Estudis d'Informàtica i Multimèdia

Semestre: Primavera de 2004

Enginyeria Tècnica en Informàtica de Sistemes

Treball de Fi de Carrera

Desenvolupament en java per a Internet.

Memòria del Projecte

Jordi Guillaumes i Pons
jguillaumes@uoc.edu
02/03/2004

RESUM

Aquest treball de fi de carrera consisteix en el disseny i implementació d'una "botiga web" bàsica. Aquesta botiga web ha de permetre el manteniment d'un inventari d'articles a la venda, agrupats en categories, i la seva posada a disposició dels clients que hi accedeixin mitjançant un navegador estàndard.

Tècnicament, la implementació ha de basar-se en l'especificació J2EE de Sun Microsystems. En el moment de fer aquest desenvolupament la darrera versió de l'especificació és la 1.4. Tanmateix, la versió que podríem considerar com a consolidada és la 1.3. La majoria de les eines i servidors J2EE disponibles implementen aquesta versió, que és la que s'ha escollit per fer el desenvolupament.

Cal tenir present en tot moment que aquest desenvolupament és un **treball acadèmic**. L'objectiu al desenvolupar-lo no ha estat tant construir un producte explotable comercialment, sinó arribar a aprendre i entendre el funcionament de la tecnologia J2EE (i altres infraestructures utilitzades, com Struts). Per aquest motiu, s'han pres algunes decisions de disseny que serien discutibles des del punt de vista del rendiment de l'aplicació o del seguiment estricte de les *best practices*. Això s'aplica sobre tot a l'ús de EJB d'entitat per implementar la persistència de les Entitats del sistema: probablement aquesta aplicació no justifica el cost d'usar aquest recurs si no és per assolir coneixements al respecte.

Finalment, cal tenir present que aquesta implementació no és, ni pretén ser, una **implementació completa** de les funcions que ha de reunir un lloc de comerç electrònic. La complexitat i el nombre de característiques que ha de tenir una aplicació d'aquest tipus fa inviable el seu desenvolupament per part d'una única persona, treballant-hi a **temps parcial** i durant quatre mesos. En la presentació dels objectius del TFC se'ns va instruir específicament en delimitar l'abast del projecte per tal de no assolir compromisos inabastables. En particular, aquest desenvolupament deixa explícitament fora del seu abast:

- Qüestions de seguretat, llevat del pur registre dels usuaris-clients i dels administradors. Específicament, no s'usa cap mena de criptografia ni en les comunicacions (SSL) ni en l'emmagatzemament de la informació (les paraules-clau es desen en clar, sense xifrar.
- Cobrament de les comandes. No s'implementa ni es simula cap gestió de cobrament a través de tarja de crèdit o similar.
- Gestió de les comandes un cop realitzada la compra. No s'ha implementat cap seguiment de comandes. De fet, fora d'una llista bàsica de comandes realitzades no hi ha cap funció relativa a facturació, tramesa o control de les comandes dels clients.

Taula de continguts

1. INTRODUCCIÓ 1

1.1. JUSTIFICACIÓ DEL TFC I CONTEXT EN EL QUE ES DESENVOLUPA. 1
1.2. OBJECTIUS DEL TFC 1
1.3. ENFOCAMENT I MÈTODE SEGUIT. 2
1.4. PLANIFICACIÓ DEL PROJECTE. 2
1.5. PRODUCTES OBTINGUTS. 3
1.6. ESTRUCTURA DE LA RESTA D'AQUESTA MEMÒRIA 3

2. ANÀLISI. 4

2.1. PRESENTACIÓ 4
2.2. DOMINI DEL PROJECTE 4
2.3. CASOS D'ÚS. 5
2.3.1. Relació de casos d'ús. 5
2.4. DETALL DELS CASOS D'ÚS SELECCIONATS. 6
2.4.1. UCF030 – Navegació per la botiga. 6
2.4.2. UCF040 – Gestió del carret de compra 7
2.4.3. Diagrama de casos d'ús. 8
2.5. CLASSES DEL DOMINI. 9
2.5.1. Diagrama de classes del domini. 9
2.5.2. Descripció breu de les classes. 9
2.6. ANÀLISI DINÀMICA 10

3. DISSENY. 11

3.1. PRESENTACIÓ 11
3.2. CASSOS D'US DETALLATS 11
3.2.1. UCF010 – Connexió al sistema. 11
3.2.2. UCF020 – Registre de nou usuari. 11
3.2.3. UCB010 – Gestió d'usuaris. 12
3.2.4. UCB020 – Gestió de categories. 13
3.2.5. UCB030 – Gestió de Productes. 14
3.3. DISSENY. 15
3.3.1. Patrons de disseny a aplicar. 15
3.3.2. Diagrama de classes de disseny.. 16
3.3.3. Model relacional i model de persistència. 17
3.3.4. Diagrames de col·laboració 18
3.4. EXEMPLES DE PANTALLES 20

4. IMPLEMENTACIÓ. 24

4.1. ESTRUCTURA GENERAL. 24
4.1.1. edu.uoc.jguillaumes.tfc.beans 24
4.1.2. edu.uoc.jguillaumes.tfc.dao 24
4.1.3. edu.uoc.jguillaumes.tfc.ejb 24
4.1.4. edu.uoc.jguillaumes.tfc.excepcions 25
4.1.5. edu.uoc.jguillaumes.tfc.forms 26
4.1.6. edu.uoc.jguillaumes.tfc.interfaces. 26
4.1.7. edu.uoc.jguillaumes.tfc.resources 27
4.1.8. edu.uoc.jguillaumes.tfc.servlet 27
4.1.9. edu.uoc.jguillaumes.tfc.vobj 27
4.2. ACTION FORMS I ACTION BEANS 28
4.3. EJBS 28
4.3.1. Entity EJB. 28
4.3.2. Stateless Session EJB. 29
4.3.3. Stateful Session EJB: carret de la compra. 29
4.4. OBJECTES-VALOR 30
4.5. ALTRES CLASSES DE SUPORT 32

4.5.1. ControladorLlista i ControladorLlistaList. 32
4.5.2. MailBean. 33
4.6. ALTRES DETALLS D'IMPLEMENTACIÓ....33
4.6.1. Logging. 33
4.6.2. Internacionalització. 33

5. DESPLEGAMENT SOTA JBOSS I POSTGRESQL. 34

5.1. REQUISITS PER A LA INSTAL·LACIÓ. 34
5.2. TASQUES PRÈVIES AL DESPLEGAMENT. 34
5.3. DESPLEGAMENT DE L'APLICACIÓ. 34
5.4. INICIAR L'APLICACIÓ. 35
5.5. CONFIGURACIÓ DE LA TRAMESA DE CORREU ELECTRÒNIC....35

6. CONCLUSIONS. 37

6.1. OBJECTIUS ASSOLITS. 37
6.2. OBJECTIUS NO ASSOLITS. 37
6.3. ERRORS I INCIDÈNCIES AL PROGRAMARI. 38
6.4. HIPOTÈTICA EVOLUCIÓ FUTURA DE L'APLICACIÓ. 38
6.4.1. Implementació de funcionalitats que falten. 38
6.4.2. Correcció de problemes i millores generals. 38

Figures

FIGURA 1 - DIAGRAMA DE CASOS D'US 8
FIGURA 2 - DIAGRAMA DE CLASSES DEL DOMINI 9
FIGURA 3 - DSS PER AL CAS D'ÚS UCF030 10
FIGURA 4 - DSS PER AL CAS D'ÚS UCF040 10
FIGURA 5 - CASOS D'ÚS RELATIUS A GESTIÓ D'USUARIS I SESSIONS 13
FIGURA 6 - DIAGRAMA DE CLASSES DE DISSENY 17
FIGURA 7 - DIAGRAMA E/R 18
FIGURA 8 - DIAGRAMA DE COL·LABORACIÓ: REFRESC DE LLISTA DE CATEGORIES 19
FIGURA 9 - DIAGRAMA DE COL·LABORACIÓ: CREACIÓ D'UNA NOVA CATEGORIA. 19
FIGURA 10- DIAGRAMA DE COL·LABORACIÓ: MODIFICACIÓ D'UNA CATEGORIA 19
FIGURA 11 - PANTALLA DE GESTIÓ DE CATEGORIES 20
FIGURA 12 - PANTALLA DE MODIFICACIÓ DE CATEGORIES 21
FIGURA 13 - PANTALLA DE GESTIÓ D'ARTICLES 22
FIGURA 14 - PANTALLA DE MODIFICACIÓ D'ARTICLES. 23
FIGURA 15 - CARRET DE COMPRA I CLASSES RELACIONADES 30
FIGURA 16 - RELACIONS ENTRE ELS OBJECTES-VALOR 31
FIGURA 17 - CONTROLADORLLISTA I CONTROLADORLLISTALIST 32
FIGURA 18 - PANTALLA INICIAL DE L'APLICACIÓ 35

1. Introducció

1.1. Justificació del TFC i context en el que es desenvolupa.

No hi ha cap mena de dubte que els desenvolupaments relacionats amb la xarxa Internet són actualment, i seguiran sent-ho els propers anys, un dels entorns de major activitat en la indústria del programari. El fet de que aquest entorn és relativament nou, i el fet de que les tecnologies que podríem anomenar *tradicionals* no s'ajustin bé del tot als reptes que presenta aquest entorn, han fet que els darrers anys hagin sorgit tota una sèrie de tecnologies que competeixen a l'hora de captar l'atenció dels desenvolupadors que treballen en aquest entorn. Entre aquestes tecnologies podem situar des de desenvolupaments de l'àmbit del programari lliure, com ara PHP, fins a infraestructures purament propietàries, com a .NET, i finalment, l'entorn J2EE, proposat per Sun Microsystems i recolzat pels pesos pesants de la indústria, com són Oracle, BEA Systems i IBM.

Centrant-nos en la tecnologia que serà central en el nostre desenvolupament, J2EE, cal dir també que no es tracta d'una especificació congelada; ben al contrari, està en continu canvi i desenvolupament. En el moment en que s'escriu aquesta memòria, la darrera versió és la 1.4. Tanmateix, el projecte es basarà en la versió anterior, 1.3, que es pot considerar com a consolidada i d'ampli ús a la indústria.

Les diferències entre les diferents versions de l'especificació (i les seves sub-especificacions) té una gran importància tant des del punt de vista d'un desenvolupador d'aplicacions, com des del d'un tècnic o analista de sistemes. La raó és que apareixen avenços que tenen un fort impacte tant en el plantejament del disseny a nivell purament aplicatiu (quines parts de l'especificació farem servir? Davant d'alternatives, quina és la més convenient?) com a nivell d'establiment de normes i *policies* per part dels administradors d'un sistema (com dividirem els mòduls d'una aplicació? Farem una instal·lació distribuïda? Separarem el contenidor web del contenidor d'EJBs?). Són aquestes qüestions les que fan interessant adquirir coneixements de primera ma sobre el funcionament d'aquesta tecnologia tant al desenvolupador d'aplicacions (per raons òbvies) com al tècnic o enginyer de sistemes, que si bé no haurà de desenvolupar aplicacions de gestió, sí que haurà de poder valorar l'impacte que representa una aplicació desenvolupada sota J2EE per al rendiment i funcionament d'un sistema informàtic, així com les diferents opcions relatives a la configuració i desplegament de les aplicacions en entorns de producció reals.

Així, tot i que en un principi el plantejament d'aquest TFC pot resultar més atractiu per a un futur estudiant de la branca de Gestió, s'ha d'entendre que l'enginyer de sistemes es trobarà probablement en un entorn en el que haurà de fer recomanacions i/o prendre decisions en les que un coneixement pràctic i de primera ma de la tecnologia J2EE pot resultar inavaluable. Això justifica plenament l'interès d'aquest projecte per als professionals de les ambdues branques i, de fet i en el cas concret d'aquest projecte, suposarà un dels aspectes que més es tindrà en compte a l'hora de fer el disseny i implementació: el contacte amb les diferents parts d'aquesta tecnologia i la comprensió del seu us.

1.2. Objectius del TFC

Hem de separar els objectius a assolir en aquest TFC en dos grups: els objectius *acadèmics* i els objectius que podríem anomenar *de negoci*.

Respecte als primers, ja s'han comentat al punt anterior. L'objectiu és l'aprenentatge i familiarització amb la tecnologia J2EE i els seus entorns de desenvolupament, desplegament i producció. Aquestes no són les qüestions principals que es tocaran en aquesta memòria, tot i que se'n farà algun comentari quan resulti adient.

En el que respecta als *objectius de negoci*, els definirem com les prestacions/funcionalitats operatives que volem implementar en la nostra aplicació de *botiga virtual*. Com que el temps i els recursos són limitats, hem d'acotar l'abast del projecte, deixant de banda força coses que serien necessàries en una aplicació real. Concretant més, els objectius que pretenem assolir són:

- Gestionar un inventari de productes a la venda, agrupats per categories.
- Permetre a un usuari-client registrar-se a la botiga per fer comandes.
- Permetre als usuaris registrats fullejar l'inventari de productes a la venda, i afegir-los a un carret de la compra virtual.
- Permetre a l'administrador de la botiga gestionar la llista d'usuaris registrats, fent altes, baixes, modificacions i canvis de privilegis.

De la mateixa manera, exclourem explícitament de l'abast del projecte les funcionalitats següents:

- Seguretat i criptografia.

- Cobrament de les comandes. No es farà ni es simularà el cobrament electrònic de les comandes dels clients.
- Gestió de comandes realitzades. No es desenvoluparan les funcions relatives a facturació ni tramesa de comandes. La única funció que es desenvoluparà serà un llistat bàsic de comandes realitzades.

Tot i aquestes limitacions, forçades pel límit de temps de treball, aquesta memòria contindrà indicacions relatives a la forma en la que s'hauria d'enfocar el desenvolupament de la part que faltaria per convertir l'aplicació en un paquet implementable i comercialitzable.

