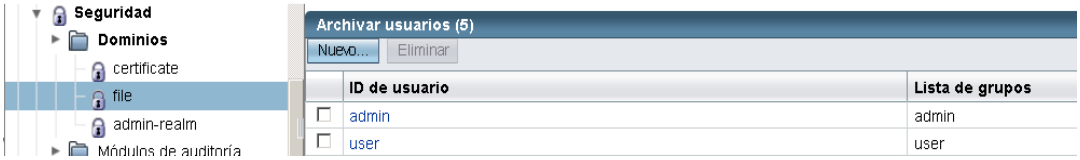

Figura 23 – Creació d'usuaris a GlassFish

3. Des de la consola d'administració del servidor, anar a *Aplications*, clickar al botó *Deploy* i anem a cercar el fitxer Escacs_TFC.war dins la carpeta 'dist' del projecte
4. A partir d'aquest primer hem de pensar a arrencar el Servidor GlassFish, i a partir d'aquest moment ja podrem disposar de l'aplicació web des de l'adreça local http://localhost:8080/Escacs_TFC

Per fer les proves amb els diferents rols, haurem de tenir en compte de reiniciar el navegador web, abans de canviar d'usuari, ja que l'aplicació està configurada perquè la durada d'una sessió sigui de 30', i en cas de no fer això, es produirà un error.

Capítol 7: Conclusions

Haver escollit aquesta àrea per fer el Treball Final de Carrera, va estar una decisió crec que valenta i arriscada, atès que potser m'hagués assegurat un bon resultat escollint algun altra àrea menys complexa i més coneguda per mi mateix. A mig semestre em vaig veure una mica atrapat dins de l'univers d'aquesta tecnologia, sense saber massa cap on tirar, però lluitant cada dia, poc a poc vaig anar trobant el camí.

El que m'ha aportat aquest TFC, ha estat una visió global sobre com s'ha de crear una aplicació web amb algunes de les tecnologies de J2EE, des del principi fins al final, i aquest aprenentatge m'ha ensenyat que la tecnologia és tan flexible i oberta, que potser aquest és un dels principals factors que fa que la corba d'aprenentatge inicial sigui tan i tan forta. A part d'haver vist com encaixar totes les peces perquè una aplicació connectada a una base de dades, funcioni des d'un navegador web, passant per totes les capes del circuit, també he après a configurar tot l'entorn amb un servidor GlassFish i un SGBD MySql.

Però són tantes les tecnologies que entren en joc, i les que podem escollir que un punt complicat és decidir-se cap a unes o altres, i quines combinacions farem entre elles. És cert que tenim alguns exemples a l'aula, però són per a una tecnologia i IDE's molt concrets, i en el meu cas no em van ser d'utilitat pràctica quant a concretar funcionalitats i casos d'ús, però si per tenir una visió de la jerarquia de fitxers, i un ordre a seguir en el procés d'elaboració del projecte.

Si hagués de tornar a fer aquest projecte faria alguns canvis respecte al tema escollit i les tecnologies seleccionades per desenvolupar-lo. En primer lloc, escolliria un tema més senzill, que amb dues o tres taules d'una Base de Dades n'hi hagués prou, i després escolliria unes tecnologies que complissin els requisits, però que estiguessin ben documentades amb exemples complerts a la documentació oficial d'algun dels IDE's més coneguts. El problema principal amb què m'he topat ha estat precisament no haver trobat aquests exemples per les tecnologies escollides (Struts2 i Hibernate). Crec que en aquests moments escolliria JSF en comptes de JSP, i JPA en comptes d'Hibernate. Però sense cap mena de dubte, aquests dos mesos i mig que m'he passat tancat, cercant informació, fent milers de proves, llegint molta informació, donant cops de cap a la paret... finalment m'hauran estat molt útils perquè encara que sigui molt dur haver d'aprendre d'aquesta manera, alguns conceptes que m'havíen de quedar clars, m'han quedat ben fixats.

