# JAVA 2 PLATFORM, ENTERPRISE EDITION

# J2EE

Universitat Oberta de Catalunya
Curs 2007 – 2008 / 1er. semestre
Aula 4
**Consultor:**
Joan Vicent Orenga Serisuelo
**Alumne:**
Xavier Gusi Ballester

**Agraïments:**

Al meu fill Marc, en primer lloc, perque ha estat l'estímul constant per anar cursant mica en mica aquesta carrera, i poder-l'hi demostrar així, que el coneixement mai no està de més i que mai es tard per a aprendre coses noves. A la Comunitat UOC en general (alumnes, consultors, personal de suport, etc.) per dur a terme aquest apassionant projecte educatiu que s'ha anat obrint pas en el temps, ja que de no haver estat així, la meva fita de superar aquesta carrera universitària hauría estat del tot impossible.

## Resum del Treball Final de Carrera

En aquest treball he volgut desenvolupar una aplicació que pugui resultar d'utilitat a la empresa on treballo. Consisteix en una eina que permet, en temps real, poder saber l'estock de terminals de telèfons mòbils existents en un determinat magatzem, tenint en compte que es treballa de manera descentralitzada i les entrades i sortides de telèfons son constants.

L'anàlisi i disseny del projecte s'emmarca dins del que podriem anomenar eines basades en software lliure i es desenvolupa mitjançant el paradigma de la **Orientació a Objectes**. També està pensat per a treballar en xarxa, aprofitant el gran avantatge que representa avui dia la xarxa internet i les noves tecnologíes.

Tota la implementació i codificació és en **Java**, i per tal de donar a l'aplicació les funcionalitats requerides, he optat per utilitzar l'arquitectura **J2EE**, que permet una gran robustesa, escalabilitat i manteniment de l'aplicació.

Com a patró de disseny, bàsicament he utilitzat **hibernate**, ja que degut a la seva constant evolució i acceptació per tota la comunitat de programadors permet dotar a l'aplicació de totes les funcionalitats necessàries i alliberar de molta feina a l'hora de la implementació, així com dotar de una clara separació de funcionalitats entre els components de l'aplicació.

La part de persistència de les dades està suportada per una Base de Dades relacional com es **MySQL**, també lliure de llicències, perfectament reconeguda per tota la comunitat i suficientment robusta.

El resultat final ha estat una aplicació que aconsegueix realitzar les funcionalitats demanades i que, a més, és portable en diferents Sistemes Operatius, Servidors d'Aplicacions i Navegadors de client.

## INDEX

# 1. INTRODUCCIÓ

J2EE és una plataforma creada per SUN a l'any 1997 i, segons la opinió de molts experts, la que ofereix millors perspectives de desenvolupament per a empreses que vulguin basar la seva arquitectura en productes basats en software lliure.
HI ha molta confusió respecte del què és J2EE. El més habitual és pensar que J2EE és un producte concret que distribueix SUN juntament amb el JDK i, per tant, descarregable de forma gratuïta desde la seva pàgina web. Això no és cert, el cert és que J2EE és una especificació, un JSR (concretament el JSR-151), que defineix una plataforma empresarial, anomenada J2EE. Per a dir-ho de una manera més 'plana', J2EE és una *manera més o menys protocolaritzada de fer les coses*.

Java Platform, Enterprise Edition o Java EE és actualment el què abans anomenàvem Java 2 Platform, Enterprise Edition ò J2EE fins a la versió 1.4. És una plataforma de programació, part de la plataforma Java, per desenvolupar i executar software d'aplicacions en lenguatge Java i amb arquitectura de n nivells distribuïda, basant-se àmpliament en components de software modulars executant-se sobre un servidor d'aplicacions. La plataforma Java EE està definida per una especificació, i és considerada informalment també com un estàndar degut a què els subministradors han de cumplir certs requisits de conformitat per declarar que els seus productes son conformes a Java EE, encara que no es tracti de cap estandar de tipus ISO o ECMA.

Java EE inclou vàries especificacions de API, com JDBC, RMI, e-mail, JMS, Serveis web, XML, etc. i defineix com coordinar-los. Java EE també configura algunes especificacions úniques per Java EE per components, que inclouen Enterprise JavaBeans, servlets, portlets, Java Server Pages i diverses tecnologíes de serveis web. Tot això permet al desenvolupador crear una aplicació d'empresa portable entre plataformes i escalable, a més de integrable amb tecnologíes anteriors. Però això no és tot, hi ha altres beneficis afegits, com poden ser que el servidor d'aplicacions pugui implementar transaccions, seguretat, escalabilitat, concurrència i gestió dels components desplegats, cosa que es tradueix en una facilitat per el desenvolupador, que pot concentrar-se més en la lògica de negoci dels components que en tasques de manteniment de baix nivell.

La millor manera de veure els requisits que cumpleix una aplicació de model J2EE és veure alguns conceptes de desenvolupament d'aplicacions sota aquesta plataforma i el què seria una arquitectura típica basada en aquesta tecnología. Normalment una aplicació J2EE pot tenir cinc capes diferents:

- Capa client. Representa la interfície d'usuari amb la què interacturà.

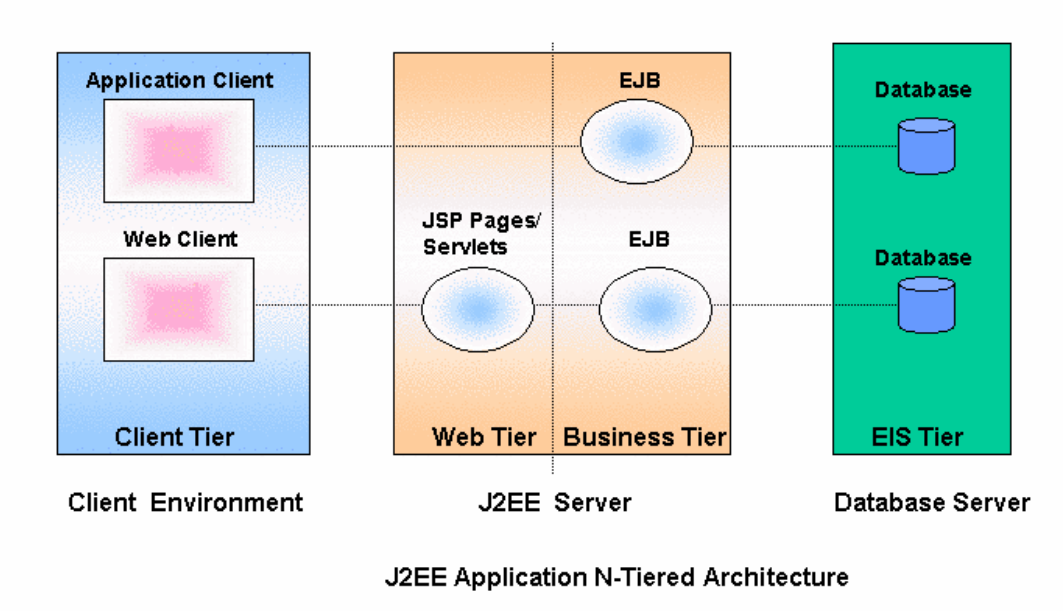
- Capa de presentació. Representen el conjunt de components que generen la informació que es representarà a la interfície d'usuari del client. Típicament això es farà a través de components basats en Servlets i JSP's.
- Capa de lògica de negoci. Conté els nostres components de negoci reutilitzables. Normalment es formen a partir de compopnents EJB.
- Capa de integració. Aquí es troben els components que ens permeten fer més transparent l'accés a la capa de sistemes de informació. Per exemple, és el lloc on es trobaràn la llògica de Objectes d'accés a dades, DAO (Data Acces Objects).
- Capa de sistemes de informació. Aquesta capa engloba els nostres sistemes de informació, per exemple, bases de dades relacionals, bases de dades orientades a objectes, sistemes legacy, etc.

Els avantatges d'un model així son molt importants, ja que al tenir les seves diferents capes separades, tenim molt poc acoplament entre elles de manera que és molt més senzill fer modificacions sense que s'interfereixin, garantint-se l'escalabilitat, el manteniment i la reutilització de components, principis bàsics de qualsevol aplicació orientada a objectes. Un altre avantatge important és la variabilitat i heterogeneitat dels diferents tipus de clients a través del dispositiu que facin servir, ja sigui per mòbil, set-top-boxes, PC, etc. En aquests casos només cla canviar la interfície d'usuari i la presentació, sense tenir que modificar la resta de capes.

Imaginem una aplicació d'empresa a la qual accedim mitjançant una interfície web, que rebi dades mitjançant formularis web, que aquestes dades s'emmagatzemen en una base de dades (o vàries) i que permeti accedir simultàniament a molts usuaris, independentment de la seva localització. Imaginem ara que amb els coneixements que tenim, hem de programar alguna cosa així. Hauríem de, no només programar i dissenyar l'evident, com les pàgines web, sinó que a més hauríem de programar les comunicacions amb les bases de dades, a més de fer possible la multitasca i tenir en compte la seguretat de tot el desenvolupament. Podríem fer tot això o utilitzar la solució adequada.

La plataforma J2EE (Java 2 Platform Enterprise Edition) i JBoss en concret, com a servidor d'aplicacions, ens permetrà oblidar-nos d'aquestes tasques menys evidents que esmentàvem abans, i centrar-nos en el disseny de la nostra base de dades, de la nostra interfície web i en concret de la nostra aplicació. De la resta, s'encarregarà el servidor d'aplicacions.

Qualsevol servidor d'aplicacions Java està compost per dues parts: un servlet engine, on s'executen les aplicacions de servidor (Java Server Pages –JSP's- i Servlets) i un EJB engine (per a aplicacions EJB's, que per exemple serviran per a comunicar-se amb la base de dades, etc...).




J2EE Application N-Tiered Architecture

Aquestes dues parts (Servlet i EJB Containers) no estan clarament identificades i separades en molts dels Servidors d'Aplicacions actuals, la qual cosa influeix negativament en el treball del programador. Jboss és una excepció.

JBoss realment és un EJB Container, que s'ha d'utilitzar al costat d'un Web Container, que podria ser qualsevol. L'avantatge és que amb Jboss ve inclòs Tomcat com a Web Container. A més, ambdós són Open Source i per tant gratuïts.

## 1.1 Justificació del TFC i punt de partida

L'objectiu principal d'aquest TFC és poder demostrar l'assoliment dels coneixements adquirits al llarg de tota la carrera, i en concret, la de Enginyeria Tècnica en Informàtica de Sistemes, analitzant un probleme real i transformant-lo en un projecte informàtic el.laborant un plà de treball, un anàlisi del problema, un disseny de la sol.lució, la seva implementació, i, finalment, obtenint una sol.lució fiable i eficient.

El punt de partida ha estat doncs, els coneixements adquirits durant tota la carrera, amb la dificultat afegida d'haver d'adquirir uns coneixements que no s'aprenen durant la carrera, com son la programació distribuïda, els patrons de treball o els components (al menys no es veuen amb la profunditat que ha fet falta després per a poder fer aquest projecte), així doncs aquest aspecte ha estat un esforç addicional important en aprendre aquestes característiques, i molt especialment les de la tecnologia escollida.

D'aquesta manera si com a punt de partida teniem els coneixements adquirits en assignatures com programació OO, enginyería del programari o tècniques de desenvolupament del programari, el punt final ha estat l'aprenentatge d'un conjunt de tecnologies relacionades amb la programació distribuïda i concretament en el desenvolupament d'aplicacions web empresarials, i això és el que desitjo que quedi de manifest en la present memòria.

## 1.2 Descripció del projecte escollit

El projecte que he escollit respòn a una necessitat detectada en el meu actual entorn laboral. Sovint a la meva feina em trobo que necessito consultar l'stock de terminals de telèfons mòbils disponibles per a poder oferir a futurs clients que decideixen contractar els seus serveis de telefonía mòbil amb la meva empresa.

Actualment aquest stock s'ha de consultar per separat, desde diferents aplicacions:

- Fitxer excel amb la relació de marca i model de terminal, així com el nombre d'unitats disponibles en els diferents magatzems.
- Fitxer power point amb les imatges de cada un dels terminals disponibles, així com una breu descripció de les seves característiques principals.
- Configurador en format web per a poder confeccionar la oferta al client.

## 1.3 Funcionalitats que ha de tenir el projecte

En primer lloc, el projecte es desenvoluparà en format web i seguint l'estàndar J2EE, dincs del què podríem anomenar *MVC*[1], de manera que la seva consulta es pugui fer '*en línia*' aprofitant les noves tecnologíes i l'accés a internet.

El projecte ha de poder donar a l'usuari una visió molt ràpida dels terminals disponibles, tant bon punt es carrega l'aplicació. La informació que rebrà l'usuari serà:

- Imatge reduïda del terminal.
- Marca i model.
- Nombre d'unitats.
- Magatzem on es troba el terminal.

A més, cada imatge representarà un *link* cap a la pàgina del fabricant on es podràn consultar totes les característiques detallades del terminal, així com imprimir-ne una foto. Prèviament però, l'usuari s'haurà d'identificar amb el seu id i password, de manera que l'aplicació el pugui 'reconèixer' i donar-li a aquest els privilegis adients.

## 1.4 Perfils d'usuari

Es definiràn dos perfils d'usuari:

- Usuari administrador. Tindrà privilegis, a més de fer les consultes normals, de afegir nous terminals a la base de dades i de suprimir-ne. També podrà donar d'alta nous usuaris de consulta.
- Usuari de consulta. Només podrà consultar els terminals disponibles als diferents magatzems.

1

[1] Model Vista Controlador és un patró d'arquitectura software que separa les dades de una aplicació, la interfície d'usuari i la lògica de control en 3 components diferents. El patró MVC es veu sovint en aplicacions web, ón la vista és la pàgina HTML i el codi que proporciona dades dinàmiques a la pàgina amb un model de Sistema de Gestió de Bases de Dades (SGBD), i el controlador representa la lògica de negoci.