1.3. Enfocament i mètode seguit.

La metodologia de desenvolupament aplicada ha estat el Rational Unified Process (RUP), adaptada a les especials circumstàncies del projecte:

- És un projecte d'una sola persona.
- L'analista-dissenyador-desenvolupador ha d'aprendre a manegar la tecnologia utilitzada a l'hora que es du a terme el projecte.
- No hi ha un *client* o usuari final amb el que interactuar.
- El constrenyiment del temps dificulta seguir puntualment totes les etapes del RUP.

Per aquests motius, el seguiment de les tasques i les activitats que estableix el RUP no ha estat completa. Concretament:

- L'anàlisi ve a correspondre a una *iteració inicial* del RUP. Es centra en els casos d'ús més crítics, i només en relaciona la resta.
- El disseny està abreviat. Cal tenir present que el disseny s'ha fet a l'hora que s'adquiria familiaritat amb les eines de desenvolupament, l'especificació J2EE i el servidor JBoss que s'anava a usar en la implementació. Per aquest motiu, no s'ha assolit tot el que hauria de contenir el document de disseny. En particular, no s'ha fet tot el disseny extern (pantalles), sinó que se n'han dissenyat unes de bàsiques que després s'han usat en totes les funcions implementades (amb els canvis corresponents).

És interessant també comentar una mica l'entorn (o entorns) de desenvolupament que s'han usat. Durant el treball s'ha pogut comprovar de primera ma la portabilitat d'una aplicació java/J2EE. S'han usat tres màquines durant el treball:

- Un ordinador Powerbook G4 amb sistema operatiu MacOS X 10.3, que ha estat la màquina principal de desenvolupament. Aquest ordinador compta amb una JVM 1.4.2 d'Apple, i un servidor JBoss 3.2 empaquetat també per Apple.
- Un portàtil Compaq Presario, amb processador Pentium 3 i sistema operatiu Windows XP Home. Aquest ordinador compta amb la JVM 1.4.2 de Sun Microsystems i el servidor JBoss 3.2 obtingut del lloc de descàrrega oficial a Internet.
- Un ordinador d'escriptori amb processador Intel Celeron i sistema operatiu Linux (versió de nucli 2.4.X), amb la màquina virtual 1.4.2 de Sun Microsystems i servidor JBoss 3.2 obtingut d'Internet.

Les dues primeres màquines s'han usat per al desenvolupament, que s'ha basat en l'entorn Eclipse (versions 2.2 i 3.0). La darrera màquina s'ha usat com a "màquina de producció", a la que s'ha desplegat l'aplicació quedant aquesta accessible des de la xarxa Internet real (http://tfc.jordi.guillaumes.name).

En tots els casos, el SGBD usat ha estat PostgreSql, versions 7.3 i 7.4[1]. S'ha usat aquest SGBD i no el suggerit (MySQL) ja que ja tenia una certa familiaritat amb aquest entorn, cosa que ha estalviar l'haver d'aprendre una cosa més.

1.4. Planificació del Projecte.

La planificació inicial del Projecte està inclosa a l'anàlisi, que és la base del capítol 2 d'aquesta memòria.

L'avaluació posterior del progrés del Projecte indica que s'han assolit pràcticament tots els objectius i dates de control de la planificació inicial, amb la salvedat de que la finalització de la fase de disseny ha estat tardana, cosa que no ha influenciat massa la data final ja que s'ha estat codificant pràcticament des del primer dia.

[1] Cal prestar atenció a la versió utilitzada, ja que els drivers JDBC són diferents i poden donar (de fet donen) problemes si s'usen amb una versió incorrecta.

1.5. Productes obtinguts.

Els productes (*deliverables*) obtinguts s'empaquetaran en un arxiu ZIP, que tindrà el següent contingut:

Arxiu	***Contingut***
jguillaumes-tfc-Memoria.doc	Aquest document
jguillaumes-tfc-javadoc.jar	Estructura de directoris dels javadocs del projecte. Es poden expandir o bé usar algun entorn que pugui llegir-los directament dins del jar (com ara NetBeans).
jguillaumes-tfc-Presentacio.ppt	Presentació powerpoint.
jguillaumes-tfc.instal.lacio.doc	Guia d'instal·lació i desplegament de l'aplicació en un servidor JBoss.
jguillaumes-tfc.ear	L'aplicació empaquetada per a ser desplegada a un servidor JBoss 3.2.X. Aquest arxiu contindrà només codi compilat (binaris).
jguillaumes-tfc-fonts.jar	Projecte Eclipse complet. Aquest paquet contindrà el codi font i els arxius auxiliars usats per al desenvolupament. Es pot importar directament a una àrea de treball d'Eclipse, o descomprimir-lo directament per examinar-lo com es vulgui.
jguillaumes-tfc.sql	Script amb comandaments SQL per crear la base de dades que ha de donar suport a l'aplicació. Conté tant el DDL per crear les taules relacionals, com les sentències per inserir les files necessàries per inicialitzar el sistema (bàsicament, un usuari administrador).
jguillaumes-tfc-ds.xml	Definició del *datasource* usat per l'aplicació. Es pot desplegar directament al JBoss.
jguillaumes-tfc-login.xml	Fragment XML que defineix el domini de seguretat sota el que ha de córrer l'aplicació. Cal inserir-lo a l'arxiu de configuració login-config.xml del JBoss, tal com descriu el document d'instal·lació.

Val a dir que durant aquest desenvolupament s'ha usat CVS per administrar els canvis. Si fos necessari es podria facilitar el buidat complet del repositori en un arxiu addicional.

1.6. Estructura de la resta d'aquesta memòria.

La resta de capítols de la memòria té el següent contingut:

- El **capítol 2** conté l'anàlisi. És bàsicament el contingut de la PAC-2, amb alguns canvis menors d'estructura.
- El **capítol 3** conté el disseny. Està basat en la PAC-3, ampliada amb alguns detalls indicats pel consultor, i actualitzada amb alguns dels canvis que s'han fet durant la implementació.
- El **capítol 4** exposa quina ha estat finalment la implementació efectuada. Conté una breu descripció de cada una de les classes implementades, així com comentaris relatius als problemes i incidents trobats (i resolts) durant el desenvolupament.
- El **capítol 5** reprodueix bàsicament la guia d'instal·lació i desplegament.
- Finalment, el **capítol 6** resumeix l'experiència que ha estat fer aquest desenvolupament, i fa indicacions sobre els treballs que caldria fer per completar el producte per tal de fer-lo implementable i comercialitzable en el mon real.

2. Anàlisi.

2.1. Presentació

Aquest document conté l'anàlisi de l'aplicació de "botiga electrònica" que cal desenvolupar per completar el TFC. Tal com es deia al document de planificació, això no s'ha d'entendre com a una anàlisi estricta segons les metodologies en cascada, sinó com el resultat d'una primera iteració en una metodologia tipus UP, en la que el pes del treball es centra en l'esmentada anàlisi

El contingut d'aquesta anàlisi estarà format, doncs, per:

- Domini del Projecte.
- Relació de casos d'ús no detallats.
- Casos d'ús més rellevants en quant a criticitat i/o risc tecnològic, detallats.
- Diagrama de classes de disseny.
- Diagrames de seqüència de disseny (els més rellevants).

Altrament, d'acord amb l'esperit del mètode UP s'ha d'entendre que aquesta anàlisi no és definitiva, i que pot ser canviada durant el desenvolupament per fer front a modificacions forçades per la tecnologia que s'usarà durant la construcció del programari. Cal no oblidar que una de les "feines" a fer és precisament aprendre les possibilitats del J2EE i tecnologies associades, i el que es vagi aprenent pot tenir el seu reflex descobrint alternatives d'anàlisi i/o disseny millors o més eficients. En qualsevol cas, les modificacions que es facin posteriorment seran reflectides en els documents corresponents i, en últim extrem, a la memòria del projecte.

2.2. Domini del projecte.

En aquest apartat establim quins són els límits del nostre desenvolupament. Bàsicament indicarem fins on arribarem en el desenvolupament, i quines possibles funcionalitats en quedaran explícitament fora.

Una aplicació de *vendes* (no necessàriament electròniques) pot arribar a abastar molts temes, i pot arribar, per tant, a ser molt ambiciosa. Només cal pensar en la complexitat que pot arribar a tenir tot el tema comptable, o tota la liquidació d'impostos. Com que aquest Projecte no deixa de ser un treball universitari, i donat que hi ha una limitació evident en els recursos de que es disposa (una força de treball total d'una persona sense dedicació completa per fer anàlisi, disseny i implementació) no tenim altra opció que limitar l'abast de l'aplicació al tema central: la implementació d'unes determinades funcionalitats usant J2EE.

En el document de planificació ja es feia una llista de tasques més o menys elementals en que es dividirà el desenvolupament. En aquest apartat farem una relació *negativa* (el que **no** farem en cap cas) de funcionalitats.

- **No farem cap consideració de tipus comptable**. Tanmateix, sí que guardarem un diari de les vendes efectuades amb la corresponent informació econòmica: donem per suposat que en una iteració posterior aquestes dades seran usades per alimentar alguna aplicació de comptabilitat externa.
- **No farem cap tractament corresponent al IVA, ni a altres taxes o impostos**. Igual que en el punt anterior, donarem per suposat que aquest tema es tractarà posteriorment. Per ser més concrets, **suposarem que la nostra botiga electrònica declara sempre preus nets**, taxes incloses.
- **No contemplarem la problemàtica de la tramesa dels articles comprats**. No calcularem despeses de tramesa ni contemplarem opcions a aquest respecte.
- **No contemplarem el pagament online**. Suposarem que els pagaments es fan per una altra via (transferència bancària, a la recepció de la mercaderia, etc...).
- **No contemplarem criptografia**. El nostre lloc web no usarà SSL. Totes les dades (incloses les paraules-clau) es transmetrà per la xarxa en clar. Gestionar un lloc segur seria un Projecte en sí mateix.
- **No contemplarem gestió de paraules-clau**. L'usuari escollirà una paraula-clau al registrar-se. Aquesta paraula-clau es desarà en forma de HASH a les nostres bases de dades, i en cas d'oblit caldrà que l'administrador la restauri.

2.3. Casos d'ús.

2.3.1. Relació de casos d'ús.

Assignarem un identificador per cada cas d'us, que tindrà la forma UCxnnn, on:

- La "x" valdrà F per als casos d'ús del frontal, i B per al backend.
- Les "nnn" seran un identificador numèric.

A més de la seva identificació cada cas d'ús tindrà un nom significatiu, que serà una frase curta indicativa de què contempla el cas d'ús.

En aquest moment no analitzarem les extensions dels casos d'ús (amb excepció dels casos d'ús que considerem prou crítics com per fer-ne una anàlisi detallada). Cada cas d'ús serà descrit en un paràgraf narratiu curt.

1. Frontal.

- **UCF010 – Connexió al sistema.** El client introdueix el seu identificador d'usuari i la seva paraula-clau. El sistema valida que l'usuari existeix i que la paraula-clau és correcta i, si és així, crea una sessió per a l'usuari.
- **UCF020 – Registre de nou usuari.** El client introdueix les seves dades personals i la seva adreça, així com una paraula-clau. L'identificador d'usuari serà en tots els casos la seva adreça de correu electrònic. El sistema valida que l'usuari no existeixi i, si és així, el dóna d'alta al sistema.
- **UCF030 – Navegació per la botiga.** El sistema presenta al client (que s'ha hagut de connectar prèviament) una relació dels productes a la venda. El client pot:
    - Filtrar la llista per categories.
    - Buscar un producte per nom
    - Afegir unitats d'un producte al seu carret de compra virtual.
    - Cancel·lar totalment la compra.
    - Passar a gestionar el seu carret de la compra.
- **UCF040 – Gestió del carret de la compra**. El sistema presenta al client el contingut del seu carret de la compra. El client pot:
    - Tornar a la navegació per la botiga (UCF030).
    - Cancel·lar totalment la compra.
    - Suprimir parcial o totalment la compra d'un article.
    - Confirmar i finalitzar la compra.

2. Backend

- **UCB010 – Gestió d'usuaris.** El sistema presenta a l'administrador una llista d'usuaris. L'administrador pot:
    - Filtrar la llista per dades de l'usuari.
    - Afegir un nou usuari.
    - Seleccionar un usuari i editar-lo.
    - Donar de baixa un usuari.
- **UCB020 – Gestió de categories.** El sistema presenta a l'administrador una llista de categories. L'administrador pot:
    - Filtrar la llista per nom parcial de categoria.
    - Afegir una nova categoria.
    - Seleccionar una categoria i editar-la.
    - Donar de baixa una categoria.
- **UCB030 – Gestió de productes.** El sistema presenta a l'administrador una llista de productes. L'administrador pot:
    - Filtrar la llista per dades del producte, incloent-hi la categoria.
    - Afegir un nou producte.
    - Afegir unitats d'estoc a un producte.

- Seleccionar un producte i editar-lo (exceptuant l'estoc).
- Seleccionar un o més d'un producte i modificar-ne el preu.
- Donar de baixa un producte.

- **UCB040 – Llistat de vendes diàries.** El sistema genera un llistat amb la relació de vendes d'un període determinat. La llista generada és enviada per correu electrònic a l'administrador.
- **UCB050 – Llistat d'alertes d'estoc.** El sistema genera un llistat amb la relació de productes amb estoc baix. La llista generada és enviada per correu electrònic a l'administrador.
- **UCB060 – Neteja de dades.** El sistema dóna de baixa físicament les sessions i carrets de compra que porten més d'un cert temps inactius.

2.4. Detall dels casos d'ús seleccionats.

Els casos d'ús que poden merèixer un tractament més detallat en aquesta etapa del projecte són, sense cap mena de dubtes, UCF030 (Navegació per la botiga) i UCF040 (Gestió del carret de compra). La resta de casos són sempre repeticions del mateix model: actualitzacions d'un conjunt de dades (altes, baixes, modificacions i consultes), que no han de presentar en principi dificultats de plantejament ni de desenvolupament.