Pel que fa a l'assoliment dels requeriments inicials, potser m'he quedat una mica curt en la programació, però és que sincerament no he tingut temps de fer més del que he fet: preparar l'entorn, programar totes les classes entitat (que no eren poques), mostrar per pantalla les consultes, entrar i esborrar registres des de pantalla i finalment la creació de tres rols diferents. També m'hagués agradat poder introduir llistes desplegables o alguns components més, però com ja he comentat més amunt, amb JSP no he estat capaç de trobat la manera de fer-ho, i crec que si hagués tirar cap a JSF, ho hagués aconseguit perquè vaig trobar exemples molt explícits al respecte.

En qualsevol cas, entenc que aquest TFC és un punt de partida per poder començar a endinsar-se en l'apassionant món de la Programació Orientada a Objectes en l'àmbit del web i que m'ha estat molt útil per fer-me una visió general dels patrons i tecnologies amb què es treballa actualment.

Glossari

Administrador És l'usuari amb més privilegis de l'aplicació que pot fer totes les funcionalitats relacionades amb els usuaris.

Àrbitre És un tipus d'usuari que pot fer el mateix que l'administrador excepte entrar jugadors i àrbitres al sistema.

Arquitectura En J2EE, és el que defineix l'encaix de les diferents tecnologies que hi entren en joc, i el funcionament i recorregut per tot el circuit, per part de les entitats.

Base de Dades És on s'emmagatzemen les dades amb les que volem treballar. Una aplicació de gestió no té sentit sense ella. També es pot treballar contra fitxers, però la potència d'accés a les dades mitjançant consultes complexes, fa que aquesta segona opció cada vegada s0usi menys per amb aquestes finalitats.

Cas d'ús És una funcionalitat que pot realitzar un actor del sistema.

Classe És un element essencial dels llenguatges de Programació Orientada a l'Objecte, i és com un motlle desl qual es poden anar creant objectes similars però amb atributs diferents.

Consulta Una consulta sobre la base de dades, serveix per extreure un resultat dels registres desitjats.

Desplegament Quan ja tenim tota l'aplicació feta i provada, es diu que el desplegament és quan traslladem aquesta aplicació a un servidor diferent del que s'ha creat l'aplicació. Haurem de facilitar un manual d'instal·lació.

Diagrama de Gantt Serveix per planificar en detall tot el procés d'un projecte, posant "sobre paper" les dates de cada activitat concreta.

Diagrama Entitat Relació És un diagrama de disseny de la base de dades, el pas previ a la seva creació amb SQL. Quan passem de la idea al paper.

Domini Conjunt d'elements que el conformen. En el cas del servidor web GlassFish, és allà on s'hi emmagatzemen les aplicacions.

EAR És el fitxer derivat de fer un desplegament a partir d'un fitxer *war*, i és el responsable de configurar els encaixos de tota la tecnologia.

EJB 'Entity Java Bean' podríem dir que és un objecte de classe que farà la funció de comunicar amb un element concret d'una taula de la Base de dades i que actuarà a les capes més altes de l'aplicació.

ELO En escacs, serveix per mesurar la força d'un jugador d'escacs, quan més, millor és el jugador.

Escacs Joc d'estratègia mil·lenari on s'enfronten dos jugadors, amb peces de diferents colors. Qui aconsegueix matar el rei adversari, haurà guanyat la partida.

Framework En català Marc de Treball. Són com unes llibreries que si les utilitzem, ens podem estalviar molta feina repetitiva, com per exemple enllaçar taules de la base de dades amb *Beans* d'entitat.

Funcionalitat És una part atòmica de l'aplicació, i el conjunt de totes elles li dóna sentit.

Gestor És una taula a la base de dades de la nostra aplicació, i també una Entitat de Java, que serveix per desar els que seran els àrbitres dels torneigs.

GlassFish Servidor d'aplicacions web per llenguatge J2EE, molt actual.

Hibernate És un *Framework* de creació ràpida, que ens fa més fàcil l'enllaç de l'aplicació amb la base de dades.

JDK És in kit de desenvolupament necessari per poder fer aplicacions en Java.

JRE Màquina virtual de Java, que ens permetrà executar *Applets* a la màquina on ho tinguem instal·lat.

j2EE "Java to Enterprise Edition" versió empresarial de desenvolupament d'aplicacions en Java, enfocat al web.

J2SE "Java to Standard Edition" versió estàndard de desenvolupament d'aplicacions en Java, enfocat a les aplicacions per escriptori.