## 2. PLA DE TREBALL

Tal com he descrit abans, el projecte s'emmarca dins dins del context d'aplicació J2EE i l'objectiu principal és assolir els coneixements necessaris per arrivar a comprendre aquest model d'aplicacions. Per això, s'hauràn de seguir les següents pautes:

- Detectar les necessitats de software a satisfer.
- Dissenyar el model d'aplicació dins de l'entorn MVC.
- Definir les diferents parts que el composen.
- Elaborar els diagrames d'ús del projecte.
- Programar l'aplicació seguint el model MVC i tenint molt en compte les futures tasques de manteniment de l'eina i de la seva reutilització en altres aplicacions.

Per tal de dur a terme aquest treball, s'han de planificar molt bé les tasques a realitzar, que es poden resumir en 3 etapes ben diferenciades:

- Anàlisi de necessitats. Son les què s'inclouen en aquest document, i es tracta de captar les necessitats de l'usuari que vol una eina que li permeti, de manera simplificada, obtenir ràpidament l'stock i caracteristiques principals dels telèfons mòbils disponibles.
- Disseny. Un cop concretades les necessitats de l'usuari, es tracta de dissenyar les possibles solucions. En aquesta fase es representaràn els diferents diagrames (de casos, d´ús, etc.) , el disseny de la base de dades, el model MVC i determinar un 'esquelet' de les diferents vistes que tindrà l'usuari.
- Implementació. En aquesta etapa es duurà a terme la programació pròpiament dita de tota l'aplicació.
- Publicació final de tot el treball.

Per tal d'anar seguint totes aquestes pautes, i d'acord amb la planificació demanada per els consultors de l'assignatura, segons el Plà Docent de la mateixa, he establert la següent taula per relacionar les diferents etapes definides abans amb les entregues parcials que es demanen:

| Etapa | Entrega |
|---|---|
| *Anàlisi de necessitats* | *PAC1* |
| *Disseny* | *PAC2* |
| *Implementació* | *PAC3* |
| *Publicació final* | *Memòria del TFC* |

## 2.1 Dates clau

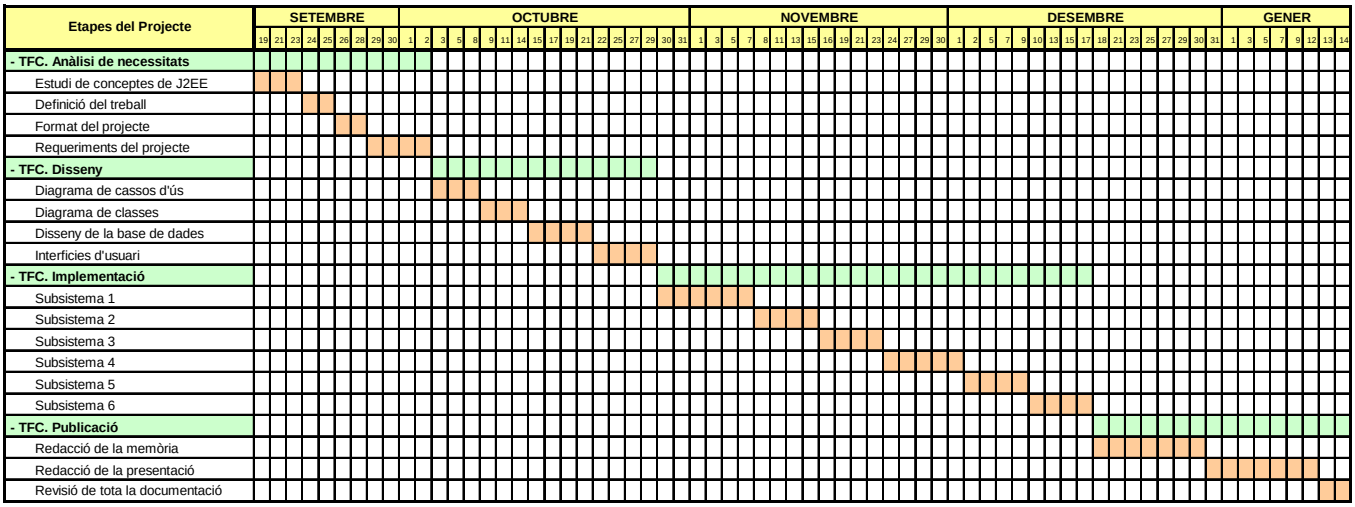
Un cop definides les entregues i el contingut de les mateixes posaré les dates clau de cada una d'elles, segons el Plà Docent de l'assignatura.

| *Títol* | *Data d'entrega* |
|---|---|
| *PAC1 (Plà de treball)* | *28/09/2007* |
| *PAC2* | *29/10/2007* |
| *PAC3* | *17/12/2007* |
| *Entrega final* | *14/01/2008* |

## 2.2 Calendari del projecte

Segons la planificació de tot el projecte, i un cop vistes les dates clau del Plà Docent, faré la següent descomposició en dates de cada una de les subtasques del Plà de Treball.

| Etapes del Projecte | Durada | Inici | Fí |
|---|---|---|---|
| **- TFC. Anàlisi de necessitats** | **14** | **19/09/2007** | **02/10/2007** |
| Estudi de conceptes de J2EE | 5 | 19/09/2007 | 23/09/2007 |
| Definició del treball | 2 | 24/09/2007 | 25/09/2007 |
| Format del projecte | 3 | 26/09/2007 | 28/09/2007 |
| Requeriments del projecte | 3 | 29/09/2007 | 02/10/2007 |
| **- TFC. Disseny** | **27** | **03/10/2007** | **29/10/2007** |
| Diagrama de cassos d'ús | 6 | 03/10/2007 | 08/10/2007 |
| Diagrama de classes | 6 | 09/10/2007 | 14/10/2007 |
| Disseny de la base de dades | 7 | 15/10/2007 | 21/10/2007 |
| Interficies d'usuari | 8 | 22/10/2007 | 29/10/2007 |
| **- TFC. Implementació** | **49** | **30/10/2007** | **17/12/2007** |
| Subsistema 1 | 9 | 30/10/2007 | 07/11/2007 |
| Subsistema 2 | 8 | 08/11/2007 | 15/11/2007 |
| Subsistema 3 | 8 | 16/11/2007 | 23/11/2007 |
| Subsistema 4 | 8 | 24/11/2007 | 01/12/2007 |
| Subsistema 5 | 8 | 02/12/2007 | 09/12/2207 |
| Subsistema 6 | 8 | 10/12/2007 | 17/12/2007 |
| **- TFC. Publicació** | **27** | **18/12/2007** | **14/01/2008** |
| Redacció de la memòria | 13 | 18/12/2007 | 30/12/2007 |
| Redacció de la presentació | 13 | 31/12/2007 | 12/01/2008 |
| Revisió de tota la documentació | 1 | 13/01/2008 | 14/01/2008 |


Etapes del Projecte
SETEMBRE
OCTUBRE
NOVEMBRE
DESEMBRE
GENER
- TFC. Anàlisi de necessitats
Estudi de conceptes de J2EE
Definició del treball
Format del projecte
Requeriments del projecte
- TFC. Disseny
Diagrama de cassos d'ús
Diagrama de classes
Disseny de la base de dades
Interficies d'usuari
- TFC. Implementació
Subsistema 1
Subsistema 2
Subsistema 3
Subsistema 4
Subsistema 5
Subsistema 6
- TFC. Publicació
Redacció de la memòria
Redacció de la presentació
Revisió de tota la documentació

# 3. DESCRIPCIÓ DEL MODEL DE NEGOCI

## 3.1 Introducció

L'empresa Telefónica de España (en endavant TdE) actualment està en fase de fusió/convergència amb Telefónica Moviles (en endavant Movistar). Degut a aquest fet, ha de incorporar al seu portfoli de productes per a empreses i particulars un nou negoci, el de la telefonía mòbil, en certa manera desconegut fins ara. En una primera fase, TdE funcionarà com si es tractés de un altre distribuïdor de Movistar, amb un estock de terminals molt concret que no té perquè coincidir amb el de qualsevol altre distribuïdor.

Segons campanyes promocionals puntuals, ofertes de fabricants, descatalogació de productes, renovació tecnològica, etc. es fa necessari la implementació de una eina que faci possible saber en tot moment l'estock de terminals existents en cada un dels magatzems que existeixen, que son en concret el de Canàries i el magatzem central, ja que s'han de poder admetre comandes d'arreu d'Espanya i aquestes s'han de fer amb el terminal escollit, que és el que s'ajusta a la campanya o oferta en concret. Val a dir que les comandes de la península s'han de fer tenint en compte l'estock de terminals del magatzem central, i les comandes fetes desde les Illes Canàries, s'han de fer tenint en compte l'estock de terminals del magatzem de canàries, i en cap cas es pot escollir un terminal del magatzem que no correspongui, principalment per motius fiscals en l'aplicació de l'impost corresponent.

## 3.2 Situació actual

Actualment el control de l'estock de terminals es fa mitjançant un fitxer excel que es penja de manera més o menys periòdica en una aplicació interna de l'àrea de Màrqueting, on s'indica el tipus de terminal, marca, model, etc. Per tal de veure algunes de les característiques dels terminals s'ha de consultar un altre fitxer en format power point on hi ha els diferents terminals catalogats en funció de la gama (alta, mitjana o baixa) i on es poden veure, a part de la foto, caracteriístiques globals com poden ser el tipus de càmera de fotos que porta, la inclusió de bluetooh, etc. Sovint no coincideix la informació que consta en un i altre fitxers, i, també sovint, la informació de l'estock dels terminals no és real perquè pot haver un fitxer amb l'estock de fa 3 dies.

Aquestes dades son necessàries per fer la comanda perquè s'ha de indicar clarament en aquest moment quin tipus de terminal ha escollit el client, quina campanya se li aplica, i quin cost final tindrà tota la operació, de manera que passa algún cop que, un vegada s'ha decidit el client per un tipus de terminal, i donada la volatilitat de l'estock, i una vegada s'ha cursat la comanda, en el moment de

gravar-la a l'aplicació de Movistar ens trobem que el terminal escollit no existeix, o està esgotat o bé s'ha descatalogat, cosa que provoca tornar a començar amb tota la tramitació de l'expedient.

## 3.3 Sol.lució proposada

La sol.lució proposada objecte d'aquesta memòria es basa en una aplicació que en temps real i mitjançant una consulta a una Base de Dades (que en aquest cas s'ha fet en MySQL) doni una visió molt ràpida de l'estock dels terminals en cada un dels 2 magatzems existents, quan un usuari en vulgui fer la consulta. També ha de donar la facilitat al/s administrador/s de l'aplicació de donar d'alta o baixa terminals que s'incorporin a un determinat magatzem o que es donin de baixa per el motiu que sigui.

La idea és que sigui relativament fàcil incloure o excloure terminals de la base de dades de manera que la informació que hi tingui sigui la real en el moment en què es consulta, ja que es podrà interactuar tant si l'àrea de màrqueting inclou un nou terminal, com si el magatzem serveix una comanda i ha de rebaixar l'estock o reb una remesa de terminals.

Amb aquesta sol.lució, aquests requeriments queden àmpliament coberts degut a els avantatges de la base de dades (MySQL) que admet múltiples sesions d'usuari, la tecnología Java per a assegurar-ne l'accés a la mateixa i les finestres en format web, que tenen la facilitat que els diferents usuaris no han de instal.lar cap tipus de programari per accedir a l'aplicació.

## 3.4 Recursos de software

Per a la implementació del projecte, s'ha fet servir el següent software i versió:

- IDE de programació: *Eclipse Europa*
- Framework: *Hibernate 3.2.5*
- SGBD: *MySQL 5.0*
- Servidor d'aplicacions: *JBOSS 4.0.5.GA*
- Driver JDBC: *mysql-connector-java-5.0.7*
- Java: *jdk1.5.0_12*
- *Apache-ANT-1.7.0*
- *Macromedia Dreamweaver UltraDev4*

## 3.5 Productes obtinguts

Bàsicament, el producte obtingut amb aquest projecte ha estat una aplicació web realitzada amb tecnología J2EE.

Es descompon en els següents elements:

- Fitxer de distribució .war, que inclou les capes web, les classes compilades i resta de fitxers de utilitat a l'aplicació.
- Fitxer .jar amb les classes compilades.
- El codi font de totes les classes.
- Documentació detallada i normalitzada en format javadoc de totes les classes.
- Els fitxers de creació de la base de dades, script.
- Diferents guies de instal.lació i descripció de funcionament.

# 4. OBJECTIUS

El principal objetiu d'aquest projecte és facilitar al màxim tot el procés previ a la comanda formal de terminals mòbils en el cas que un client vulgui fer una portabilitat o demanar una alta nova, ja que ara mateix existeix el risc de que una comanda no progressi, principalment per manca d'estock de terminals o descatalogació dels mateixos.

Un altre objectiu és que la informació estigui disponible de una manera clara, ràpida i en una mateixa aplicació.
A nivell de Treball de Fí de Carrera (o acadèmic), l'objectiu principal és:

- Aprendre bé com funciona l'especificació J2EE.
- Assimilar el concepte de MVC.
- Definir una correcta separació entre capes.
- El seu entorn de treball.
- La seva implementació.
- La interactuació amb la Base de Dades.
- Com es fa la persistència dels objectes que hi ha.
- Com es fa la interactuació per part dels clients

En definitiva, implementar i construïr una eina informàtica seguint el model MVC ( *model vista controlador*) i utilitzant per aquest fet software lliure de llicències gràcies a desenvolupadors com Sun Microsystems o MySQL.

Una vegada fet aquest projecte crec que els objectius bàsics esmentats han quedat perfectament assolits, ja que recull tot un conjunt de problemàtiques d'implementació que son comuns a la majoría d'aplicacions que d'alguna manera gestionen bases de dades, principalment altes, baixes, consultes i modificacions.