L'operativa de compra, pel contrari, planteja més dificultats i, a més, és la funcionalitat central a la qual va adreçada aquest projecte: tota la resta de funcionalitats són accessòries a permetre realitzar vendes online.

2.4.1. UCF030 – Navegació per la botiga.

Nom del Cas d'Ús : Navegació per la botiga		**Codi**: UCF030
Autor	Jordi Guillaumes Pons	
Descripció	Interacció del client-usuari amb la llista d'articles a la venda	
Prioritat (1.. 10) 1 com a màxim. 10 com a mínim.	3	
Actor/s	Client	
Precondicions:	**Casos d'Ús**	El client s'ha autentificat al sistema
	D'altres	
Postcondicions		
Flux normal	1. El sistema recupera les preferències de cerca de l'usuari. Aquestes preferències poden estar a una *cookie* desada a l'ordinador de l'usuari. 2. El sistema presenta la primera pàgina d'articles a la venda recuperats segons les preferències de cerca de l'usuari. 3. L'usuari procedeix a navegar per la botiga. Les opcions possibles són avançar pàgina, retrocedir pàgina, seleccionar una categoria, seleccionar un article (afegir-lo al carret de la compra) 4. L'usuari indica que ha acabat la navegació i procedeix a efectuar la compra.	

Extensions	1-2a. Si no hi ha preferències guardades, el sistema presenta la primera pàgina de la llista de tots els articles. 3a. Si l'usuari selecciona "avançar pàgina", el sistema mostra la següent pàgina d'articles d'acord amb els paràmetres de cerca. 3b. Si l'usuari selecciona "retrocedir pàgina", el sistema mostra la pàgina anterior (si n'hi ha). 3c. Si l'usuari selecciona "Seleccionar categoria", el sistema mostra la primera pàgina d'articles de la categoria seleccionada. 3d. Si l'usuari selecciona "comprar article", el sistema afegeix l'article seleccionat al carro de la compra. El nombre d'unitats que s'afegeixen dependrà de l'article.

2.4.2. UCF040 – Gestió del carret de compra.

Nom del Cas d'Ús : Gestió del carret de compra		**Codi**: UCF040
Autor	Jordi Guillaumes Pons	
Descripció	Interacció del client amb el contingut del carret de compra	
Prioritat (1.. 10) 1 com a màxim. 10 com a mínim.	3	
Actor/s	Client	
Precondicions:	**Casos d'Ús**	El client s'ha autentificat al sistema El client està dins d'una sessió de compres.
	D'altres	
Postcondicions		
Flux normal	1. El sistema presenta a l'usuari el contingut del carret de la compra. Davant de la llista l'usuari pot escollir entre confirmar la compra, tornar a la navegació, descartar el carret o bé modificar el nombre d'unitats comprades de qualsevol article del carret. 2. Un cop confirmada la compra, el sistema presenta a l'usuari un resum final del preu i demana la confirmació final. 3. Un cop donada la confirmació final, el sistema actualitza els estocs i enregistra la compra. 4. El sistema presenta a l'usuari el contingut del carret de la compra. Davant de la llista l'usuari pot escollir entre confirmar la compra, tornar a la navegació, descartar el carret o bé modificar el nombre d'unitats comprades de qualsevol article del carret. 5. Un cop confirmada la compra, el sistema presenta a l'usuari un resum final del preu i demana la confirmació final. 6. Un cop donada la confirmació final, el sistema actualitza els estocs i enregistra la compra.	

Extensions	1a. Si l'usuari selecciona tornar a la navegació, s'entra de nou al cas d'us UCF030 amb el carret tal com estava a l'abandonar-la. 1b. Si l'usuari selecciona descartar el carret, s'elimina tot el seu contingut i es tanca la sessió de compra. 1c. Si l'usuari modifica les unitats comprades d'algun article, i les noves unitats són superiors a zero, es modifica l'ítem corresponent i es mostra el carret de la compra actualitzat. 1d. Si l'usuari modifica les unitats comprades d'un article, i el nou nombre d'unitats és zero, s'elimina l'ítem del carret de la compra i es mostra de nou el contingut del carret. 2a. Si l'usuari no confirma la compra, es torna a presentar la llista del contingut del carret.

2.4.3. Diagrama de casos d'ús.

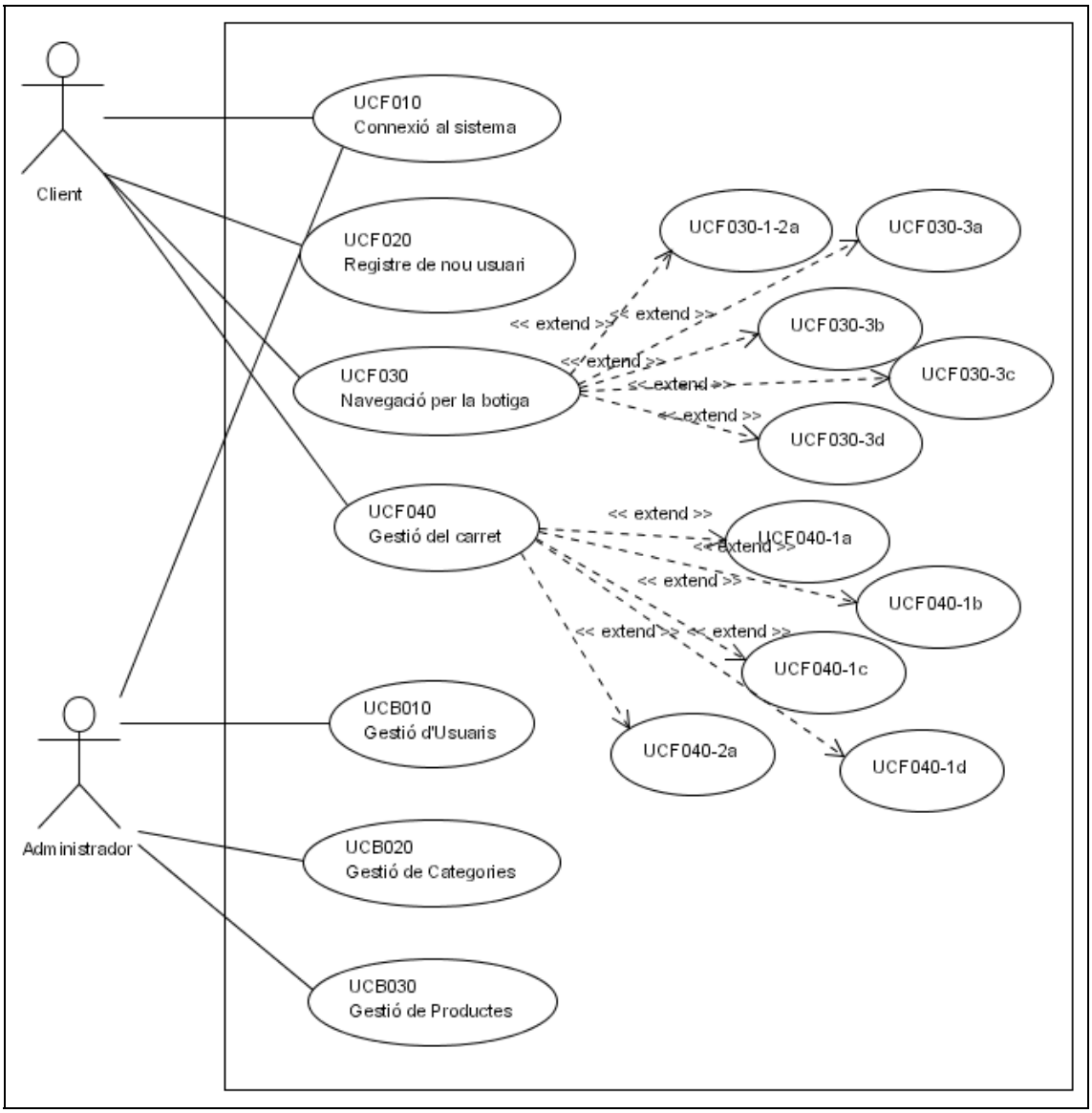

Figura 1 - Diagrama de casos d'us

2.5. Classes del domini.

En aquesta iteració no s'aprofundirà en el disseny complet de les classes. Únicament es definiran les classes principals, i alguns dels seus atributs i operacions, així com les relacions entre les classes. **No** es definiran les classes de suport, ni les corresponents a la implementació J2EE (servlets, EJB, etc.).

2.5.1. Diagrama de classes del domini.

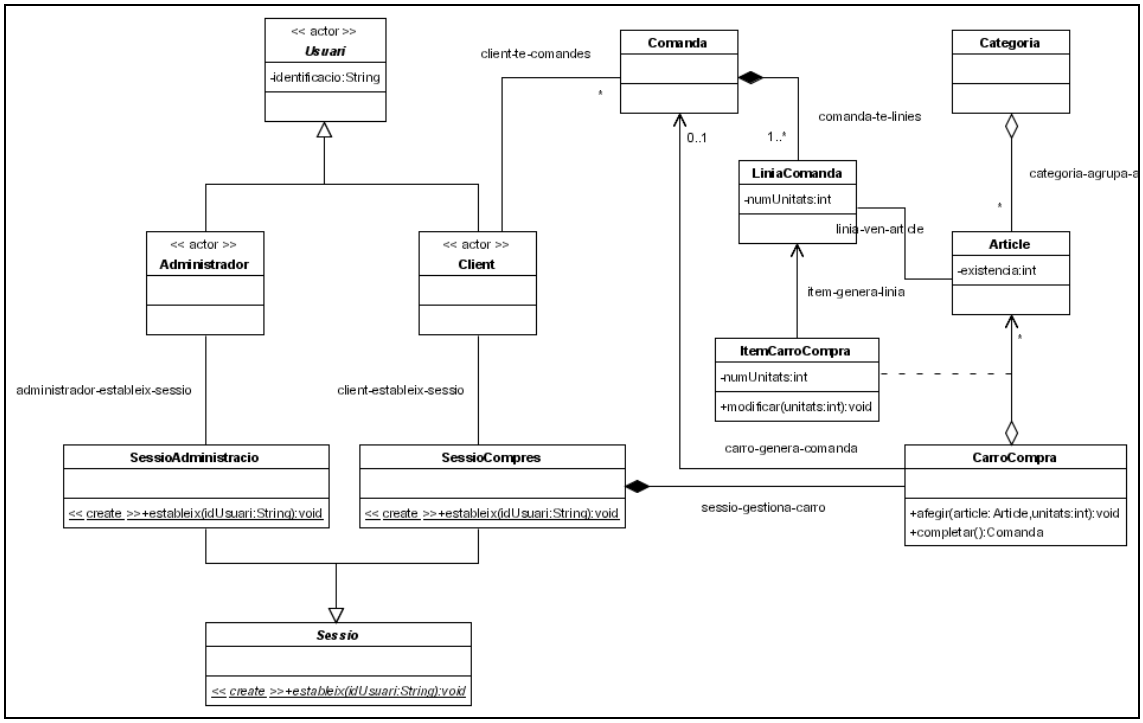

Figura 2 - Diagrama de classes del domini

2.5.2. Descripció breu de les classes.

- **Usuari**. Classe abstracta. Se'n deriven les classes Administrador i Client, que representen, respectivament, un usuari administrador del sistema i un client registrat de la botiga.
- **Administrador**. Usuari amb privilegis d'administrador de la botiga virtual.
- **Client**. Usuari final de la botiga. Correspon a un usuari registrat que té permís per fer compres a la botiga virtual.
- **Sessió**. Classe abstracta, que conté les funcionalitats necessàries per establir una sessió al sistema. Se'n deriven les classes SessioAdministracio i SessioCompres.
- **SessioAdministracio**. Conté les funcionalitats relacionades amb les sessions que estableixen els administradors de la botiga virtual.
- **SessioCompres**. Conté les funcionalitats relacionades amb l'operativa de compres.
- **Categoria**. S'usa per classificar i agrupar els articles a la venda.
- **Article**. Ítem a la venda a la botiga virtual. A més dels atributs que permeten identificar-lo, conté la resta de dades necessàries, com ara el preu, la unitat de venda i l'existència.
- **CarroCompra**. Classe auxiliar (instanciada i gestionada per SessioCompres) que conté l'estat de la compra que fa el client.
- **LiniaCarroCompra**. Classe associativa, que relaciona CarroCompra i Article. Permet mantenir el nombre d'unitats de cada un dels articles que s'està comprant.
- **Comanda**. Classe que representa una comanda en ferm feta i confirmada per un client. Una comanda s'origina sempre a partir d'un CarroCompra.
- **LiniaComanda**. Cada un dels ítems que formen part d'una Comanda. Les ocurrències de LiniaComanda s'originen a partir d'ocurrències de LiniaCarroCompra.

2.6. Anàlisi dinàmica.

Presentarem a continuació els DSS (Diagrames de Seqüència del Sistema) corresponents als casos d'ús UCF030 i UCF040.