Java Llenguatge de Programació Orientada a l'Objecte, obert i multiplataforma. Actualment està molt en boga per la seva flexibilitat, potència, robustesa i er ser una solució no porpietària.

JSP "Java server pages" pàgines que s'executen al servidor i que poden ser visibles des d'un navegador web.

HTML És el llenguatge per excel·lència de la part visible pels usuaris d'internet, i gairebé totes les tecnologies que hi ha al darrera acaben utilitzant els mètodes més habituals d'aquest llenguatge.

MVC "Model Vista Controlador" és un patró de capes per a dissenyar aplicacions distribuïdes i que té la particularitat de separar el codi font en tres capes. Això fa que sigui més senzill identificar els errors, i assegura millor les dades de l'aplicació.

MySql És un Sistema Gestor de Bases de Dades molt utilitzat en l'actualitat i que es porta molt bé amb el J2EE.

NetBeans Entorn integrat de desenvolupament, que ens ajuda a desenvolupar amb més rapidesa, i des d'on podem integrar aplicació amb Base de dades i els *Frameworks* amb què treballarem.

Patró En J2EE un patró és un esquema arquitectònic de les capes que tindrà l'aplicació.

Plataforma En informàtica és la tecnologia base on s'executen les aplicacions. Les plataformes més conegudes són UNIX, Linux, MAC-OS amb Apple i Microsoft amb PC.

POO "Programació Orientada a l'Objecte" és un paradigma de programació que ens permet treballar amb una filosofia molt més àmplia que la Programació Estructurada. La seva potència rau en la definició de les classes que són com motlles dels que després es crearan objectes, que podran interactuar entre ells.

Rol En informàtica són les característiques concretes d'un usuari i les funcionalitats que li pertoquen.

Struts2 *Framework* molt útil per tal de comunicar la capa vista amb la capa de negoci.

Servidor És un programa o màquina que allotja l'aplicació web i els usuaris s'hi connecten sense saber-ho des de navegadors de qualsevol punt de la xarxa Internet.

Sistema Operatiu El programa base de qualsevol ordinador. Sense ell no es pot fer absolutament res.

SQL Llenguatge de consultes per a Bases de Dades.

Taula Element d'una base de dades, que emmagatzema elements del mateix tipus amb diferents atributs heterogenis.

TFC Treball Final de Carrera.

UML "Unified Modeling Language" serveix per construir diagrames visuals de gairebé qualsevol tipus, i que en informàtica s'usa pels diagrames de classes, entre altres.

War Fitxer a partir del que desplegarem l'aplicació a una altra màquina que no sigui on hem creat l'aplicació.

Bibliografia

Programación Java Server con J2EE Edición 1.3, V.V.A.A. Anaya, Madrid, 2008.

Pàgina web oficial de NetBeans
http://netbeans.org/kb/trails/java-ee.html?utm_source=netbeans&utm_campaign=welcomepage

Consulta d'un exemple bàsic amb Struts2
http://mundogeek.net/archivos/2009/02/08/struts-2/

Entendre les diferències entre Struts1 i Struts2
http://www.roseindia.net/struts/struts2/Struts2vsStruts1.shtml

Manual molt complert d'Struts2
http://www.roseindia.net/struts/struts2/index.shtml

Vídeo explicatiu de com crear una aplicació amb NBeans + Struts2 + Hibernate
http://www.youtube.com/watch?v=z4ys0dXST94

Exemple de la volta complerta d'una entitat per totes les capes en Struts2+Hibernate
http://viralpatel.net/blogs/2010/01/tutorial-struts2-hibernate-example-eclipse.html

Exemple d'implantació de rols amb GlassFish Server
http://download.oracle.com/javaee/6/tutorial/doc/bnbxj.html#bnbxr

Instal·lació de GlassFish
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=InstalacionConfiguracionGlassfish

Manual de JSP
http://www.jsptut.com/

Lloc oficial d'Struts2
http://struts.apache.org/

Documentació oficial de JAVA i J2EE
http://download.oracle.com/javaee/index.html

Seguretat JAAS
http://download.oracle.com/javase/1.4.2/docs/guide/security/jaas/JAASRefGuide.html