## 4.1 Funcionalitats no implementades

Una aplicació d'aquest tipus pot requerir molt de temps per a poder recullir qualsevol casuïstica que pugui necessitar l'usuari final, fins i tot un cop posada en marxa poden surgir necessitats no detectades fins a aquest moment, es per això que he intentat implementar les funcions més bàsiques, donanda la manca de temps de que es disposa amb un semestre acadèmic. La idea ha estat fer la eina bàsica, a partir d'aquest punt es pot anar creixent a base de anar afegint funcionalitats, cosa que és un altre avantatge de fer servir la tecnologia J2EE, la seva escalabilitat i el posterior manteniment. Algunes de les funcionalitats que es podrien afegir podrien ser:

- Afegir més filtres per a poder acotar molt més la cerca de terminals segons diferents característiques (gama, tipus de càmera, bluetooh, tipus de teclat, tipus de banda, etc.

- Afegir la funció d'extreure llistats en format pdf o excel, per a poder obtenir una llista de manera ràpida i poder-l'hi presentar al client.
- Afegir la funcionalitat de poder configurar la oferta a mida del client.
- Afegir la funcionalitat de poder imprimir els diferents tipus de contracte que ha de signar el client.
- Modificar la fase de *login* a l'aplicació de manera que es faci servir les *acegui security* en comptes de fer servir una taula amb la relació de usuaris donats d'alta.

# 5. DISSENY

L'arquitectura J2EE permet la creació de components d'aplicacions distribuides escalables, transaccionals i multiusuari.

Un *Enterprise JavaBean* és un component de la part servidora, gestionat pel contenidor i pensat per la construcció modular d'aplicacions d'empresa. L'especificació EJB és una de les moltes APIs de Java de la Plataforma Java Enterprise Edition. EJB encapsula la lògica de negoci d'una aplicació. L'especificació EJB va ser desenvolupada inicialment el 1997 per IBM i adoptada posteriorment per Sun Microsystems (EJB 1.0 i EJB 1.1) i millorada sota la Java Community Process com JSR 19 (EJB 2.0), JSR 153 (EJB 2.1) i JSR 220 (EJB 3.0). L'especificació EJB pretèn aportar una manera estàndard d'implementar el codi de negoci del back-end, típicament trobat a les aplicacions d'empresa (en oposició al codi d'interfície d'usuari anomenat front-end). Aquest codi es trobava molt sovint resolent els mateixos tipus de problemes i es va trobar que solucions a aquests problemes es programaven de forma repetida pels programadors. Els Enterprise JavaBeans intentaven gestionar qüestions tan habituals com la persistència, integritat transaccional i seguretat, de manera estàndard, alliberant els programadors perquè es concentressin en el problema que els ocupava realment.

Els clients no podran interactuar directament amb els EJB d'entitat, sino que ho faran mitjançant un EJB de sessió sense estat que concentrarà la lògica de negoci a la qual podran accedir els clients.

## 5.1 Tipus d'EJB's

Un contenidor d'EJB pot representar tres categories de beans principals:

- Session Beans (beans de sessió).
- Stateless Session Beans (sense estat).
- Stateful Session Beans (amb estat).
- Entity Beans (beans d'entitat).
- Message Driven Beans (MDBs o Message Beans)

*Stateless Session Beans* (beans de sessió sense estat) són objectes distribuïts que no tenen estat associat a ells, de manera que s'hi pot accedir concurrentment. Els continguts de variables de cada instància no garanteixen ser preservades entre diferents crides. La manca de càrrega addicional per mantenir una conversa amb el programa cridant els fa menys intensius en recursos que els beans amb estat.

*Stateful Session Beans* (beans de sessió amb estat) són objectes distribuïts que tenen estat, es mantenen al corrent de quin programa que els crida és el que estan tractant durant una sessió. Per exemple, la comprovació en una botiga web podria ser gestionada per un bean de sessió amb estat, que usaria el seu estat per estar al corrent de quin client és el que està al procés de comprovació. Per altra banda,

enviar un e-mail a atenció al client podria ser gestionat per un bean sense estat, donat que és una operació puntual i no és part d'un procés de diferents passes. L'estat dels beans de sessió amb estat hauria de ser persistent, però l'accés a la instància del bean està limitat només a un client.

*Entity Beans* són objectes distribuïts que tenen un estat persistent. L'estat persistent pot ser gestionat pel propi bean o no ser-ho. En el primer cas, els beans són *Bean-Managed Persistence* (BMP). En el segon cas, on és el contenidor qui se'n cuida de la seva persistència, són *Container-Managed Persistence* (CMP). Amb els EJB d'entitat podem representar objectes de negoci a la capa de persistència, implementant els mapejos objecte-relació entre els objectes java i les taules de la base de dades. A aquest projecte farem servir CMP (Persistència gestionada pel Contenidor), delegant al contenidor totes les tasques de gestió de les dades a les taules corresponents, i estalviant així molta feina de programació.

*Message Driven Beans* són objectes distribuïts que es comporten de manera asíncrona. És a dir, gestionen operacions que no requereixen resposta immediata. Per exemple, un usuari d'un lloc web que clica el botó "mantingueu-me informat de futures actualitzacions" podria disparar una crida a un Message Driven Bean per afegir l'usuari a una llista a la base de dades de l'empresa. Aquesta crida és asíncrona, ja que l'usuari no necessita esperar a ser informat del seu bon o mal funcionament). Aquest beans se subscriuen a cues de missatges JMS (Java Message Service) o temes que envien missatges. Els MDB van ser afegits a l'especificació 2.0 per permetre processos dirigits per events dintre dels contenidors EJB. A diferència d'altres tipus de beans, els MDB no tenen una perspectiva de client (interfícies Remote/Home), és a dir, els clients no poden visitar una instància d'MDB. Simplement escolten qualsevol missatge entrant d'una cua JMS (o tema) i els processa automàticament.

Hi ha proposats altres tipus d'Enterprise Beans. Per exemple, *Enterprise Media Beans* (JSR 86) pensats per la integració d'objectes multimèdia en aplicacions J2EE.

## 5.2 Evolució dels EJB's

Gradualment un consens en la indústria va anar fent emergir que la virtut primera dels EJBs -habilitar integritat transaccional sobre aplicacions distribuïdes- tenia l'ús limitat per la majoria de les aplicacions empresarials. La funcionalitat oferta per entorns més senzills com Spring i Hibernate era més útil per aplicacions d'empresa. D'acord amb això, l'especificació EJB 3.0 (JSR 220) era un punt de partida radical respecte els seus predecessors, seguint aquest nou paradigma. Aquí s'aprecia una clara influència respecte Spring, el seu ús de POJOs i el seu suport a la injecció de dependència per simplificar la configuració i la integració de sistemes heterogenis. Gavin King, el creador d'Hibernate, va participar en el procés d'EJB 3.0 i se'l considera una veu lliure en defensa d'aquesta tecnologia. Moltes funcionalitats d'Hibernate van ser incorporades a la Java Persistence API, el substitut dels Entity Beans a EJB 3.0. L'especificació dels EJB 3.0 usa fortament les anotacions (meta

dades), una funcionalitat afegida al llenguatge Java en la seva versió 5.0, per habilitar un estil de programació molt menys farragós.
D'acord amb això, en termes pràctics, EJB 3.0 és bastant una API completament nova, semblant-se molt poc a les especificacions prèvies d'EJB.

Per tal de dissenyar aplicacions MVC de manera que segueixin l'especificació J2EE, el desenvolupador té al seu abast diferents *frameworks* en el mercat, com puguin ser: *struts, springs, OpenXava, hibernate, etc.*

## 5.3 Framework utilitzat en el projecte

Per aquest projecte m'he decantat per fer servir el framework **Hibernate**, i més concretament la seva versió **3.2.5**. Hibernate és una solució implementada pel mapeig d'objectes relacionals (ORM) per aplicacions Java, sobre una base de dades relacional. Els seus propòsits bàsics són els d'alliberar el programador d'un seguit de tasques pròpies de la persistència de dades relacionals i dotar les aplicacions de portabilitat entre SGBDs diferents. Hibernate és lliure, de codi obert i està distribuït sota la GNU Lesser General Public License.

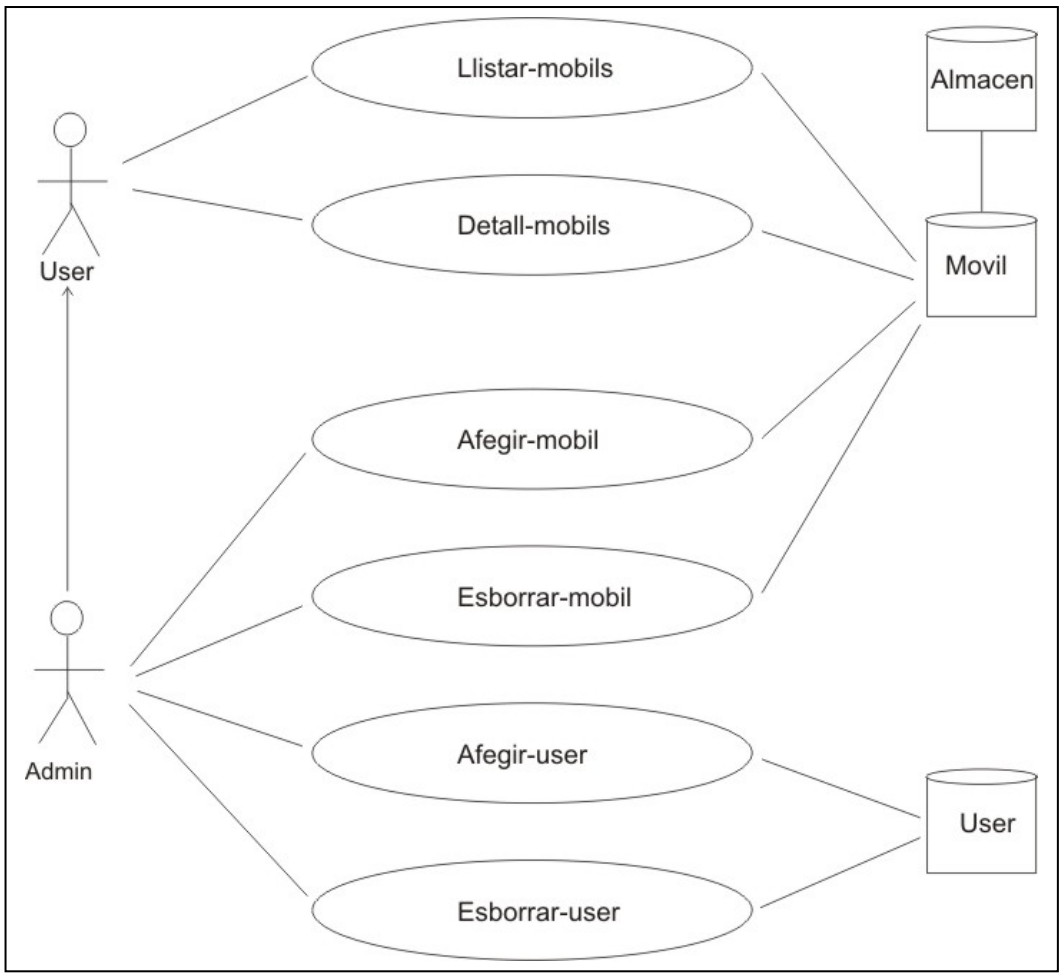
La funcionalitat bàsica d'Hibernate és la del mapeig de classes Java en taules de Base de Dades i tipus de dades Java sobre tipus de dades d'SQL. Hibernate també proveeix un llenguatge de query (HQL) i facilitats per la recuperació de dades. Hibernate genera les crides SQL i delega al desenvolupador la gestió manual del resultat de la query i la conversió a objectes. Gràcies a això una aplicació pot ser portable a la majoria de bases de dades SQL, amb mínima càrrega addicional. El requeriment necessari perquè una aplicació basada en hibernate pugui usar una base dades SQL és que existeixi la peça de software (driver) encarregada de la traducció d'HQL al dialecte SQL del SGBD sobre el que ha de còrrer l'aplicació.
Hibernate proveeix persistència per "Plain Old Java Objects"; l'únic requeriment estricte per una classe persistent és el d'oferir un constructor public i sense arguments. Hibernate pot ser usat tant en aplicacions standalone com en aplicacions Java EE que usen servlets o EJB session beans.

Aquesta versió utilitzada de hibernate permet també el control de les sessions (obrir, tancar, obtenir, etc.) amb la qual cosa obtenim, amb un sol framework, tota la funcionalitat repetitiva de la qual el programador no s'en ha de preocupar.

Hibernate va ser desenvolupat per un equip de programadors Java d'arreu del món, liderats per Gavin King. JBoss Inc. (posteriorment adquirit per Red Hat) va contractar més tard el nucli de l'equip de desenvolupadors i van treballar-hi donant suport a Hibernate. L'actual versió d'Hibernate és la 3.x. Aquesta versió té característiques noves, com ara una nova arquitectura d'Interceptor/Callback, filtres definits per l'usuari i JDK 5.0 Annotations (funcionalitat de metadades de Java). Hibernate 3 també és molt proper a l'especificació d'EJB 3.0 i serveix com la columna vertebral de la implementació d'EJB 3.0 sobre JBoss.

## 5.4 Diagrames

- **Diagrama de cassos d'ús**




Veiem que hi hauràn dos tipus diferenciats d'usuari, l'usuari normal i l'administrador, que, a més de poder fer tot el què pot fer un usuari normal, tindrà les funcions d'afegir i esborrar mòbils i usuaris.