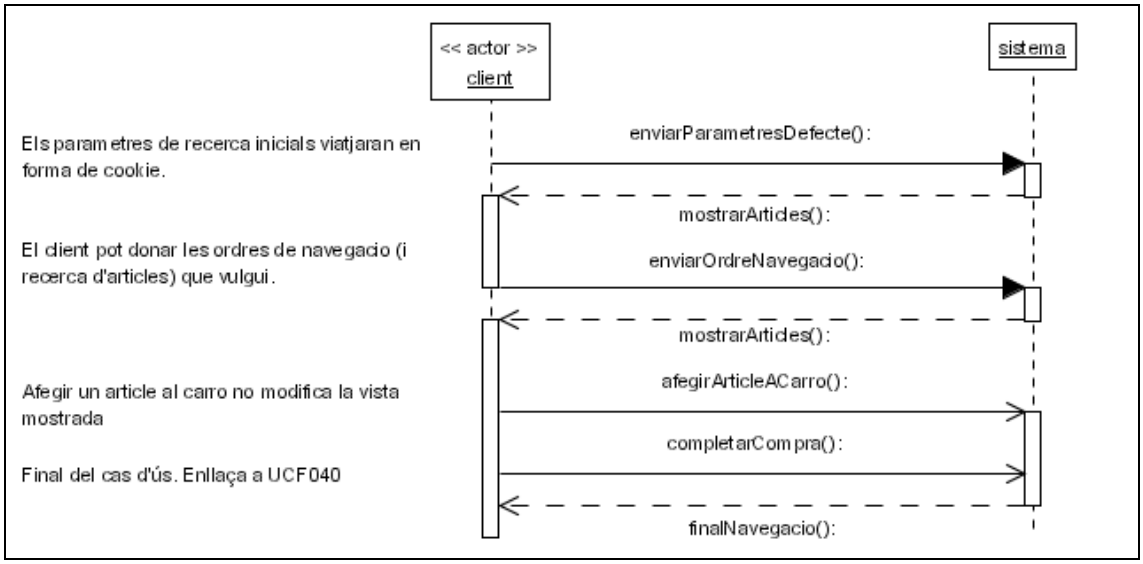

Figura 3 - DSS per al cas d'ús UCF030

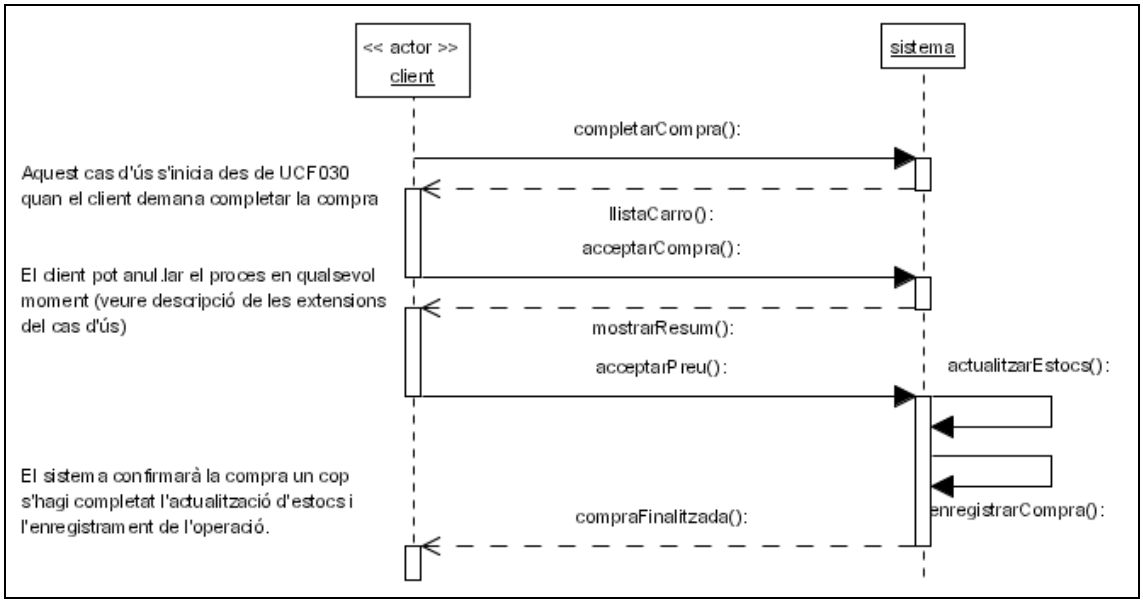

Figura 4 - DSS per al cas d'ús UCF040

3. Disseny.

3.1. Presentació

En aquest document presentarem el resultat de la segona iteració en el desenvolupament del TFC, que es centrarà sobre tot en la fase d'elaboració, tal com s'entén en el context de l'Unified Process.

Ens centrarem, així, en completar les descripcions dels casos d'ús exposats en el document d'anàlisi i en estudiar les interaccions entre els diferents objectes, mitjançant diagrames de col·laboració i de seqüència.

A partir d'aquest treball, completarem el model estàtic dissenyant les classes de disseny, definides completament a nivell d'atributs i operacions. A l'hora determinarem quin o quins patrons de disseny són d'aplicació en aquest projecte, i afegirem al model les classes necessàries per implementar-los.

3.2. Cassos d'us detallats

Detallarem a continuació la resta de casos d'us que s'havien definit de forma breu en el document d'anàlisi.

3.2.1. UCF010 – Connexió al sistema.

Nom del Cas d'Ús : Connexió al sistema.		**Codi**: UCF010
Autor	Jordi Guillaumes Pons	
Descripció	Identificació i autentificació d'un usuari.	
Prioritat (1.. 10) 1 com a màxim. 10 com a mínim.	3	
Actor/s	Client, Administrador	
Precondicions:	**Casos d'Ús**	Cap
	D'altres	
Postcondicions	S'ha establert una nova sessió al sistema	
Flux normal	1. L'usuari introdueix el seu identificador, la paraula clau i prem el botó corresponent per enviar les dades. 2. El sistema verifica l'existència de l'usuari, i la paraula clau teclejada. 3. El sistema presenta a l'usuari la pantalla inicial de l'aplicació, o bé la pantalla que l'usuari havia demanat abans d'identificar-se. 4. L'usuari introdueix el seu identificador, la paraula clau i prem el botó corresponent per enviar les dades. 5. El sistema verifica l'existència de l'usuari, i la paraula clau teclejada. 6. El sistema presenta a l'usuari la pantalla inicial de l'aplicació, o bé la pantalla que l'usuari havia demanat abans d'identificar-se.	
Extensions	1a. Si l'usuari no existeix i opta per registrar-se es bifurca al cas d'us UCF020. 3a. Si l'usuari no existeix, o la paraula-clau és incorrecta, es mostra de nou la pantalla d'identificació.	

Comentaris.

La identificació es durà a terme usant les funcionalitats definides a les especificacions J2EE. Concretament, quan l'usuari sol·liciti una funció protegida (amb restriccions de seguretat definides), el servidor mostrarà un formulari, que usarà l'acció j_security_check per validar les dades introduïdes. Si la identificació és correcta, es mostrarà la pantalla o funció que havia sol·licitat inicialment l'usuari.

3.2.2. UCF020 – Registre de nou usuari.

Nom del Cas d'Ús : Registre de nou usuari		**Codi**: UCF020
Autor	Jordi Guillaumes Pons	

Descripció	Auto-registre d'un nou usuari com a client	
Prioritat (1.. 10) 1 com a màxim. 10 com a mínim.	3	
Actor/s	Client, Administrador	
Precondicions:	**Casos d'Ús**	Cap
	D'altres	
Postcondicions	S'ha donat d'alta un nou client al sistema	
Flux normal	1. L'usuari omple un formulari amb les dades següents: • Nom i cognoms. • Adreça (física), codi postal, ciutat i país. • Adreça electrònica. • Paraula-clau. • Verificació de paraula-clau. 2. El sistema verifica que la paraula clau i la verificació de paraula clau coincideixen. 3. El sistema crea un nou usuari, de tipus Client, usant l'adreça de correu com a identificador. 4. El sistema envia un correu electrònic a l'adreça indicada confirmant l'alta. 5. El sistema presenta la pantalla inicial de l'aplicació.	
Extensions	2a. Si les paraules-clau no coincideixen, el sistema torna a presentar la pantalla d'entrada de dades. 3a. Si ja existeix un usuari amb aquest identificador (adreça de correu electrònic), el sistema torna a presentar la pantalla d'entrada de dades.	

Comentari

L'escollir l'adreça de correu com a identificador ens permet obviar coses com podrien ser suggerir un identificador en funció de nom i cognoms.

3.2.3. UCB010 – Gestió d'usuaris.

Nom del Cas d'Ús : Gestió d'usuaris		**Codi**: UCB010
Autor	Jordi Guillaumes Pons	
Descripció	Gestió de la llista d'usuaris per part de l'administrador.	
Prioritat (1.. 10) 1 com a màxim. 10 com a mínim.	5	
Actor/s	Administrador	
Precondicions:	**Casos d'Ús**	L'administrador s'ha autentificat al sistema
	D'altres	
Postcondicions	S'han enregistrat les modificacions a la llista d'usuaris	
Flux normal	1. El sistema presenta una llista d'usuaris. Per cada usuari es mostra l'identificador, el nom i els tipus d'usuari (administrador/client) actius. 2. L'administrador opta per l'acció a efectuar: filtrar, afegir, modificar o eliminar usuaris. 3. El sistema mostra la llista actualitzada i, si s'escau, filtrada.	

Extensions	2a. L'administrador selecciona afegir: 1. El sistema presenta una pantalla on es pot entrar tota la informació corresponent al nou usuari, incloent-hi els rols (client/administrador) que ha de tenir. 2. El sistema afegeix el nou usuari al sistema. 2b. L'administrador selecciona modificar: 1. El sistema presenta una pantalla amb les dades de l'usuari seleccionat. 2. L'administrador pot modificar qualsevol dada (incloent-hi els rols) amb excepció de l'identificador. 3. El sistema actualitza les dades segons les modificacions fetes per l'administrador. 2c. L'administrador selecciona eliminar: 1. El sistema presenta una pantalla de confirmació, amb les dades bàsiques (identificador, nom, rols) de l'usuari seleccionat. 2. L'administrador confirma la selecció. 3. El sistema elimina l'usuari seleccionat, sempre i quan no hi hagi dades històriques (comandes) associades.

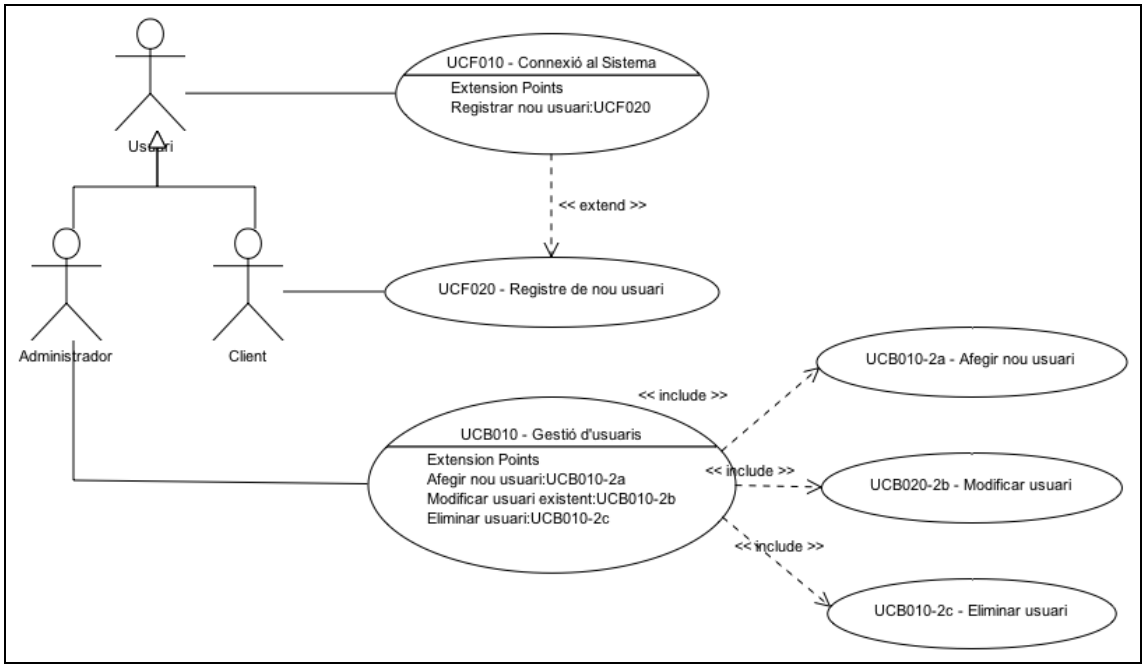

Figura 5 - Casos d'ús relatius a gestió d'usuaris i sessions

3.2.4. UCB020 – Gestió de categories.

Nom del Cas d'Ús : Gestió de categories		**Codi**: UCB020
Autor	Jordi Guillaumes Pons	
Descripció	Gestió de la llista de categories de producte a la venda	
Prioritat (1.. 10) 1 com a màxim. 10 com a mínim.	5	
Actor/s	Administrador	
Precondicions:	**Casos d'Ús**	L'administrador s'ha autentificat al sistema
	D'altres	

Postcondicions	S'han enregistrat les modificacions a la llista de categories
Flux normal	1. El sistema presenta una llista de categories seleccionables (només s'admet una selecció). 2. L'usuari (administrador) opta per l'acció a efectuar: filtrar, afegir, modificar o eliminar categories. 3. El sistema mostra la llista actualitzada, aplicant-hi si hi és el filtre establert per l'usuari.
Extensions	2a. L'usuari selecciona afegir: 1. El sistema presenta una pantalla on l'usuari pot entrar el codi i el nom de la nova categoria. 2. El sistema afegeix la categoria al sistema. 2b. L'usuari selecciona modificar: 1. El sistema presenta una pantalla on l'usuari pot modificar el nom de la categoria seleccionada. 2. El sistema actualitza el nom de la categoria. 2c. L'usuari selecciona eliminar: 1. El sistema presenta una pantalla de confirmació amb la categoria seleccionada. 2. L'usuari confirma la selecció. 3. El sistema elimina la categoria seleccionada, sempre i quan no tingui cap producte associat.