### Especificació textual dels cassos d'ús

| **Nom del Cas d'Ús** : | Llistar mòbils |
|---|---|
| **Autor** | Xavier Gusi Ballester |
| **Descripció** | Cas d'ús que llista els móbils disponibles |
| **Actor/és** | Admin, User |
| **Precondicions:** | Els mòbils que es llistaràn existeixen a la Base de Dades. |
| **Postcondicions** | L'actor obté el llistat dels móbils que volia veure de un magatzem en concret. |
| **Descripció** | **L'Administrador** selecciona el botó corresponent a la opció 'Ver estock de terminales'. Ha d'escollir també el magatzem del qual vol obtenir el llistat, ja que les quantitas i els tipus de mòbil poden ser diferents.<br>**L'user** farà la mateixa operació però desde una finestra diferent. |
| **Casos d'ús relacionats (extensió)** | |
| **Casos d'Ús relacionats (inclosos)** | *Entrar al sistema* |

| **Nom del Cas d'Ús** : | Detall mòbils |
|---|---|
| **Autor** | Xavier Gusi Ballester |
| **Descripció** | Cas d'ús que llista els móbils disponibles |
| **Actor/és** | Admin, User |
| **Precondicions:** | Els mòbils que es llistaràn existeixen a la Base de Dades. |
| **Postcondicions** | L'actor obté una taula amb les imatges en petit de tots els mòbils que abans s'han llistat aixi com la quantitat dels mateixos. |
| **Descripció** | **L'Administrador** i **l'user** selecciona el botó corresponent a la opció 'Ver imágenes en miniatura'. |
| **Casos d'ús relacionats (extensió)** | |
| **Casos d'Ús relacionats (inclosos)** | *Entrar al sistema* |

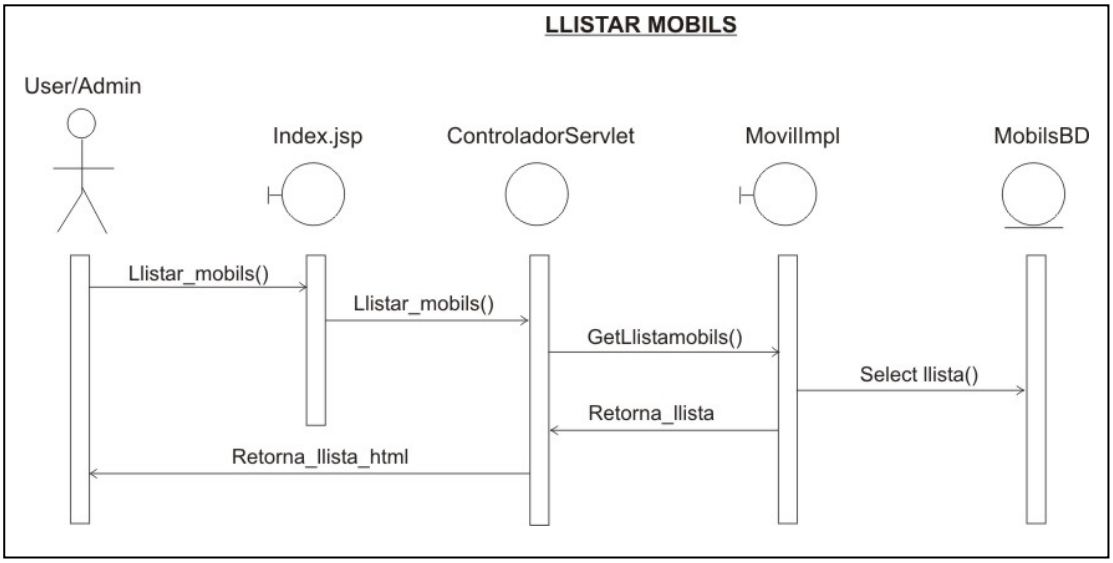
| **Nom del Cas d'Ús** : | Afegir mòbil |
|---|---|
| **Autor** | Xavier Gusi Ballester |
| **Descripció** | Cas d'ús que dona d'alta un mòbil. |
| **Actor / és** | Admin |
| **Precondicions:** | El mòbil que es vol donar d'alta no ha d'existir a la Base de Dades, per tant, abans es farà una cerca. En cas que el mòbil ja existeixi, se li informarà de tal fet mitjançant un missatge. |
| **Postcondicions** | L'actor obté un missatge que l'informa si la operació ha tingut èxit. |
| **Descripció** | **L'Administrador** selecciona la opció corresponent de 'Gestion de terminales' i selecciona prèviament 'Alta de nuevo terminal'. Després veurà un formulari on se li demanen les dades que calen per tal de donar d'alta un nou mòbil. |
| **Casos d'ús relacionats (extensió)** | |
| **Casos d'Ús relacionats (inclosos)** | *Entrar al sistema* |

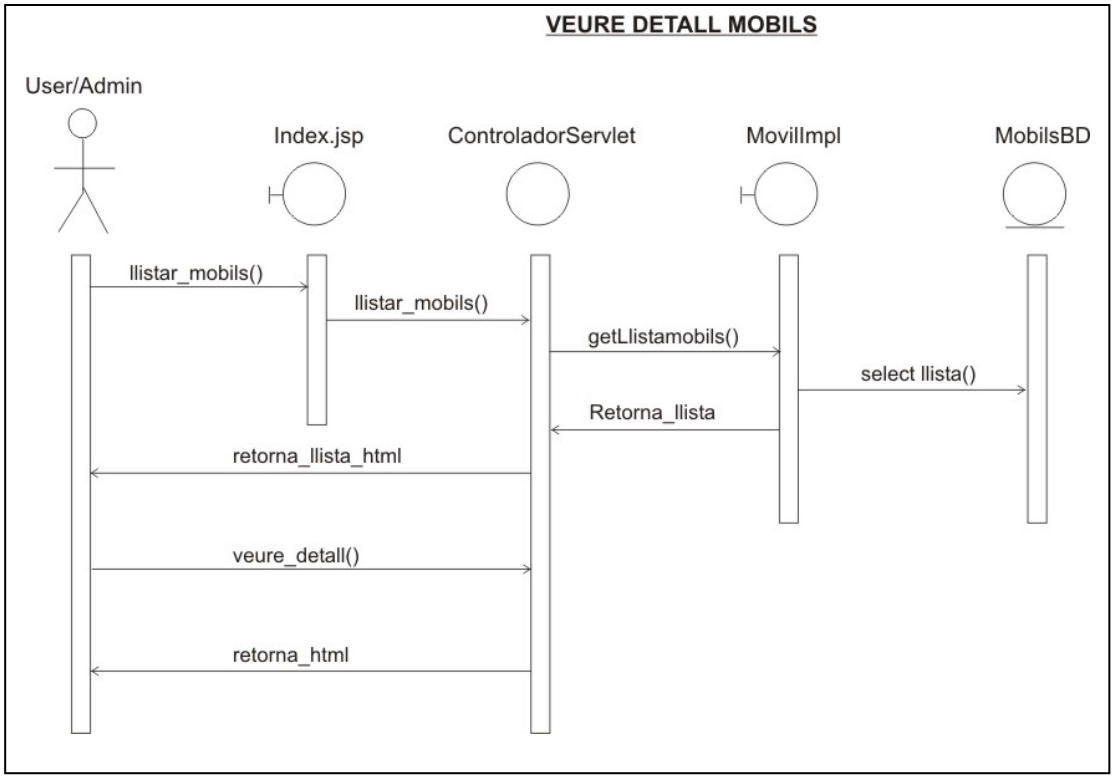
| **Nom del Cas d'Ús** : | Esborrar mòbil |
|---|---|
| **Autor** | Xavier Gusi Ballester |
| **Descripció** | Cas d'ús que dona de baixa un mòbil. |
| **Actor / és** | Admin |
| **Precondicions:** | El mòbil que es vol donar de baixa ha d'existir a la Base de Dades, per tant, abans es farà una cerca. En cas que el mòbil ja no existeixi, se li informarà de tal fet mitjançant un missatge. |
| **Postcondicions** | L'actor obté un missatge que l'informa si la operació ha tingut èxit. |
| **Descripció** | **L'Administrador** selecciona la opció corresponent de 'Gestion de terminales' i selecciona prèviament 'Baja de terminal'. Després veurà un formulari on se li demana el id del mòbil que es vol donar de baixa. |
| **Casos d'ús relacionats (extensió)** | |
| **Casos d'Ús relacionats (inclosos)** | *Entrar al sistema* |

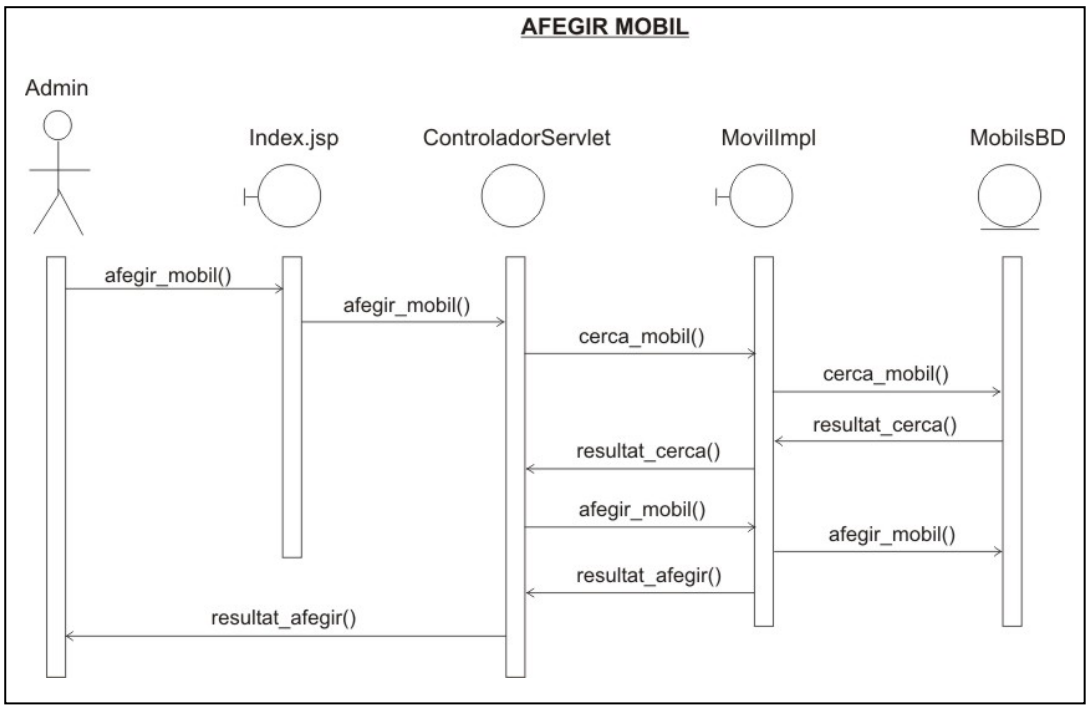
| **Nom del Cas d'Ús** : | Afegir user |
|---|---|
| **Autor** | Xavier Gusi Ballester |
| **Descripció** | Cas d'ús que dona d'alta un usuari. |
| **Actor / és** | Admin |
| **Precondicions:** | El usuari que es vol donar d'alta no ha d'existir a la Base de Dades, per tant, abans es farà una cerca. En cas que el usuari ja existeixi, se li informarà de tal fet mitjançant un missatge. |
| **Postcondicions** | L'actor obté un missatge que l'informa si la operació ha tingut èxit. |
| **Descripció** | **L'Administrador** selecciona la opció corresponent de 'Gestion de usuarios' i selecciona prèviament 'Alta de nuevo usuario'. Després veurà un formulari on se li demanen les dades que calen per tal de donar d'alta un nou usuari. |
| **Casos d'ús relacionats (extensió)** | |
| **Casos d'Ús relacionats (inclosos)** | *Entrar al sistema* |

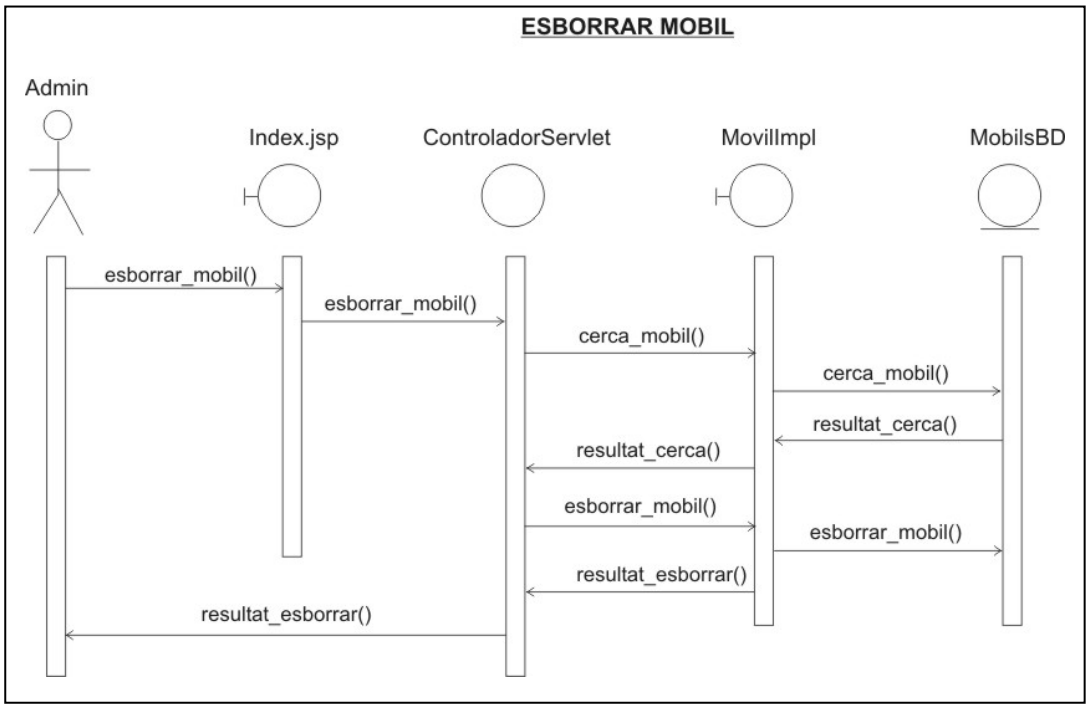
| **Nom del Cas d'Ús** : | Esborrar user |
|---|---|
| **Autor** | Xavier Gusi Ballester |
| **Descripció** | Cas d'ús que dona de baixa un usuari. |
| **Actor / és** | Admin |
| **Precondicions:** | El usuari que es vol donar de baixa ha d'existir a la Base de Dades, per tant, abans es farà una cerca. En cas que el usuari ja no existeixi, se li informarà de tal fet mitjançant un missatge. |
| **Postcondicions** | L'actor obté un missatge que l'informa si la operació ha tingut èxit. |
| **Descripció** | **L'Administrador** selecciona la opció corresponent de 'Gestion de usuarios' i selecciona prèviament 'Baja de usuario'. Després veurà un formulari on se li demana el id del usuari que es vol donar de baixa. |
| **Casos d'ús relacionats (extensió)** | |
| **Casos d'Ús relacionats (inclosos)** | *Entrar al sistema* |

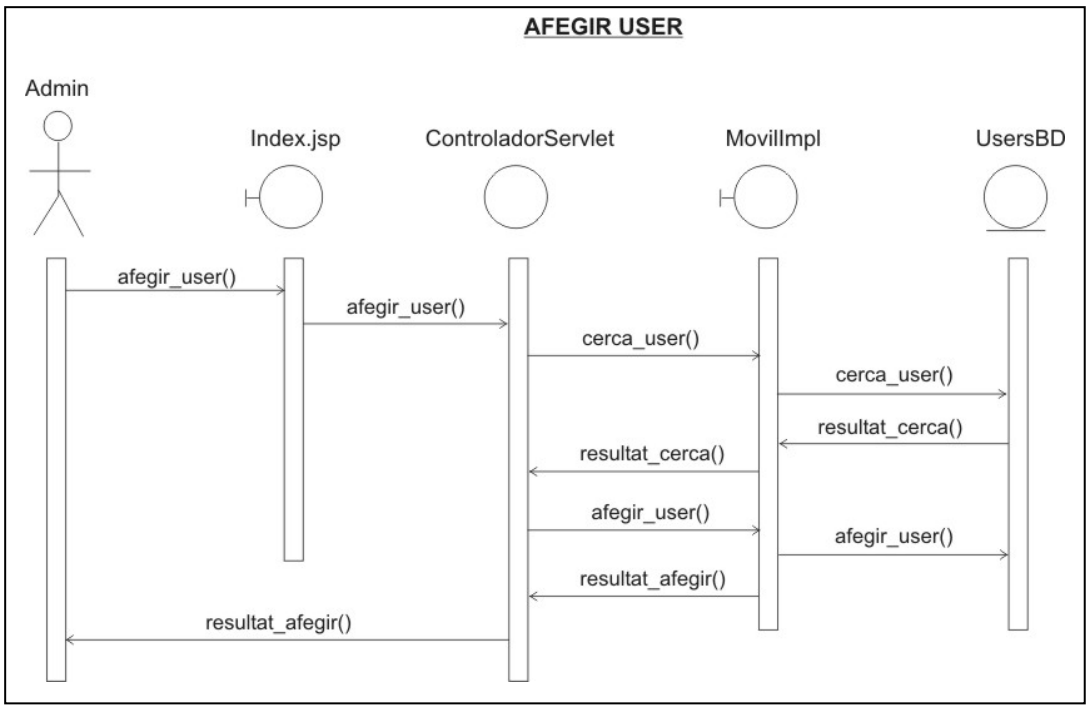
- **Diagrames de sequències**

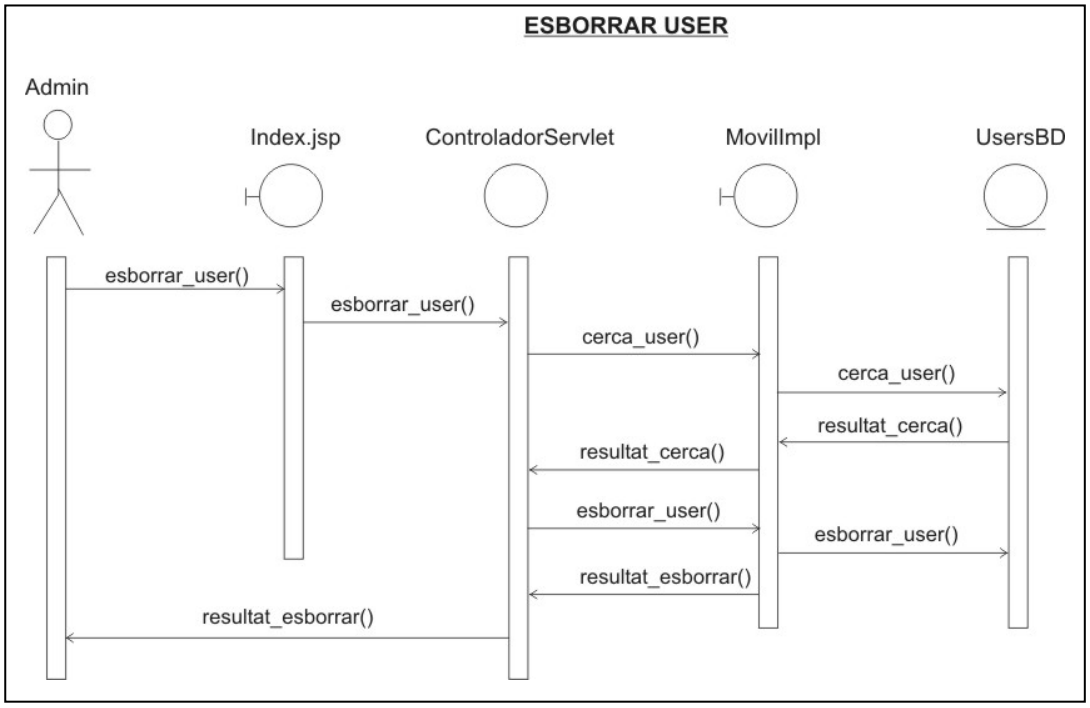


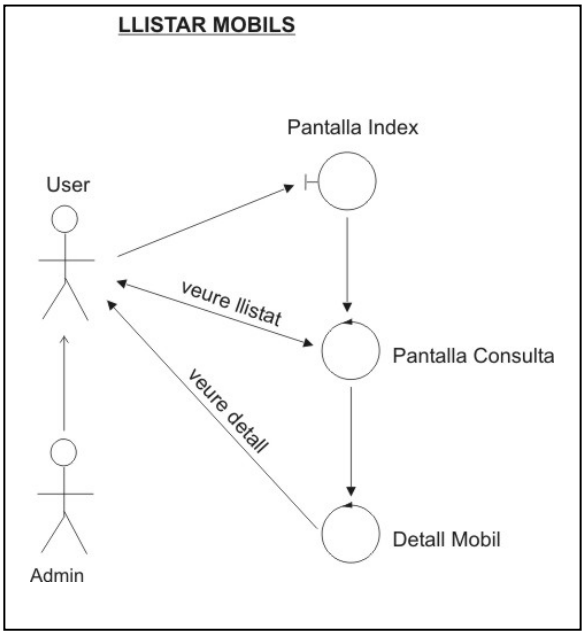

AFEGIR MOBIL
Admin
Index.jsp
ControladorServlet
MovilImpl
MobilsBD
afegir_mobil()
afegir_mobil()
cerca_mobil()
cerca_mobil()
resultat_cerca()
resultat_cerca()
afegir_mobil()
afegir_mobil()
resultat_afegir()
resultat_afegir()

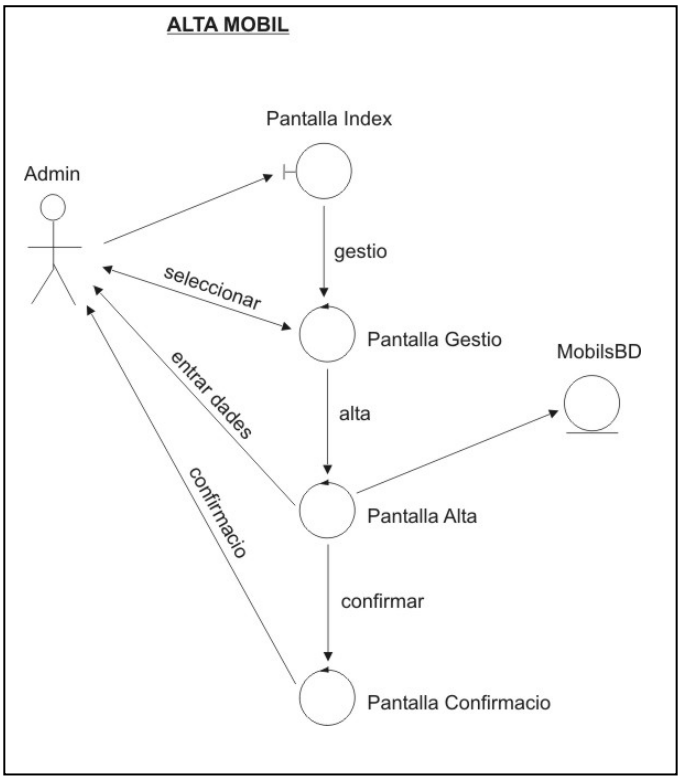

ESBORRAR MOBIL
Admin
Index.jsp
ControladorServlet
MovilImpl
MobilsBD
esborrar_mobil()
esborrar_mobil()
cerca_mobil()
cerca_mobil()
resultat_cerca()
resultat_cerca()
esborrar_mobil()
esborrar_mobil()
resultat_esborrar()
resultat_esborrar()

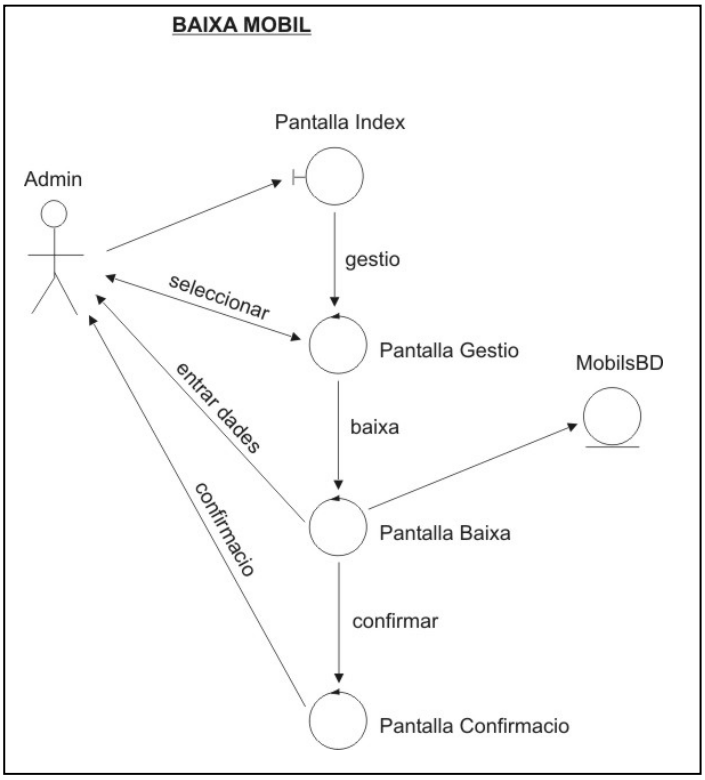

AFEGIR USER
Admin
Index.jsp
ControladorServlet
MovilImpl
UsersBD
afegir_user()
afegir_user()
cerca_user()
cerca_user()
resultat_cerca()
resultat_cerca()
afegir_user()
afegir_user()
resultat_afegir()
resultat_afegir()

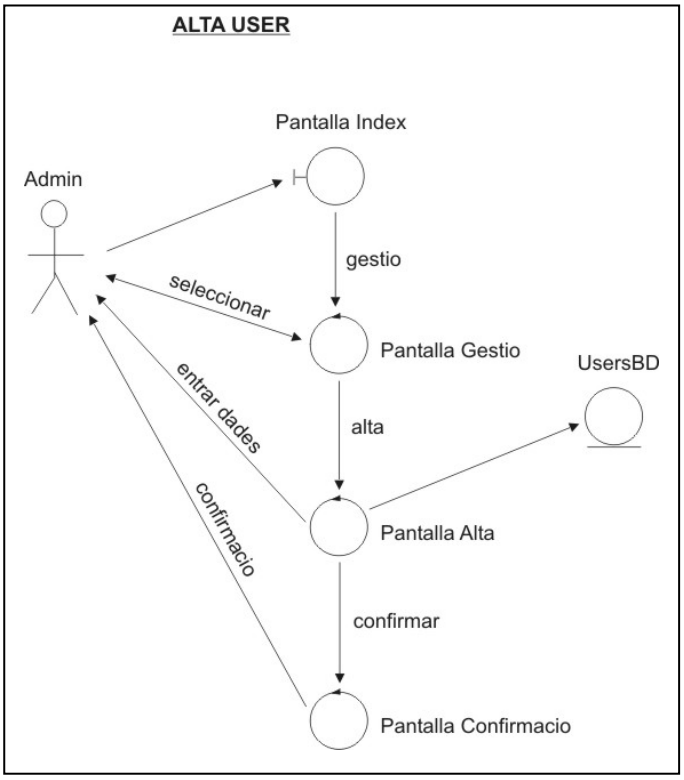

ESBORRAR USER
Admin
Index.jsp
ControladorServlet
MovilImpl
UsersBD
esborrar_user()
esborrar_user()
cerca_user()
cerca_user()
resultat_cerca()
resultat_cerca()
esborrar_user()
esborrar_user()
resultat_esborrar()
resultat_esborrar()