3.2.5. UCB030 – Gestió de Productes.

Nom del Cas d'Ús : Gestió de productes		**Codi**: UCB030
Autor	Jordi Guillaumes Pons	
Descripció	Gestió de la llista de productes a la venda	
Prioritat (1.. 10) 1 com a màxim. 10 com a mínim.	4	
Actor/s	Administrador	
Precondicions:	**Casos d'Ús**	L'administrador s'ha autentificat al sistema
	D'altres	
Postcondicions	S'han enregistrat les modificacions a la llista de productes	
Flux normal	1. El sistema mostra una llista de productes. En principi, es mostren tots ells, sense cap filtre. Per cada producte es mostra l'identificador, l'identificador de la categoria a la que està adscrit, el nom, el preu i l'estoc disponible. 2. L'administrador pot optar per fer manteniment "convencional" (alta, baixa, modificació), o bé específica (gestió d'estocs i modificacions massives de preu). També pot optar per filtrar la llista per nom parcial de producte i/o categoria. 3. El sistema mostra la llista actualitzada, aplicant-hi si s'escau el filtre determinat per l'administrador.	

Extensions	2a. L'administrador selecciona crear un nou producte. 1. El sistema presenta una pantalla en la que l'administrador ha d'especificar les dades corresponents al nou producte: • Identificador del producte. • Categoria a la que quedarà assignat. • Nom del producte. • Descripció textual. • Preu. • Estoc mínim. • Estoc inicial. 2. El sistema comprova que el nou producte no sigui un duplicat i que la categoria indicada existeixi, i si és així, afegeix el nou producte a l'inventari. 2b. L'administrador selecciona modificar un producte existent. 1. El sistema verifica que l'administrador ha seleccionat únicament un producte. 2. El sistema presenta una pantalla amb les dades corresponents al producte seleccionat. 3. L'administrador pot modificar les dades següents: • Categoria. • Nom del producte. • Descripció. • Preu. • Estoc actual. • Estoc mínim. 4. El sistema verifica que la categoria sigui correcta, i si és així, actualitza les dades segons les modificacions efectuades per l'administrador. 2c. L'administrador selecciona donar de baixa un producte existent. 2e. L'administrador selecciona fer modificacions massives de preus.

3.3. Disseny.

3.3.1. Patrons de disseny a aplicar.

Un dels pressupòsits en aquest projecte és l'ús de l'arquitectura J2EE per implementar les seves funcionalitats. Aquesta arquitectura proporciona un conjunt d'infraestructures per servir les diferents necessitats de les aplicacions de tipus transaccional. Això, lògicament, condiciona estructura de control i d'interfície que hem d'implementar.

J2EE és una arquitectura que té una certa maduresa, arrel de la qual han sorgit diferents recomanacions (*best practices*) a tenir en compte; en aquest projecte haurem de valorar també el fet de que es tracta d'un problema acadèmic, i que un dels seus objectius és la *formació* de l'estudiant en l'ús d'aquestes tecnologies.

Tenint en compte això, podem escollir els patrons de disseny que seguirem, tant en el que respecta a la presentació i control, com a la persistència de dades.

1.1.1.1 Arquitectura de presentació i control.

Basarem la presentació (pantalles d'entrada/sortida) i el control (flux de pantalles) en el conegut model **MVC (Model-View-Controller)**. Concretament, ens basarem en la infraestructura struts, que proporciona la part de controlador, i eines per construir la presentació (vistes).

Així, totes les vistes seran generades usant components JSP, *darrera* de les quals hi haurà els corresponents Form Beans, que invocaran Action Classes per dur a terme la seva lògica, que serà controlada pel servlet Action, proporcionat per struts.

1.1.1.2 Model (gestió de persistència i lògica de negoci).

La part del Model dins del patró MVC es basarà en l'ús d'EJB d'entitat, usant persistència manegada per contenidor (CMP) sempre que sigui possible.

Es procurarà fer que els EJB d'entitat siguin com més simples millor. Idealment, definiran únicament els mètodes accessors i la lògica de negoci més bàsica. Per definir la resta de lògica de negoci ens basarem en el patró **façana de sessió** (session façade). Els controladors de façana s'implementaran en forma de EJB de sessió sense estat. En la nostra aplicació definirem dues façanes:

- **FacanaInventari** encapsularà les funcions relatives a la gestió i consulta sobre els objectes d'inventari: categories i articles a la venda.
- **FacanaUsuari**: encapsularà les funcions relatives a la gestió i ús dels usuaris i dels privilegis dels mateixos.

Els controladors de façana presentaran a la capa de presentació els objectes persistents en forma d'**objectes-valor**[2]**.** L'ús d'objectes-valor simplifica l'accés als atributs dels objectes persistents, ja que els accessos es fan sobre objectes "rics" i no sobre atributs individuals. En un entorn distribuït això permet estalviar tràfic de xarxa, al no haver de fer un accés remot per cada atribut consultat (cosa que estalvia també l'haver d'estar serialitzant i deserialitzant els atributs un per un). Tanmateix, té un inconvenient: en realitat estem transferint *còpies dels objectes persistents*, i no referències als mateixos. Això suposa que si en un moment determinat hi hagués 100 clients connectats a l'aplicació tindríem 100 còpies de cada un dels objectes compartits (com ara les instàncies dels elements del catàleg). Això fa que l'aplicació sigui poc escalable i que, en cas que es volés fer una implementació real i no acadèmica fos necessari reconsiderar aquesta part (sobre tot en el que respecta a les llistes de valors per paginar, que haurien de convertir-se en llistes de referències en lloc de llistes de còpies).

Hem de considerar també que haurem de manegar tot un seguit d'objectes volàtils (no persistents), que corresponen a tot el que penja d'una sessió d'usuari – especialment d'una sessió de compra. Concretament, això es refereix als objectes de les classes CarroCompra i ItemCarroCompra. Aquests objectes només viuen mentre dura la sessió de compra. En el moment en que aquesta finalitza, la instància de CarroCompra haurà generat una instància de Comanda, i cadascun dels ItemsCarroCompra haurà generat una LiniaComanda. El CarroCompra s'implementarà en forma de EJB de sessió amb estat, mentre que el ItemCarroCompra serà una classe pura que, a tots els efectes, tractarem com un objecte-valor.

3.3.2. Diagrama de classes de disseny..

El diagrama de classes de disseny és bàsicament el mateix que es va presentar a la fase d'anàlisi. S'han fet, però, alguns canvis significatius:

- Ha desaparegut la diferència (a nivell de classes) entre Administrador i Client. Com que s'usarà l'arquitectura de seguretat del contenidor J2EE per mantenir els permisos no té sentit considerar dues classes separades: els rols d'administrador i de client són atributs d'una única classe. Això permet que un usuari pugui actuar com a administrador i com a client. A més, s'ajusta millor a les facilitats que dona el contenidor JBoss de cara a implementar la seguretat usant taules de base de dades.
- S'ha afegit els atributs i les operacions al diagrama.
- Ha aparegut una nova interfície (ControladorLlista) per facilitar gestionar llistes paginables en pantalla. Aquesta interfície es basa parcialment en el patró Value List Handler, amb la diferència de que les dades a paginar es desen a la sessió web en lloc de a un EJB de sessió. S'ha definit també una implementació d'aquesta interfície (ControladorLlistaList) basada en objectes de tipus List. En una hipotètica revisió de l'aplicació es podrien crear altres implementacions (per exemple, basades en guardar els elements a mostrar en una taula relacional).

Val la pena recordar que la decisió de disseny és usar Entity Beans per representar les entitats, i JSPs i Struts per construir la interfície d'usuari. Les particularitats d'aquestes implementacions (interfícies locals i remotes en el cas dels EJB, classes Action i ActionForm en el cas de struts) no es mostren en el diagrama de classes.

[2] En la documentació a la web de Sun Microsystems aquest patró ha estat rebatejat com a "Transfer Objects".

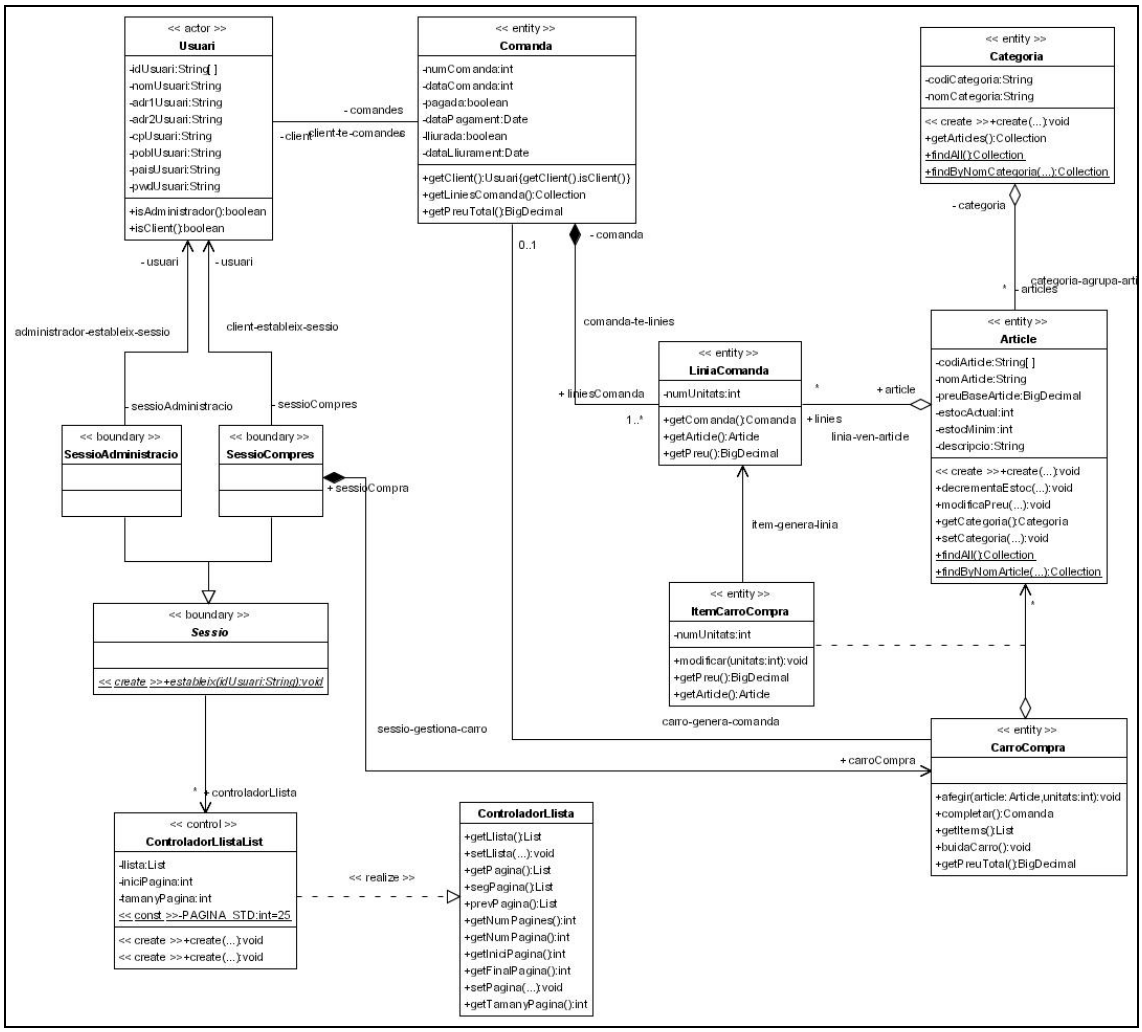

Figura 6 - Diagrama de classes de disseny

3.3.3. Model relacional i model de persistència.

La conversió al model relacional de les entitats que requereixen persistència és força directa i immediata. Si observem el diagrama de classes d'entitats veurem que, realment, només és necessària la persistència per:

- Categories.
- Articles.
- Comandes.
- Línies de comanda.
- Usuaris.

En tots els casos, amb l'excepció dels usuaris, la conversió a entitats relacionals és immediata. El cas dels usuaris no té altra complicació que afegir una taula de Rols per complir amb la primera forma normal. De fet, aquesta taula és necessària per poder usar el gestor de login basat en Datasources que incorpora el servidor JBoss.

Pel que respecta a la implementació de la persistència, s'ha optat per usar EJB d'entitat amb persistència gestionada pel contenidor (CMP) en tots els casos, exceptuant els Usuaris i Rols. La raó d'aquesta excepció és la percepció (que ha resultat errònia) de que l'ús de EJBs provocava diferiments en l'actualització de la informació a les taules relacionals que provocava que el gestor de logins no trobés les dades quan intentava validar un accés. En realitat, el problema està en que la implementació de JAAS de JBoss construeix una *cache* de credencials, els elements de la qual caduquen per temps; això provoca que les actualitzacions fetes sobre les taules no es vegin fins que les còpies corresponents a la cache han caducat. La solució és, òbviament, desactivar la cache o reduir el temps de caducitat, i és independent de l'ús o no d'EJB per mantenir les dades. Tanmateix,

com que quan he trobat aquest problema ja havia fet el desenvolupament sense EJB he preferit deixar-ho tal com està.

D'aquesta manera, la persistència s'ha resolt així:

- Per **Categoria, Article, Comanda i LiniaComanda** s'han creat EJB-CMPs amb els noms **CategoriaBean, ArticleBean, ComandaBean i LiniaComandaBean**.
- Per **Usuari i Rol** s'ha creat un únic DAO que encapsula les dues entitats i ofereix una visió equivalent a la presentada al disseny de classes d'Entitat (els rols, Administrador i Client, són atributs booleans de l'Usuari).