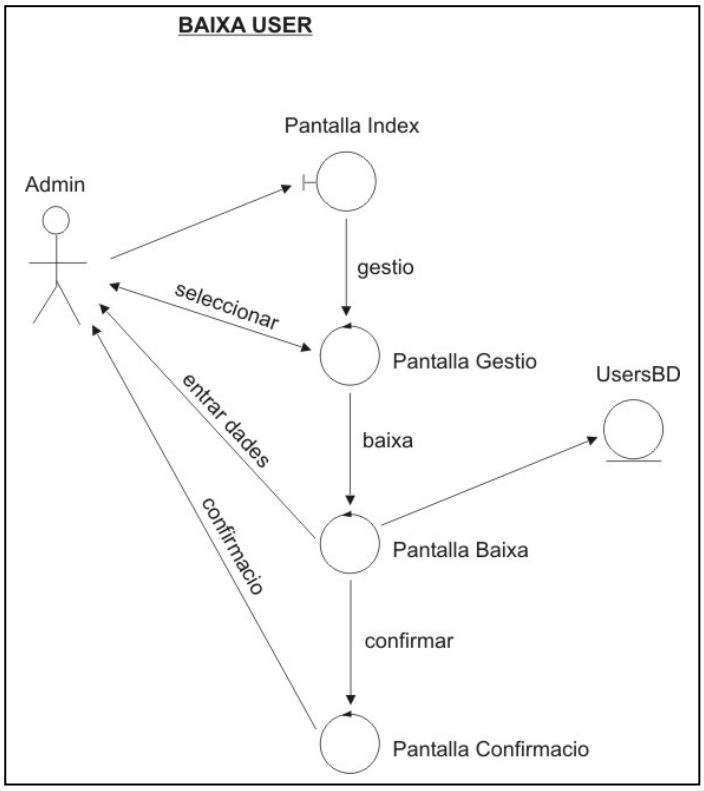
- **Diagrames de col.laboració**



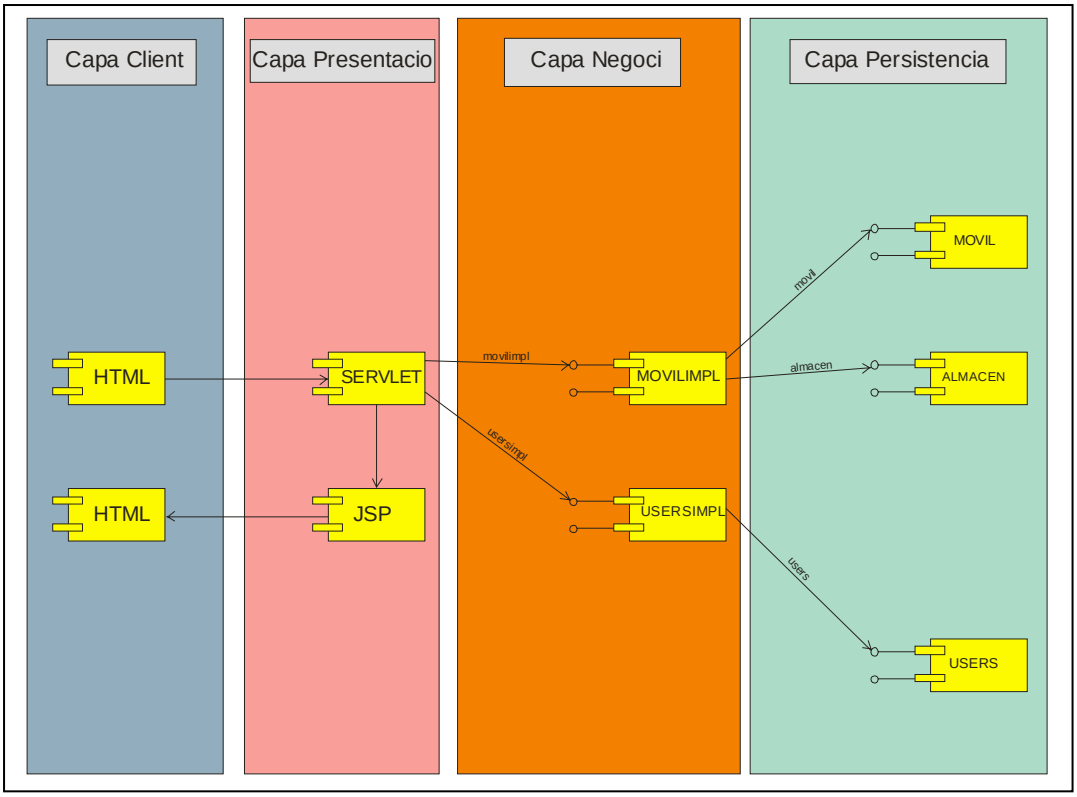




BAIXA MOBIL
Pantalla Index
Admin
gestio
seleccionar
Pantalla Gestio
MobilsBD
entrar dades
baixa
confirmacio
Pantalla Baixa
confirmar
Pantalla Confirmacio


ALTA USER
Pantalla Index
Admin
gestio
seleccionar
Pantalla Gestio
UsersBD
entrar dades
alta
confirmacio
Pantalla Alta
confirmar
Pantalla Confirmacio


BAIXA USER
Pantalla Index
Admin
gestio
seleccionar
Pantalla Gestio
UsersBD
entrar dades
baixa
confirmacio
Pantalla Baixa
confirmar
Pantalla Confirmacio

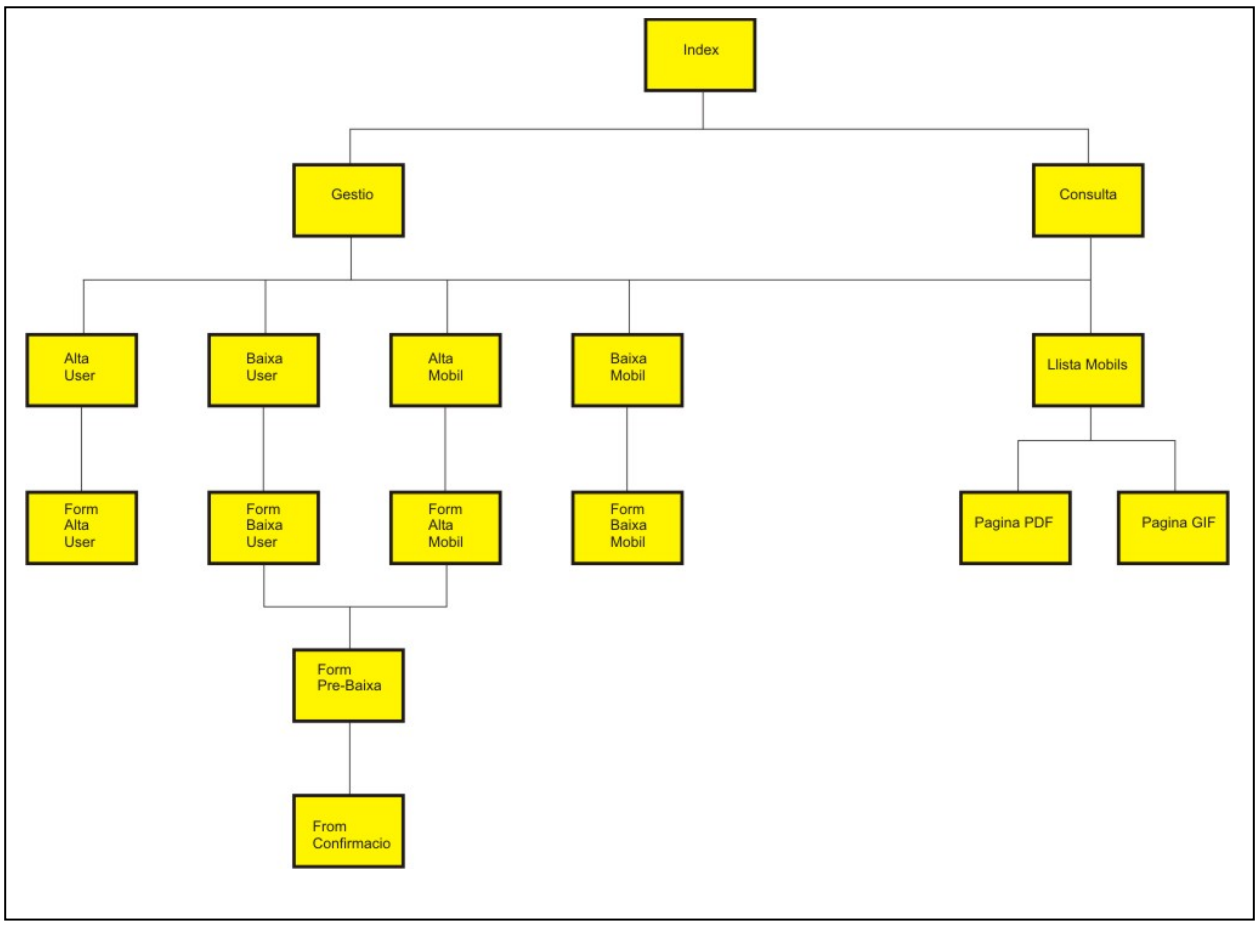
- **Diagrama de components**




Com es pot veure hi ha una diferència entre aquest diagrama i el presentat a l'entrega de la PAC2 (fase disseny). S'han suprimit les interfícies home i remota dels EJB a la part de persistpencia i de negoci degut a que finalment s'ha fet servir hibernate, que no precisa d'aquestes interfícies. També s'ha suprimit la taula URL, ja que no tenia sentit degut a que un mòbil només hauria de tenir una única URL, que es pot representar com un camp més (en el cas de la taula 'movil') o com un atribut més (en el cas de l'objecte 'movil').

## 5.5 Jerarquía de finestres

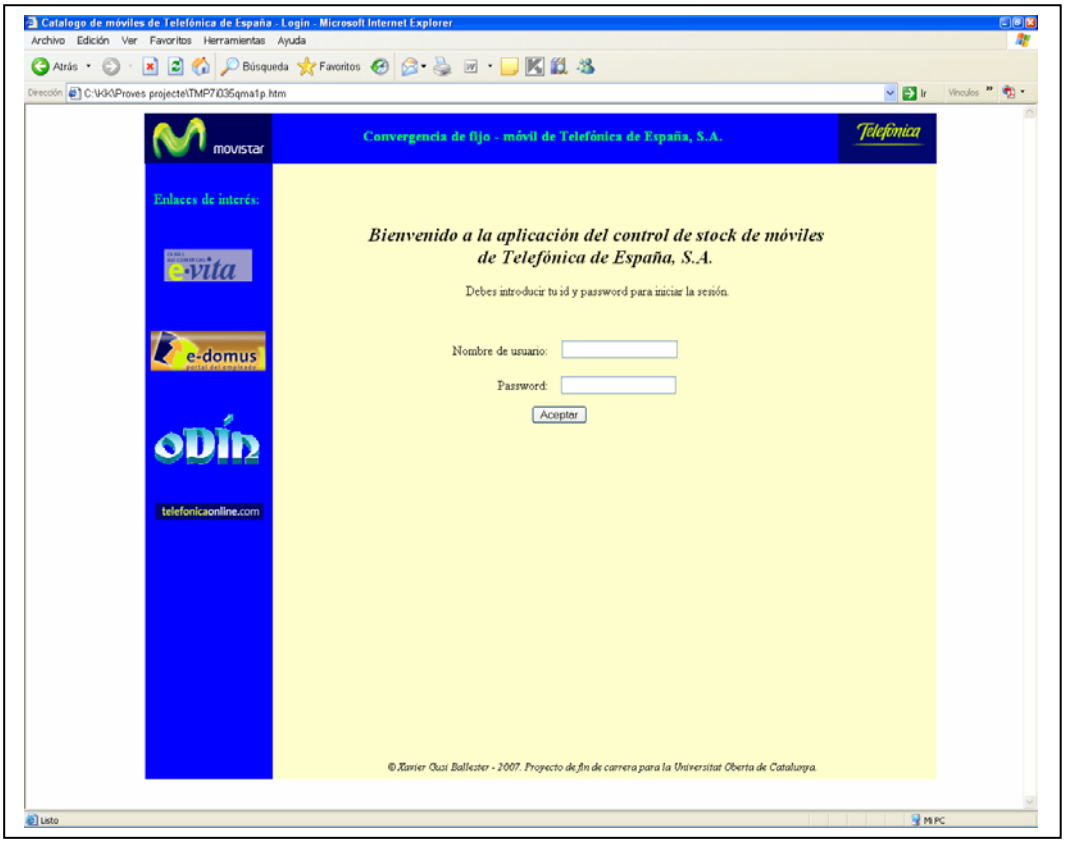
Tota la aplicació presenta una interfície per l'usuari basada en finestres web, amb les que podrà interactuar per a obtenir les diferents peticions y/o sortides. Normalment seràn finestres en html generades per el servlet a partir de jsp's, que cumpliràn una jerarquía d'aquesta manera:



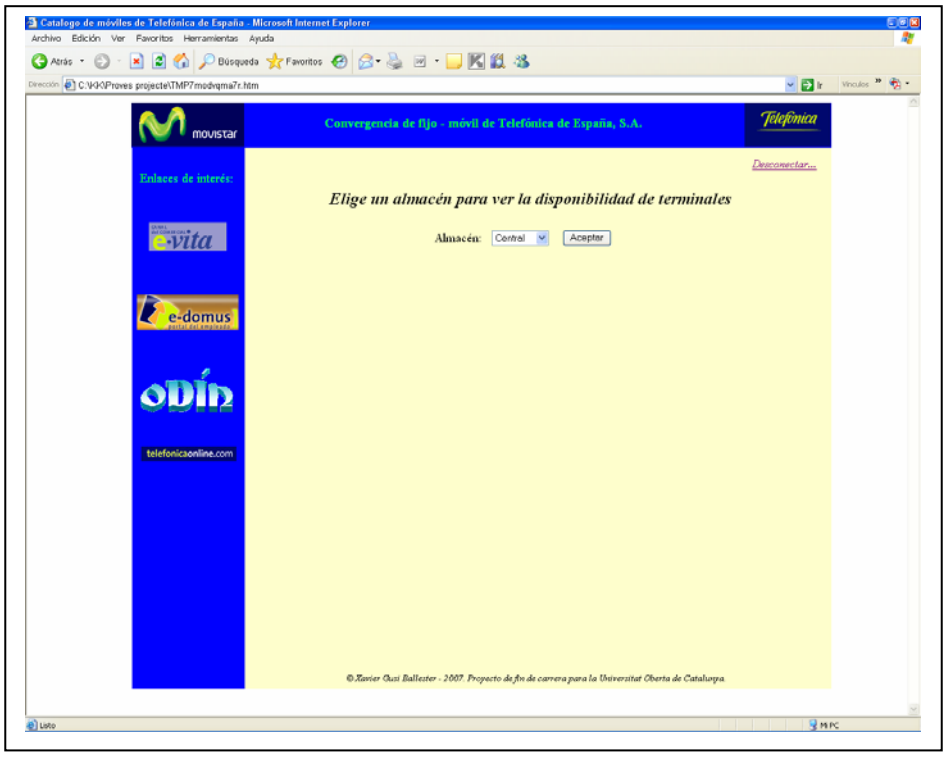

La primera finestra amb la què es trobarà l'usuari serà la del index, que l'obligarà a identificar-se amb l'aplicació, de manera què, depenent del tipùs d'usuari podrà tenir privilegis per fer determinades tasques, això sí, l'usuari administrador podrà fer totes les tasques ofertes per l'aplicació.