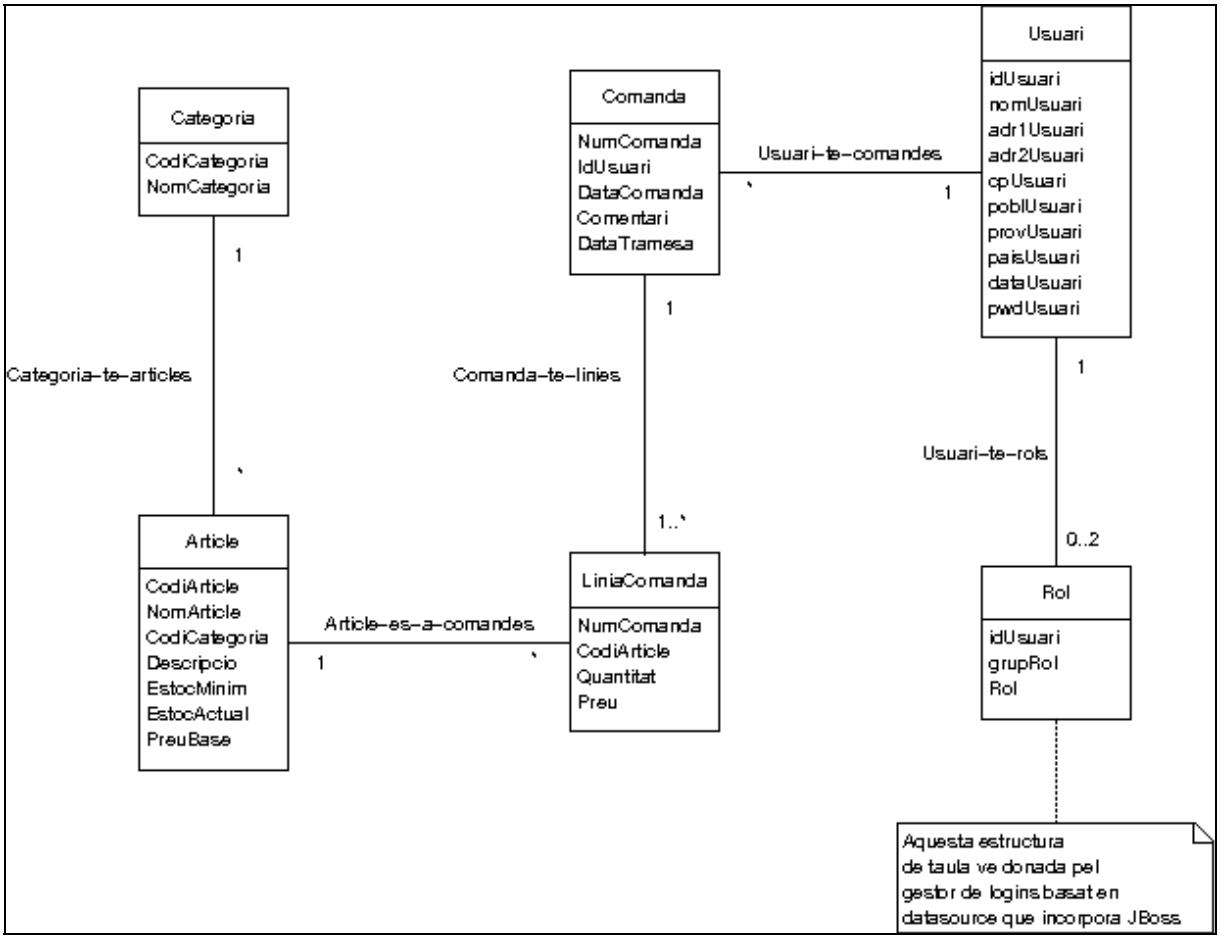

Figura 7 - Diagrama E/R

3.3.4. Diagrames de col·laboració

A títol d'exemple es presenten els diagrames de col·laboració lligats al cas d'ús UCB020 (gestió de categories). El primer diagrama correspon al refresc de la informació mostrada (que inclou la primera presentació de dades quan l'administrador selecciona la funció de gestió de categories al menú corresponent.

A tenir en compte:

- La interacció entre els diferents elements de control de la infraestructura struts no es representa amb exactitud.
- No es presenta tot el joc corresponent a la gestió dels objectes i interfases dels EJB.

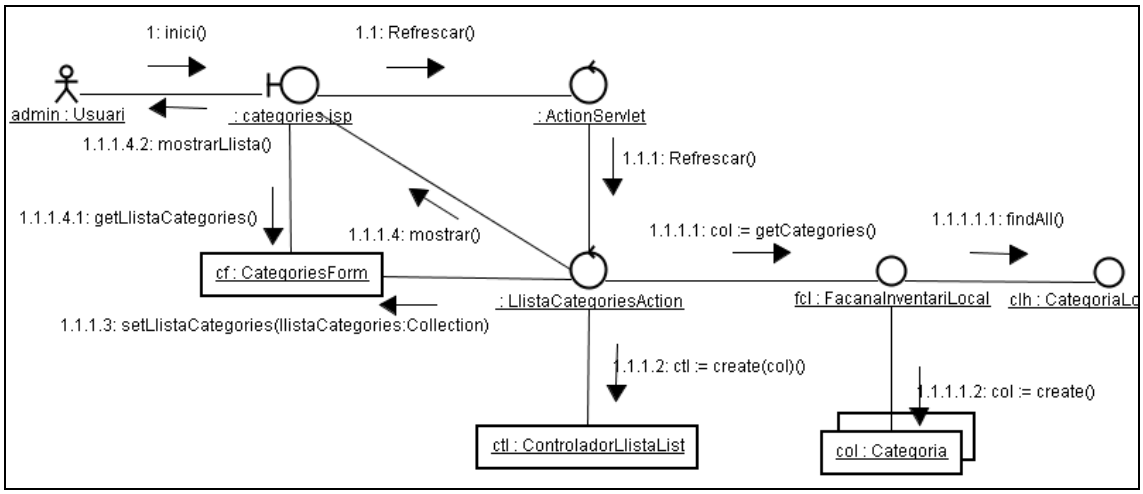

Figura 8 - Diagrama de col·laboració: refresc de llista de categories

El diagrama que segueix correspon a la creació d'una nova categoria. El diagrama acaba en el moment en que es crea la nova instància al contenidor EJB. Després d'això, el sistema fa un refresc de la pàgina mostrada, equivalent al mostrat a la figura 3.

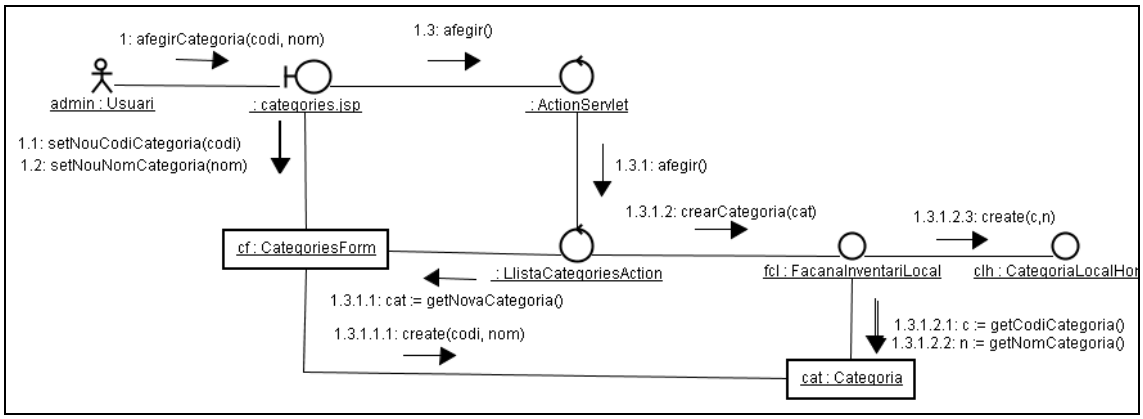

Figura 9 - Diagrama de col·laboració: creació d'una nova categoria.

A continuació es presenta el diagrama corresponent a la modificació d'una categoria existent. S'ha simplificat lleugerament el flux de missatges per permetre seguir el procediment.

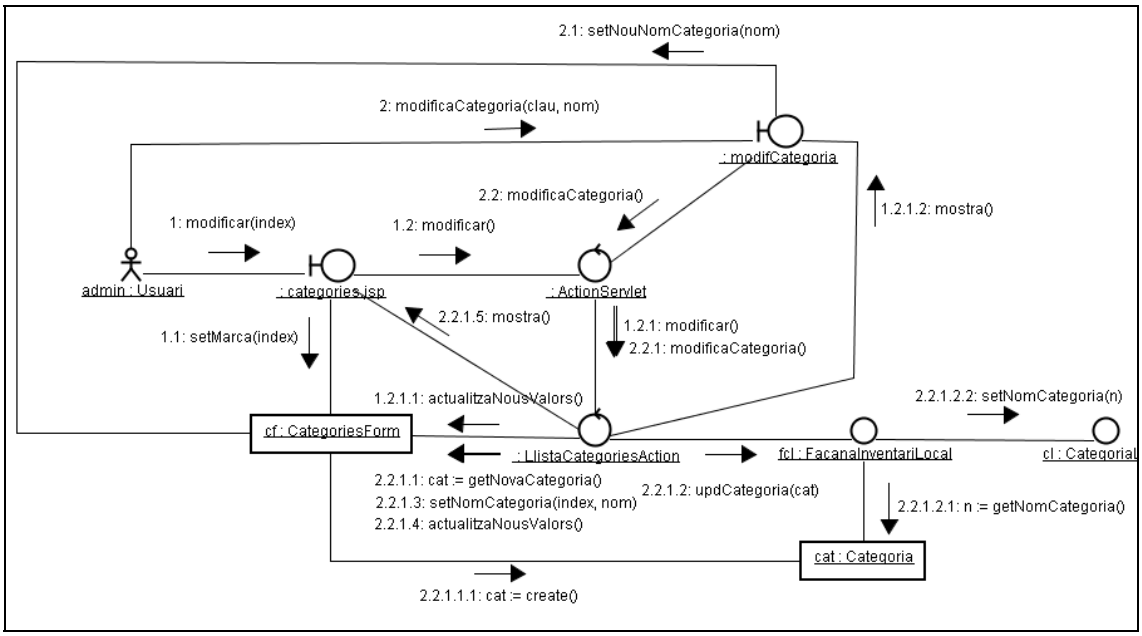

Figura 10- Diagrama de col·laboració: modificació d'una categoria

3.4. Exemples de pantalles

Es presenten a continuació, sense més comentaris, alguns exemples de pantalles d'entrada de dades, corresponents a les funcions d'administració de l'inventari (categories i articles).