## 5.6 Finestres de client

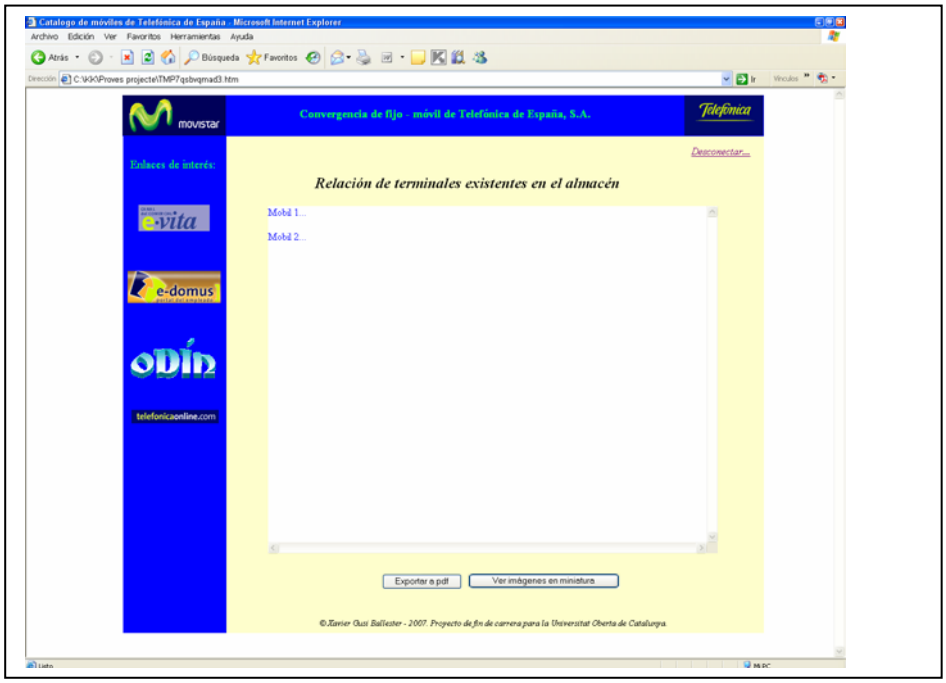
En aquest apartat faré una exposició de les diferents finestres que es trobarà l'usuari a mida que vagi seleccionant les diferents opcions que se li donaràn i tenint en compte el seu perfil.



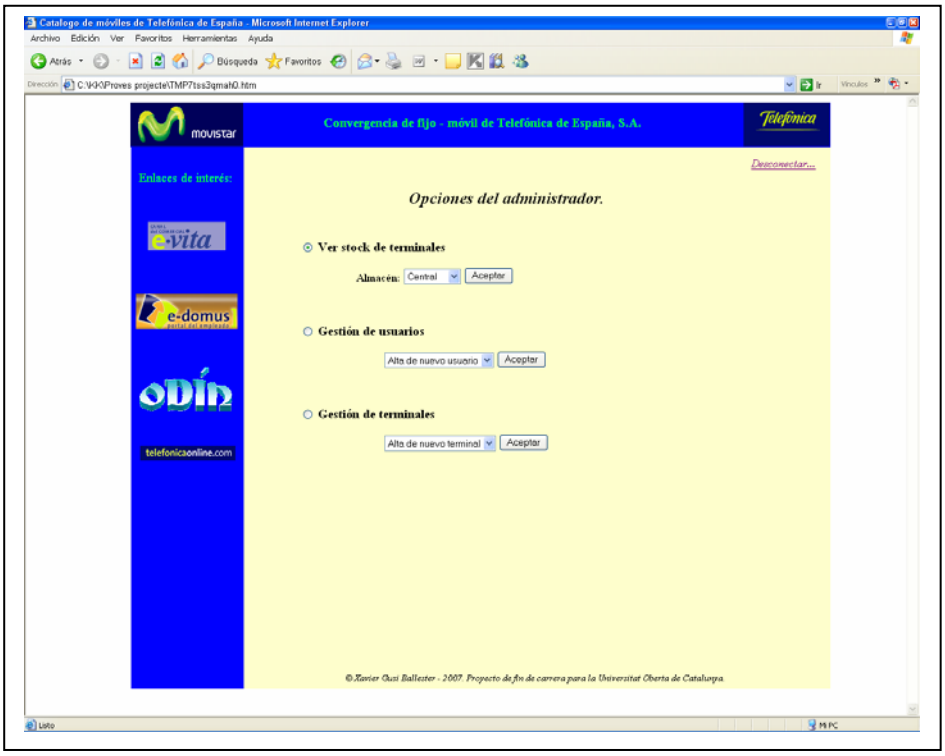

*Finestra d'entrada a l'aplicació que permet identificar-se.*



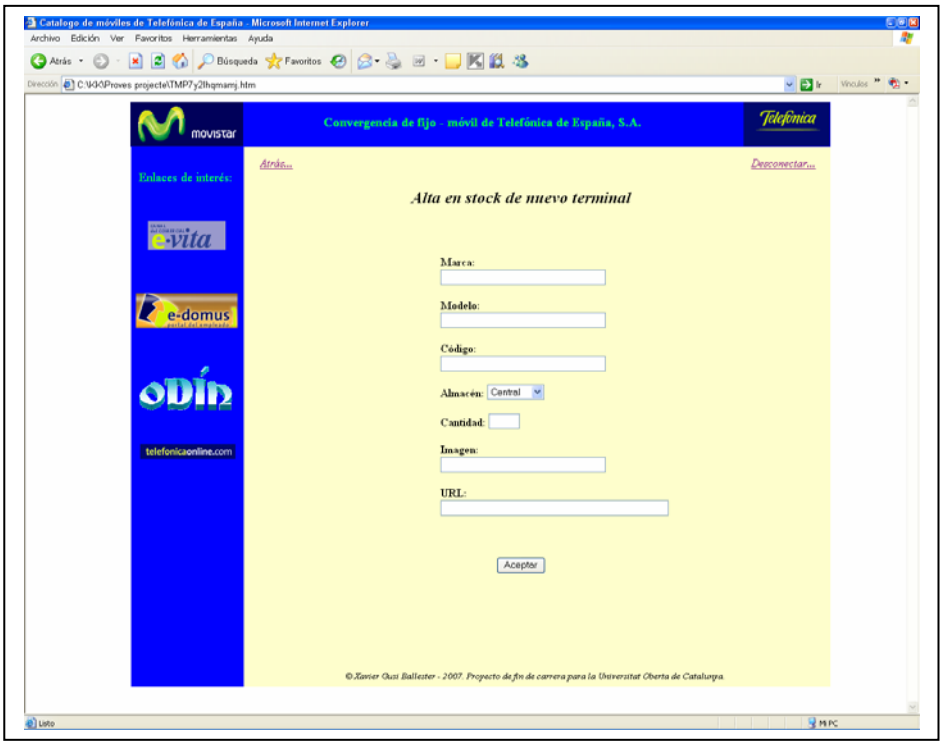

*Finestra per escollir el llistat general dels mòbils disponibles al magatzem escollit.*



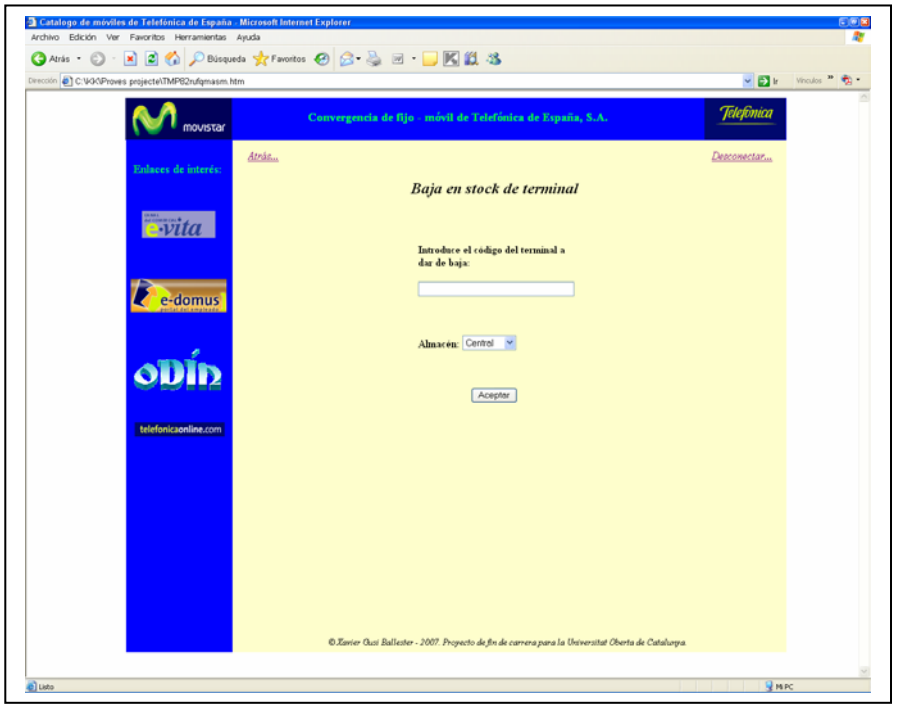

*Finestra que retorna la relació de mòbils disponibles.*




*Finestra que es trobarà l'usuari administrador per a poder fer totes les opcions.*




*Finestra per a donar d'alta un mòbil nou.*




*Finestra per a donar de baixa un mòbil.*

# 6. IMPLEMENTACIÓ

## 6.1 SGBD utilitzat

Per tal d'implementar la persistència de les dades, he escollit el SGBD MySQL, en concret, en la seva versió Server 5.0. Es tracta de un SGBD de codi lliure de llicències i que reuneix totes les necessitats per la implementació del projecte. EN concret algunes de les característiques que es requereixen son:

6 Integritat referencial.
7 Control de les transaccions (BEGIN, COMMIT y ROLLBACK).
8 Tot allò suportat per SQL estandar.

**veure annex script.sql*

## 6.2 La capa de persistència

Els mapejos, amb les corresponents classes, de Objectes – Taules de la BD s'han fet seguint el model relacional de manera que hi ha una classe per cada taula, les classes representen els objectes que son creats amb les característiques de la taula, i això es defineix als fitxers .xml, que son els que fa servir hibernate per a trobar la correspondència objecte – taula i poder gestionar, d'aquesta manera, la persistència.