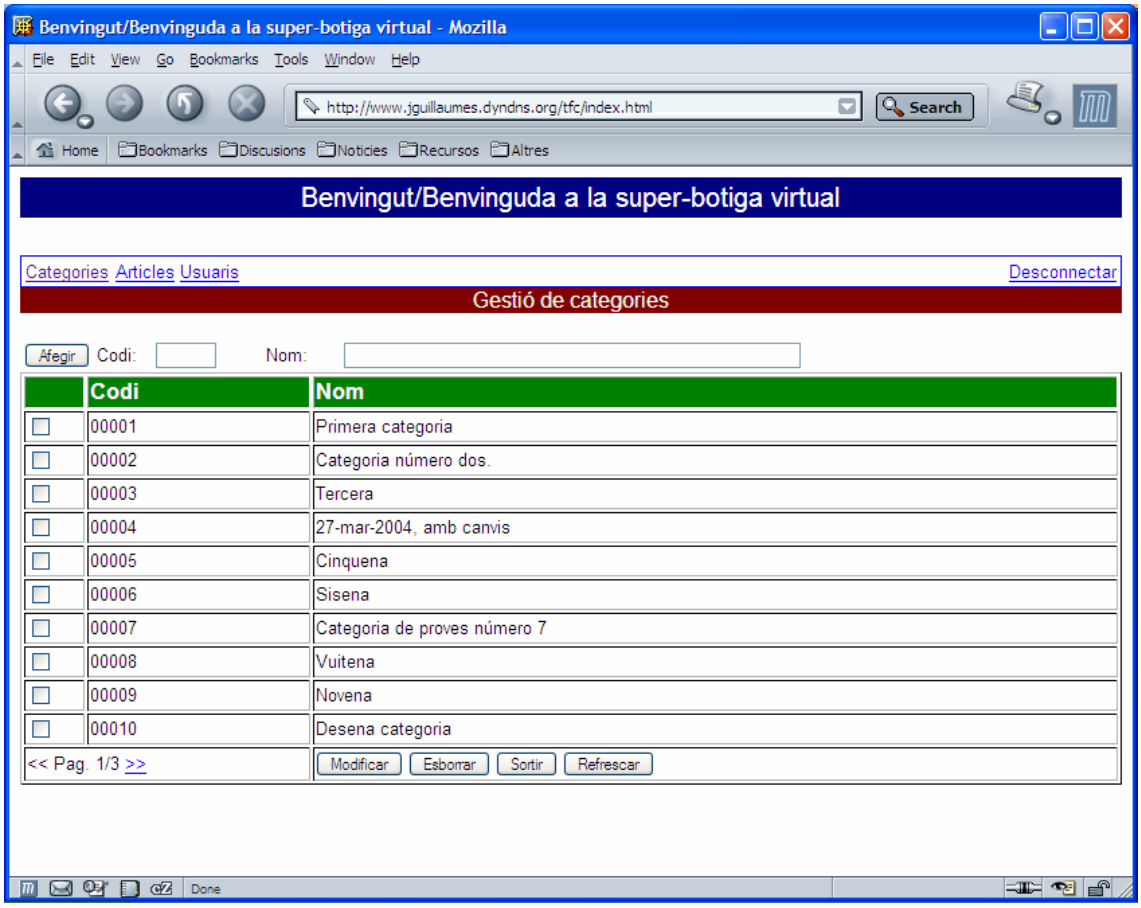

Figura 11 - Pantalla de gestió de categories

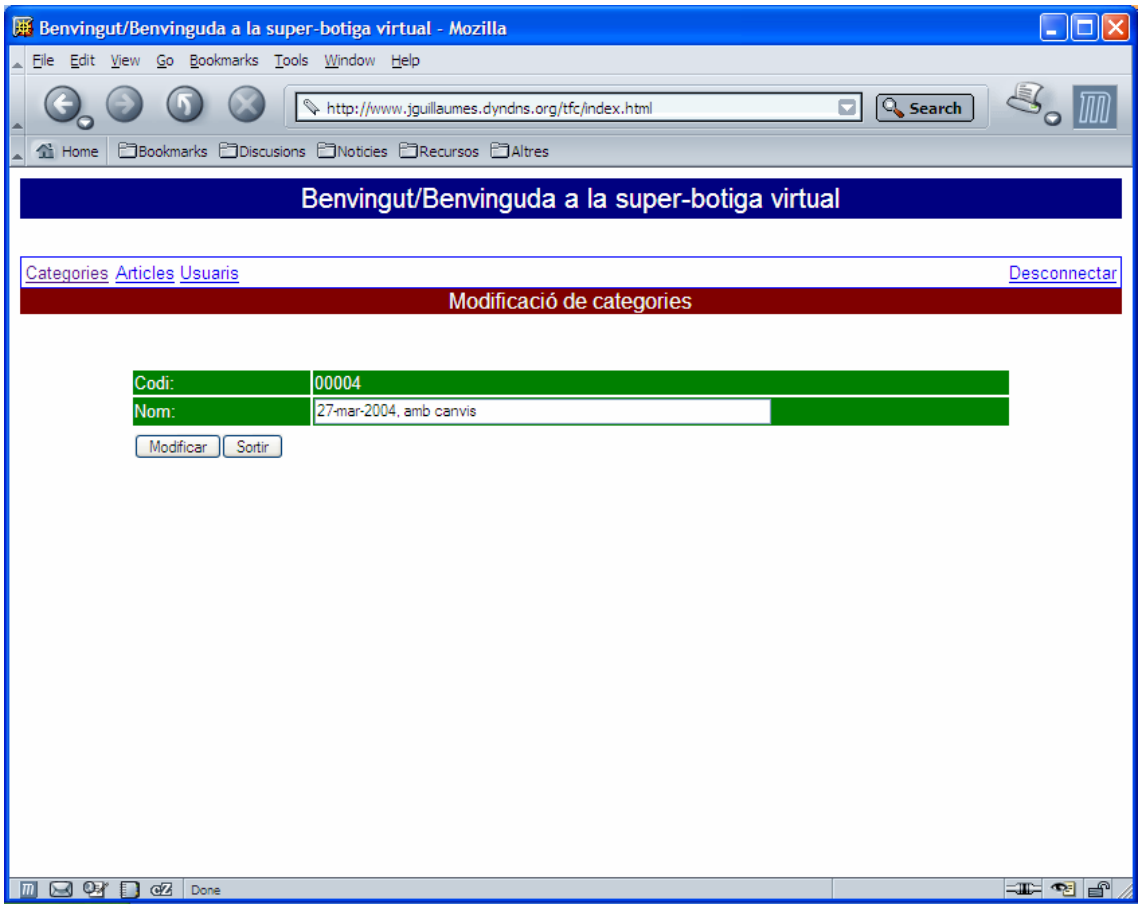

Figura 12 - Pantalla de modificació de categories

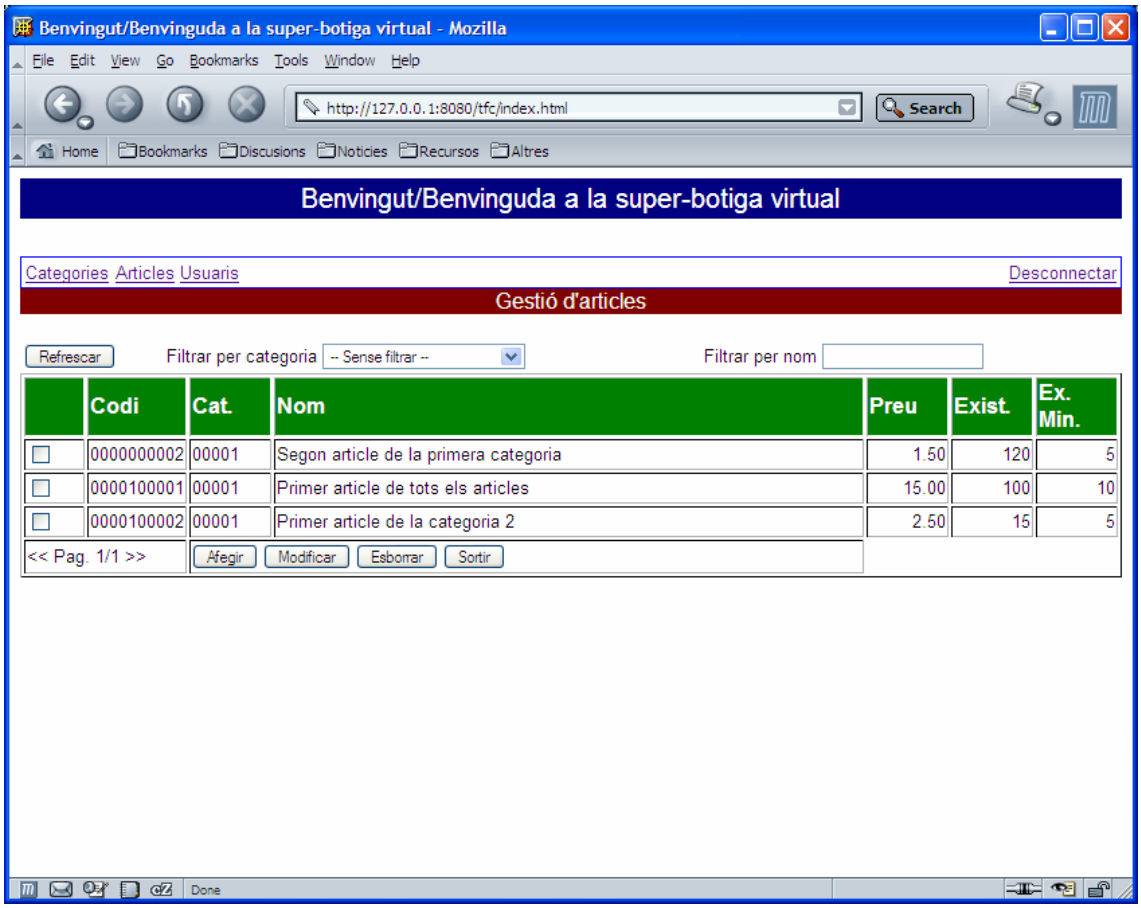

Figura 13 - Pantalla de gestió d'articles

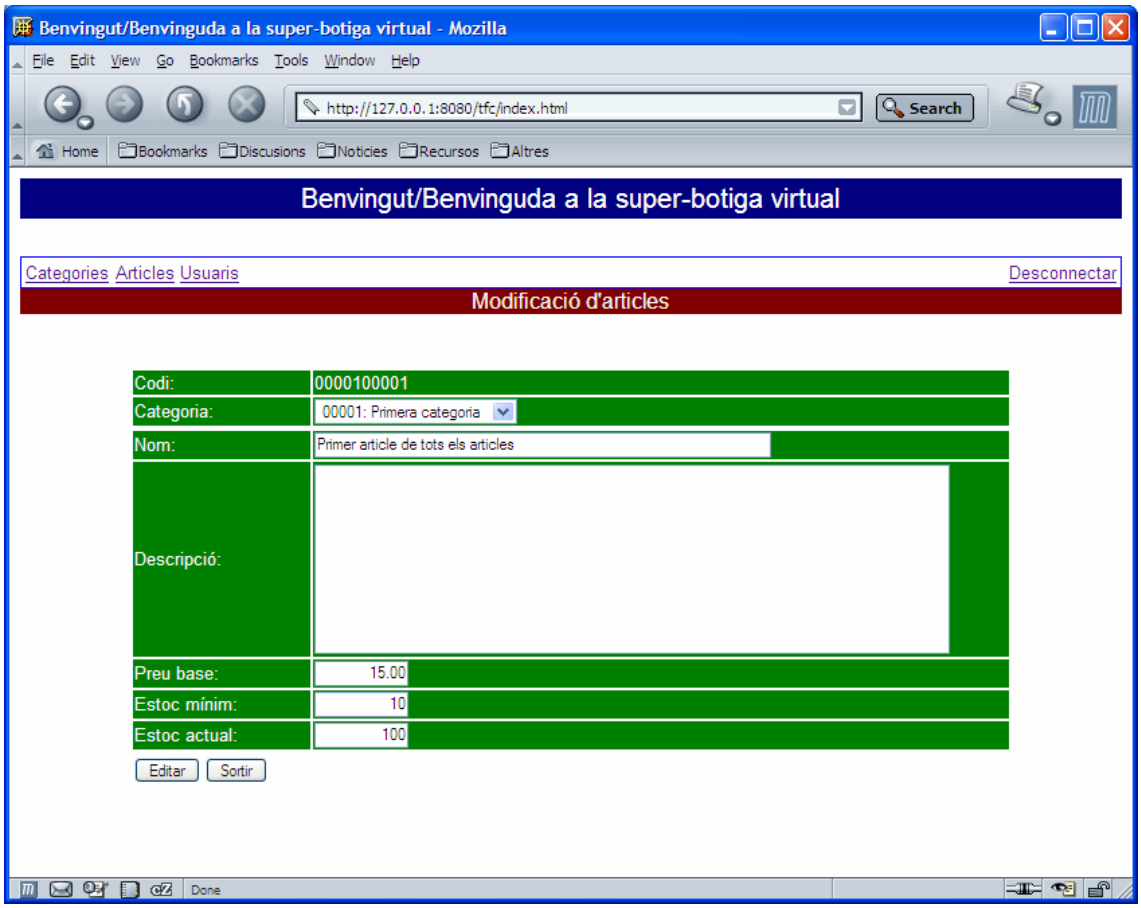

Figura 14 - Pantalla de modificació d'articles.

4. Implementació.

En aquest capítol s'exposarà l'estructura de l'aplicació tal com s'ha implementat: estructura de paquets, interrelacions entre classes, etc.

4.1. Estructura general.

Per enunciat, aquesta aplicació es basa en l'especificació J2EE 1.3. Això implica –recapitulant una mica- que la interfície d'usuari es basa en servlets i pàgines JSP, mentre que el backend *pot* basar-se en EJBs. Tal com s'ha comentat en els apartats anteriors, s'usarà EJBs per resoldre la persistència d ela major part d'entitats del sistema, mentre que la interfície d'usuari estarà basada en pàgines JSP gestionades per la infraestructura Struts (que no deixa de ser un "servlet universal" que incorpora el patró de disseny MVC).

Per facilitar la codificació de les regles de construcció Ant, i per separar lògicament els diferents tipus de components, s'ha adoptat per una estructura en paquets que reflexa la classificació formal dels components (i no la seva divisió funcional). Tots els paquets pengen d'una arrel **edu.uoc.jguillaumes.tfc**, sota la qual hi ha la classificació per tipus. La relació de paquets i classes és la que segueix:

4.1.1. edu.uoc.jguillaumes.tfc.beans

Nom del paquet: edu.uoc.jguillaumes.tfc.beans		
Contingut: Classes auxiliars		
Tipus	***Nom***	***Funció***
Interface	ControladorLlista	Definició de les funcionalitats d'una llista paginable.
Classe	ControladorLlistaList	Implementació de la interfase ControladorLlista usant una col·lecció de tipus List
Classe	MailBean	Tramesa de missatges de correu electrònic usant javamail.

4.1.2. edu.uoc.jguillaumes.tfc.dao

Nom del paquet: edu.uoc.jguillaumes.tfc.dao		
Contingut: Implementacions de persistència usant el patró DAO		
Tipus	***Nom***	***Funció***
Classe	UsuariDAO	Controlador de persistència per la classe Usuari (usuaris i clients).

4.1.3. edu.uoc.jguillaumes.tfc.ejb

Nom del paquet: edu.uoc.jguillaumes.tfc.ejb		
Contingut: Classes d'implementació pels EJBs amb marques Xdoclet i classes generades per Xdoclet (no inclou les interfícies).		
Tipus	***Nom***	***Funció***
Classe	ArticleBean	Implementació de l'entitat Article en forma d'EJB d'entitat CMP. Aquesta classe conté la definició del EJB i les marques Xdoclet corresponent.
Classe	ArticleBeanCMP	Classe generada per Xdoclet a partir de ArticleBean.
Classe	CarretCompraBean	Implementació del carret de compra en forma d'EJB de sessió amb Estat. Aquesta classe conté els mètodes de negoci del EJB i les marques Xdoclet.
Classe	CarretCompraSession	Classe generada per Xdoclet a partir de CarretCompraBean

Classe	CategoriaBean	Implementació de l'entitat Categoria en forma d'EJB d'entitat CMP. Aquesta classe conté la definició del EJB i les marques Xdoclet.
Classe	CategoriaBeanCMP	Classe generada per Xdoclet a partir de CategoriaBean
Classe	ComandaBean	Implementació de l'entitat Comanda en forma d'EJB d'entitat CMP. Aquesta classe conté la definició del EJB i les marques Xdoclet.
Classe	ComandaCMP	Classe generada per Xdoclet a partir de ComandaBean.
Classe	FacanaComandaBean	Implementació de la façana de comandes en forma d'EJB de sessió sense estat. Aquesta classe conté els mètodes de negoci i les marques Xdoclet.
Classe	FacanaComandaSession	Classe generada per Xdoclet a partir de FacanaComandaBean.
Classe	FacanaInventariBean	Implementació de la façana d'inventari (categories i articles) en forma d'EJB de sessió sense estat. Aquesta classe conté els mètodes de negoci i les marques Xdoclet.
Classe	FacanaInventariSession	Classe generada per Xdoclet a partir de FacanaInventariBean.
Classe	FacanaUsuariBean	Implementació de la façana d'usuaris (clients i administradors) en forma d'EJB de sessió sense estat. Aquesta classe conté els mètodes de negoci i les marques Xdoclet.
Classe	FacanaUsuariSession	Classe generada per Xdoclet a partir de FacanaUsuariBean.
Classe	LiniaComandaBean	Implementació de l'entitat LiniaComanda en forma d'EJB d'entitat CMP. Aquesta classe conté la definició del EJB i les marques Xdoclet.
Classe	LiniaComandaCMP	Classe generada per Xdoclet a partir de LiniaComandaBean
Classe	Messages	Classe auxiliar, generada per l'assistent d'internacionalització de l'Eclipse. Conté el suport per obtenir el text del missatge de correu electrònic a enviar al client en altes.
Classe	NumeradorBean	Implementació en forma d'EJB d'entitat CMP d'un numerador per obtenir claus numèriques correlatives. En l'aplicació s'usa només per numerar comandes, però podria servir per qualsevol altra cosa. Conté la definició i les etiquetes Xdoclet corresponents.
Classe	NumeradorCMP	Classe generada per Xdoclet a partir de NumeradorBean.