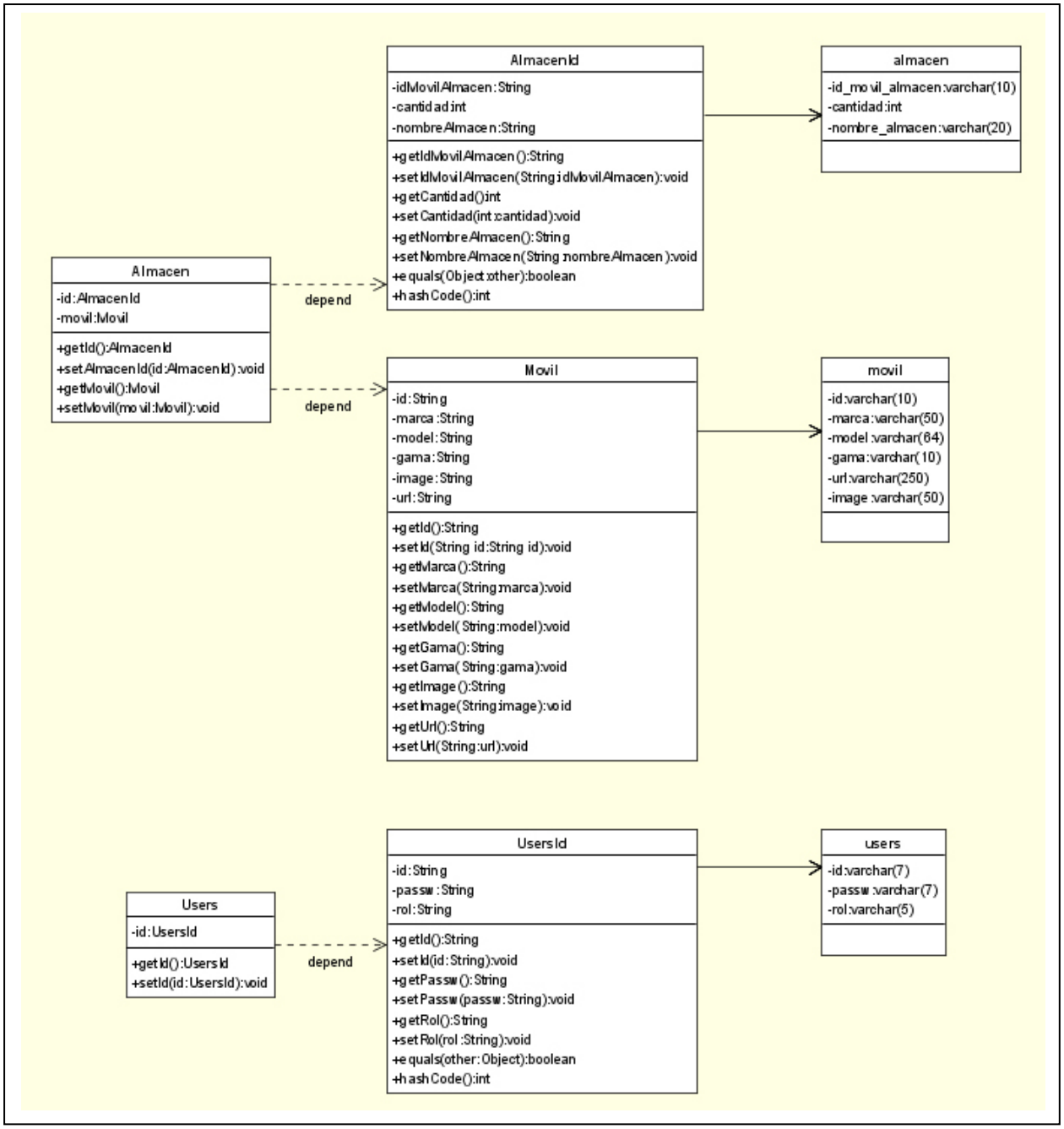
En aquests fitxers, hibernate fa el mapeig de les taules de la BD de manera que associa cada taula amb la seva classe corresponent i cada camp de la taula amb l'atribut de la classe corresponent.

En el moment d'arrencar l'aplicació, hibernate crea les correspondències descrites de manera que l'usuari treballa amb objectes, els quals dona valors o en recupera les seves dades, les quals més tard son guardades a la BD, per tal de donarl-s'hi la persistència en cas de desconnexió de l'aplicació.

D'aquesta manera, tenim una classe i un fitxer .hbm.xml per cada taula de la BD, cosa que simplifica molt la feina rutinària del programador, que no s'ha de preocupar per definir les transaccions típiques com puguin ser:

- Establir la connexió amb la BD.
- Establir un mètode per a afegir nous registres.
- Establir un mètode per a suprimir registres.

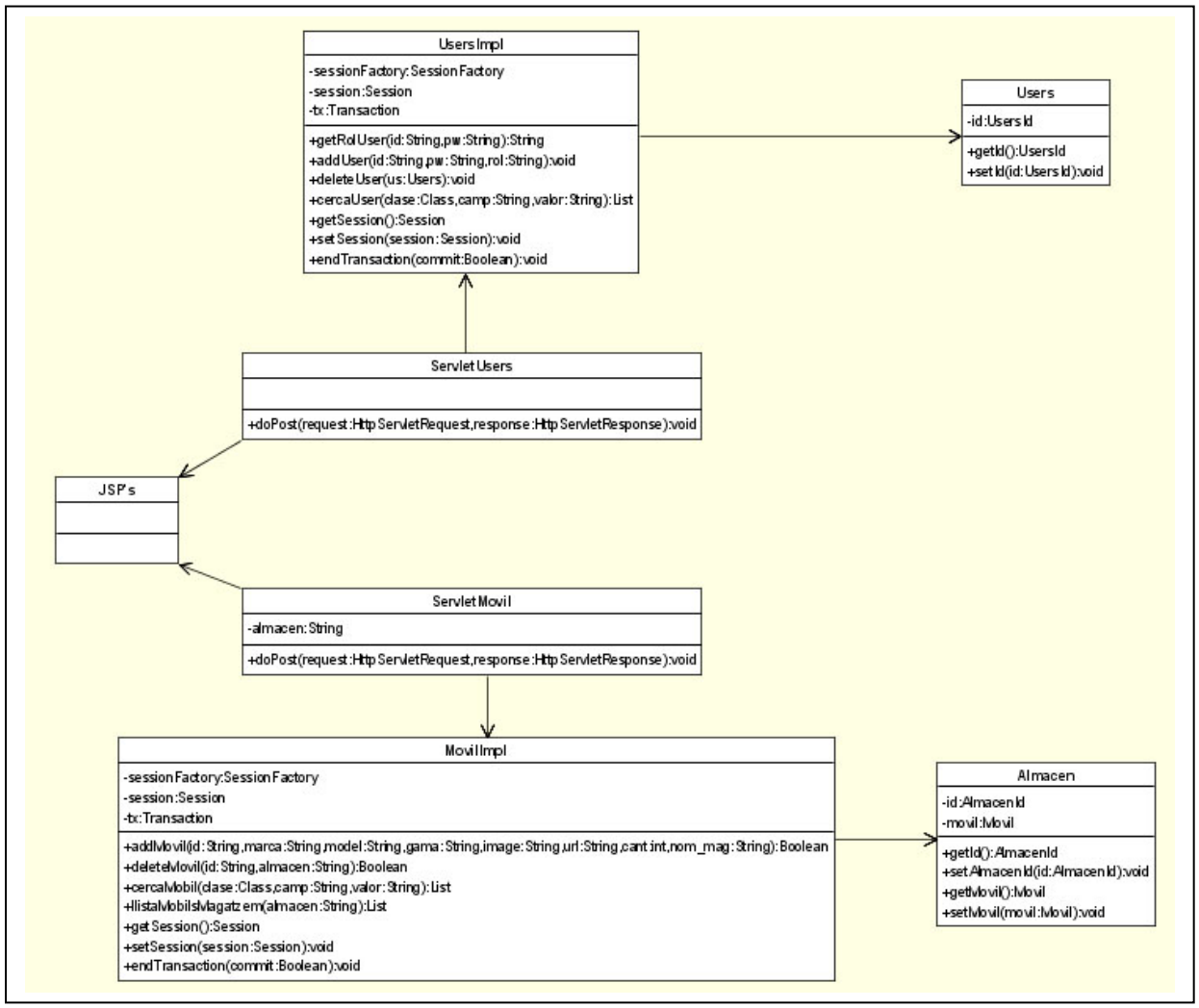
El diagrama de les classes que gestionen la persistència queda de la següent manera:




Tal com es pot veure, la classe AlmacenId correspòn a la taula almacen, la classe Movil correspòn a la taula movil i la classe UsersId correspòn a la taula users. Cal remarcar que hi han, a més, dues classes més, Almacen i Users. Les dues son creades per hibernate, la primera es crea per a poder representar la relació que hi ha entre AlmacenId i Movil, ja que tenim una clau primària (id de Movil) que, al mateix temps, és clau forània de AlmacenId, doncs bé, aquesta relació es representa mitjançant la classe Almacen. La segona, la classe Users, es crea per a poder representar els objectes UsersId, que també té una clau primària, i les relacions que pugui tenir, encara que ara mateix no en tingui.

## 6.3 La capa de negoci

El diagrama de les classes que gestionen la capa de negoci queda de la següent manera:




Realment les pàgines JSP corresponen a la capa client i les pàgines Users i Almacen corresponen, com hem vist abans, a la capa de persistència, però les poso en aquest esquema perquè es vegi millor la relació entre les diferents capes.

Les classes ServletMovil i ServletUsers son les encarregades de generar les sortides/entrades desde la capa de client, generen el codi html que veu el client i recuperen les dades que desde aquestes pàgines entra el client.

Per la seva banda, les classes MovilImpl i UsersImpl, s'encarreguen de generar les crides cap a la capa de persistència per tal de recuperar, crear o esborrar objectes que després son guardats a la BD.

Amb aquesta implementació aconseguim una bona separació entre les diferents capes, de manera que les tasques de manteniment de l'aplicació o implementació de noves funcionalitats resulti realment sencill. Així, si per exemple volem modificar la presentació que veu el client i afegir un altre botó, n'hi ha prou amb modificar només la pàgina jsp que li dona aquesta funcionalitat. Si per exemple, volem afegir un nou criteri de cerca, haurem de incloure aquest a la pàgina jsp i incloure un nou mètode a la classe que fa la implementació (UsersImpl o MovilImpl).

## 6.4 Classes auxiliars

En el projecte existeix un paquet que l'hi he posat de nom 'constants' que bàsicament té dues classes que podriem denominar auxiliars.

La primera classe es diu igual, *constants* en la què es defieneixen uns valors de uns atributs publics constants per a l'aplicació, com puguin ser el nom de usuari de la BD, els noms dels fitxers de les pàgines jsp, el driver de connexió amb la BD, etc. Això és especialment útil si tenim en compte que l'aplicació es podria fer servir amb d'altres jsp, o fins i tot amb una BD diferent (sempre i quan també cambiéssim els mapejos del fitxer que fa servir hibernate).

L'altre classe és la *HibernateSessionFactory* i, com el seu nom indica, s'en encarrega de gestionar les *Sessions*. Té molta utilitat en la reutilització, ja que sabem que és una classe que funciona perfectament i, en cas de voler fer una altre aplicació només caldrà importar-la en el nou projecte.

# 7. CONCLUSIONS

Una vegada finalitzat aquest projecte, la conclusió ha estat un aprenentatge de la programació distribuïda i arquitectura J2EE, conceptes com separació entre capes i persistència (desconeguts per a mí fins a aquest moment) a partir dels coneixements adquirits a la carrera, com programació OO, enginyeria del programari o tècniques de desenvolupament del programari. Realment el model MVC té molta utilitat en el món empresarial per la seva reutilització, economia i manteniment. Fruit de tot aquest treball he tret les següents conclusions:

- Patrons de disseny com hibernate m'han facilitat molt el treball.
- L'arquitectura J2EE és idònia per el desenvolupament d'aplicacions distribuïdes a nivell empresarial que requereixin escalabilitat, integració, consistència, robustesa i seguretat.
- Paral·lelament he fet servir eines que no coneixia gaire, com UltraDev4, Poseidon o Eclipse.
- Les aplicacions creades amb aquest model tenen un gran avantatge addicional, que son idònies per a treball distribuït, es a dir, del tipus client-servidor. El client no necessita instal·lar cap tipus de software en local, només requereix saber la URL on es troba l'aplicació i connectar-s'hi a través del seu navegador, cosa que permet el treball des de qualsevol punt en el cas que l'aplicació sigui visible des de internet.
- Conseqüentment, les actualitzacions de l'aplicació no requereixen la intervenció en cada una de les màquines dels clients, sinó que només es duu a terme en el propi servidor, de manera que, en actualitzar-la, el client ja veu la nova versió de manera automàtica.
- La seguretat és quelcom que queda implícit, sense que el programador se'n hagi de preocupar.

## Notes

El ventall de possibilitats i funcionalitats que es pot fer és immens i, en el meu cas, m'ha costat molt poder arribar a implementar l'aplicació. Una aplicació d'aquestes característiques amb més funcionalitats requereix un cert bagatge i una certa experiència que només s'adquireix a mida que es va treballant amb aquest model. Vull deixar constància que l'aplicació que aquí es representa es fruit únicament del meu treball sense cap ajuda externa addicional a part de la que he anat trobant principalment per internet i, a més, des de la més absoluta inexperiència en la implementació de la arquitectura J2EE. Sens dubte, amb més temps i més pràctica arribaré a fer aplicacions molt millors i amb molta més funcionalitat que la que he pogut fer en un sol semestre acadèmic.

## 8. GLOSSARI

- **Administrador**. Persona que té permisos per a modificar les dades de l'aplicació, com creació/baixa de usuaris, alta/baixa de telèfons.
- **User**. Usuari que té només permís per a accedir a veure l'estock de terminals.
- **Magatzem**. Lloc on resideix l'estock de telèfons.
- **Terminal**. Símil de telèfon mòbil.
- **Id**. Codi identificatiu de un objecte.
- **Password (pw)**. Contrasenya personalitzada de un usuari que només ha de saber ell mateix i que tampoc ningú ha de poder modificar.
- **Framework**. Bàsicament és codi ja implementat i provat a l'abast del programador per a facilitar-l'hi la seva tasca. També s'en poden dir 'patrons de treball'.
- **Capes de negoci**. Son les diferents capes que formen una aplicació, amb una tasca molt concreta per a cada una d'elles i que normalment han de quedar ben separades.
- **Persistència**. El fet que les dades quedin emmagatzemades i no es perdin una vegada s'ha sortit de l'aplicació o s'ha acabat una transacció.
- **Login**. Procés per el qual un usuari s'ha de validar per a poder entrar a l'aplicació.

## 9. BIBLIOGRAFIA


- http://java.sun.com/javaee/overview/compatibility.jsp
- http://es.wikipedia.org/wiki/Java_EE
- http://ca.wikipedia.org/wiki/Hibernate
- http://ca.wikipedia.org/wiki/Enterprise_JavaBean
- *JAVA & J2EE. Curso básico de aplicaciones web.* Manuel Salvadores Olaizola.
- *Aplicaciones web con Java.* Raul Caballero Ortega / Álvaro Sánchez Mariscal.
- *J2EE. Una plataforma de componentes distribuida.* Editorial UOC.
- *Servlets y JSP.* Juan Antonio Palos (Ozito).