4.1.4. edu.uoc.jguillaumes.tfc.excepcions

Nom del paquet: edu.uoc.jguillaumes.tfc.exceptions		
Contingut: Excepcions generades i gestionades per l'aplicació		
Tipus	***Nom***	***Funció***
Classe	ClauDuplicadaException	Alta no realitzada per clau duplicada.
Classe	EstocInsuficientException	Compra no realitzada per manca d'estocs.
Classe	NoAutoBloqueigException	Un administrador ha intentat donar-se de baixa

		a si mateix i no està permès.
Classe	NoEsborratException	No s'ha pogut esborrar una instància d'un objecte probablement perquè té objectes dependents (per exemple, una Categoria que té Articles).
Classe	NoHiHaMesPaginesException	S'ha intentat passar de la darrera pàgina o anar abans de la primera.
Classe	NoTrobatException	No s'ha trobat una instància que es buscava, o que és necessària per a realitzar l'operació en curs.
Classe	TfcException	Arrel de les excepcions generades per l'aplicació. Correspon a una excepció no classificada.
Classe	UsuariNoTrobatException	No s'ha trobat una instància d'usuari necessària per realitzar l'operació.
Classe	UsuariTeComandesException	No es pot donar de baixa un usuari que té comandes (subclasse de NoEsborratException).

4.1.5. edu.uoc.jguillaumes.tfc.forms

Nom del paquet: edu.uoc.jguillaumes.tfc.forms		
Contingut: ActionForms per Struts		
Tipus	***Nom***	***Funció***
Classe	ArticleBotiga	Classe auxiliar. Representa cada una de les línies de les pantalles de navegació per la botiga (un "article" tal com es veu a la "botiga").
Classe	ArticlesForm	ActionForm per a la gestió d'articles de l'inventari. Suporta la llista d'articles, els camps de filtre i les dades d'un article per modificar o afegir.
Classe	BotigaForm	ActionForm per a la pantalla de navegació. Conté una llista d'instàncies d'ArticleBotiga, així com els camps auxiliars necessaris per pintar la pantalla.
Classe	CarretForm	ActionForm que suporta el carret de la compra.
Classe	CategoriesForm	ActionForm per a la gestió de categories de l'inventari. Suporta la llista de categories i les dades d'una categoria per modificar o afegir.
Interface	GenericForm	Defineix constants bàsiques que són heretades per la resta de ActionForms. Bàsicament, són els noms de les tecles de submit que s'usaran per passar comandaments a les Action.
Classe	TancarCompraForm	ActionForm per a la confirmació final de la compra.
Classe	UsuariForm	ActionForm per a les dades d'un usuari a crear (per part de l'administrador, o autoregistre) o modificar.
Classe	UsuarisForm	ActionForm per a la pantalla de gestió d'usuaris de l'administrador. Conté una llista d'usuaris, així com els camps auxiliars necessaris.

4.1.6. edu.uoc.jguillaumes.tfc.interfaces.

Aquest paquet conté únicament les interfases bàsiques (*home*), locals i remotes dels EJB del paquet edu.uoc.jguillaumes.tfc.ejb, per la qual cosa no val la pena detallar-lo.

4.1.7. edu.uoc.jguillaumes.tfc.resources

Nom del paquet: edu.uoc.jguillaumes.tfc.resources		
Contingut: Recursos per internacionalització		
Tipus	***Nom***	***Funció***
Properties	tfc.properties	Missatges i literals per la infraestructura d'internacionalització de Struts en català.
Properties	tfc_es.properties	Traducció al castellà de tfc.properties.
Properties	correu.properties	Literals per missatges de correu electrònic en català.
Properties	correu_es.properties	Literals per missatges de correu electrònic en castellà.

4.1.8. edu.uoc.jguillaumes.tfc.servlet

Nom del paquet: edu.uoc.jguillaumes.tfc.servlet		
Contingut: Servlets i Action Beans per Struts		
Tipus	***Nom***	***Funció***
Classe	AdminUsuariAction	Alta i modificació d'usuaris per part de l'administrador.
Classe	BotigaAction	Navegació per la llista de productes de la botiga virtual.
Classe	CarretAction	Gestió del carret de la compra (modificació d'unitats a comprar i baixa de línies).
Classe	CheckoutAction	Finalització d'una compra: creació d'una comanda a partir d'un carret de compra.
Classe	DesconnectarAction	Logout de l'aplicació.
Classe	LlistaArticlesAction	Gestió d'articles: llista, filtre i baixa d'articles.
Classe	LlistaCategoriesAction	Gestió de categories: llista, alta, baixa i modificació.
Classe	LlistaUsuarisAction	Navegació per la llista d'usuaris: baixa i desactivació d'usuaris.
Classe	ModifArticleAction	Altes i modificacions d'articles.
Classe	NullServlet	Servlet "nul". Només s'usa per contenir les etiquetes Xdoclet necessàries per generar correctament el descriptor de desplegament.
Classe	RegistreAction	Alta per auto-registre d'un nou client, i modificació de les dades pròpies.

4.1.9. edu.uoc.jguillaumes.tfc.vobj

Nom del paquet: edu.uoc.jguillaumes.tfc.vobj		
Contingut: Objectes-valor		
Tipus	***Nom***	***Funció***
Classe	Article	Entitat Article.
Classe	Categoria	Entitat Categoria.
Classe	ItemCarretCompra	Entitat ItemCarretCompra.
Classe	Usuari	Entitat Usuari.

4.2. Action Forms i Action Beans.

Les classes derivades de ActionForm i d'Action formen part de la infraestructura Struts. Per recordar-ho breument, la comunicació entre el controlador (Struts) i la presentació (JSP) es fa mitjançant especialitzacions d'ActionForm, mentre que la comunicació entre el Model (dades i mètodes de negoci de l'aplicació) i el controlador es fa mitjançant especialitzacions de Action.

En la implementació realitzada, totes les ActionForm acaben coincidint en una de les següents tres estructures:

- Una classe que conté els atributs necessaris per construir un objecte (probablement usant una referència a l'objecte a construir), a més de les dades necessàries per gestionar el formulari (definició de botons, i identificadors de pantalla i de pàgina ocults).
- Una classe que conté una llista d'objectes a presentar, i un vector de booleans per capturar una selecció per part de l'usuari.
- Una classe que conté les dues coses.

A més, tal com s'ha mencionat, totes les classes ActionForm tenen com a part comuna:

- Un identificador de pantalla, de nom idPantalla, que és usat per diferenciar els diferents usos que es pot fer d'un mateix formulari (per exemple, per donar d'alta un element o per modificar un element existent).
- Els literals corresponents al noms i valors dels botons de submissió del formulari. Aquests literals tenen un accessor de lectura. La raó de l'existència d'aquests literals és evitar haver de codificar-los "a mà" en les pàgines JSP i en les classes Action. El tenir-los definits dins de l'ActionForm permet usar el nom simbòlic, assegurant així que els valors especificats a la pàgina JSP i els valors verificats a la Action coincideixin.

Les classes Action tenen també una estructura similar:

- En primer lloc es determina quin és el botó de submissió que s'ha premut.
- Un cop determinat el botó, es selecciona l'acció a efectuar en funció de l'identificador de pantalla (idPantalla) que porti el ActionForm d'entrada.
- Es du a terme l'acció seleccionada.
- Si arribem al final de l'Action (és a dir, si no hem fet un *return* bifurcant cap una altra Action o pantalla JSP) es comprova si hi ha algun error a la llista d'errors. Si no n'hi ha, s'actualitzen les dades que falti a l'ActionForm i es respon *success*. En cas contrari, es respon *error.*

És interessant comentar el funcionament de les llistes paginables. En aquest cas, l'ActionForm conté un Vector que s'ha de carregar amb els elements de la pàgina en curs. La llista completa d'elements a paginar es penja d'un ControladorLlistaList, el qual és així mateix penjat de la sessió web. D'aquesta manera podem respondre a les peticions de paginar endavant i endarrera usant el ControladorLlista, i a l'hora la pàgina JSP pot saber a quina pantalla està preguntant-li directament al mateix ControladorLlista.

Per a veure els detalls corresponents a cada una de les classes, consulteu els javadocs que s'han adjuntat a aquest treball.

4.3. EJBs

En aquest desenvolupament s'han usat tres tipus d'EJB. Beans d'Entitat amb persistència gestionada per contenidor (EJB-CMP), Beans de sessió sense estat, i beans de sessió amb estat.

4.3.1. Entity EJB.

Com ha s'ha explicat al parlar del model de persistència, s'ha usat EJB-CMP per a totes les entitats del sistema amb l'excepció dels Usuaris (clients i administradors). A més s'ha implementat un EJB per numerar les comandes (encara que la implementació que s'ha fet serveix com a numerador genèric).

Cal insistir en que els EJB-CMP són únicament **gestors de persistència**, i que no incorporen pràcticament gens de lògica de negoci. La lògica de negoci resideix a les façanes de sessió.

Tot i que es va estudiar, no s'ha usat CMR (relacions gestionades pel contenidor). Tot i que sobre el paper l'ús de CMR pot resultar atractiu, en la pràctica s'ha vist que fins i tot complica la lògica de l'aplicació. Per exemple, en el que respecta a la integritat referencial, el model CMR només suporta actualitzacions i esborrats en cascada: si esborrem una categoria que té articles, el sistema esborra realment la categoria i, o bé esborra els articles relacionats (cosa que no ens convé en absolut), o bé deixa a *null* la categoria corresponent als articles relacionats. No llença una excepció per violació de

la regla de I.R. Això, i altres coses (el *costum* de JBoss de carregar tots els objectes relacionats encara que no es facin servir) ha fet decidir no usar CMR i implementar les relacions "a ma". Per una altra banda, la capa de presentació usa de forma pràcticament exclusiva les representacions en forma d'objectes-valor de les Entitats. Aquest és un motiu més per no usar CMR i gestionar les relacions directament a les façanes de sessió.

Finalment, com que els EJB d'entitat s'han d'usar únicament a través de les façanes corresponents, només se n'ha definit les interfases locals. Els accessos remots (si n'hi hagués) s'haurien de fer forçosament a través de façanes, i intercanviant objectes-valor en lloc de referències remotes.

4.3.2. Stateless Session EJB.

Els EJB de sessió sense Estat s'han usat per implementar les tres façanes de sessió que proporcionen accés a les dades de l'inventari (categories i articles), usuaris de la botiga (clients i administradors, a través d'un DAO) i les comandes (comandes i línies de comanda).

Les façanes permeten a la capa web (o la capa client) accedir les entitats persistents mitjançant objectes-valor:

- FacanaInventari permet accedir a Categoria i Article.
- FacanaUsuaris permet accedir a Usuari.
- FacanaComanda permet accedir a Comanda i LiniaComanda.

FacanaInventari i FacanaUsuaris són dues façanes prou riques com per cobrir totes les necessitats respecte a la gestió de les entitats que "amaguen". Tanmateix, FacanaComanda és molt bàsica, ja que només s'han implementat les funcions bàsiques del tractament de comandes (crear una comanda nova i llistar les comandes existents).

Els detalls complets sobre aquestes classes es poden trobar als javadocs corresponents.

4.3.3. Stateful Session EJB: carret de la compra.

L'aplicació usa un únic EJB de sessió amb estat, que serveix per gestionar el carret de compra.

Aquest EJB conté únicament una col·lecció d'ítems de compra, que s'implementa en forma de TreeMap. La clau del TreeMap és el codi d'article, de forma que no hi pot haver dos ítems amb el mateix codi d'article dins del carret de la compra. Si es vol afegir més unitats d'un article, això es fa incrementant el nombre d'unitats de l'ítem corresponent.

El EJB CarretCompra té totes les funcions per gestionar la seva llista d'ítems i per obtenir informació global (nombre d'ítems, nombre d'unitats i preu total). Tanmateix, s'ha preferit minimitzar l'acoblament amb altres classes. D'aquesta manera, el Carret de Compra només té coneixement directe de les classes Usuari, ItemCarroCompra i la façana d'inventari (a través de la seva interfície local). Les dues primeres són Objectes-Valor, i la darrera és una façana per accedir l'inventari, funcionalitat indispensable per a la correcta gestió del carret de la compra.

Val a dir que les verificacions relatives a les existències es fan en dos moments diferents:

- Quan es crea el ItemCarroCompra, o quan se'n modifiquen les unitats a comprar. En aquest moment es fa una verificació d'existències, però aquesta verificació no assegura que en el moment de confirmar la compra hi hagi estoc suficient (pot haver-lo agafat un altre usuari). Es va pensar en mantenir uns comptadors d'"unitats en vol", però es va descartar per la seva complexitat i per que obligaria a donar persistència al carro de compra. A més, aquesta verificació es fa contra l'estoc reflectit en un Objecte-Valor Article, que pot no coincidir amb l'existent a la base de dades, ja que la còpia no està acoblada a les dades persistents.
- Quan es genera la comanda. Aquest cop la verificació es fa contra existències reals a base de dades. Com que la disminució de l'estoc disponible es fa en la mateixa transacció, s'assegura la integritat de l'operació (si s'ha trobat estoc suficient, es farà la compra i l'estoc quedarà actualitzat).

La primera verificació la fa la instància de ItemCarroCompra que s'està creant o modificant, mentre que la segona la fa la façana de compres.

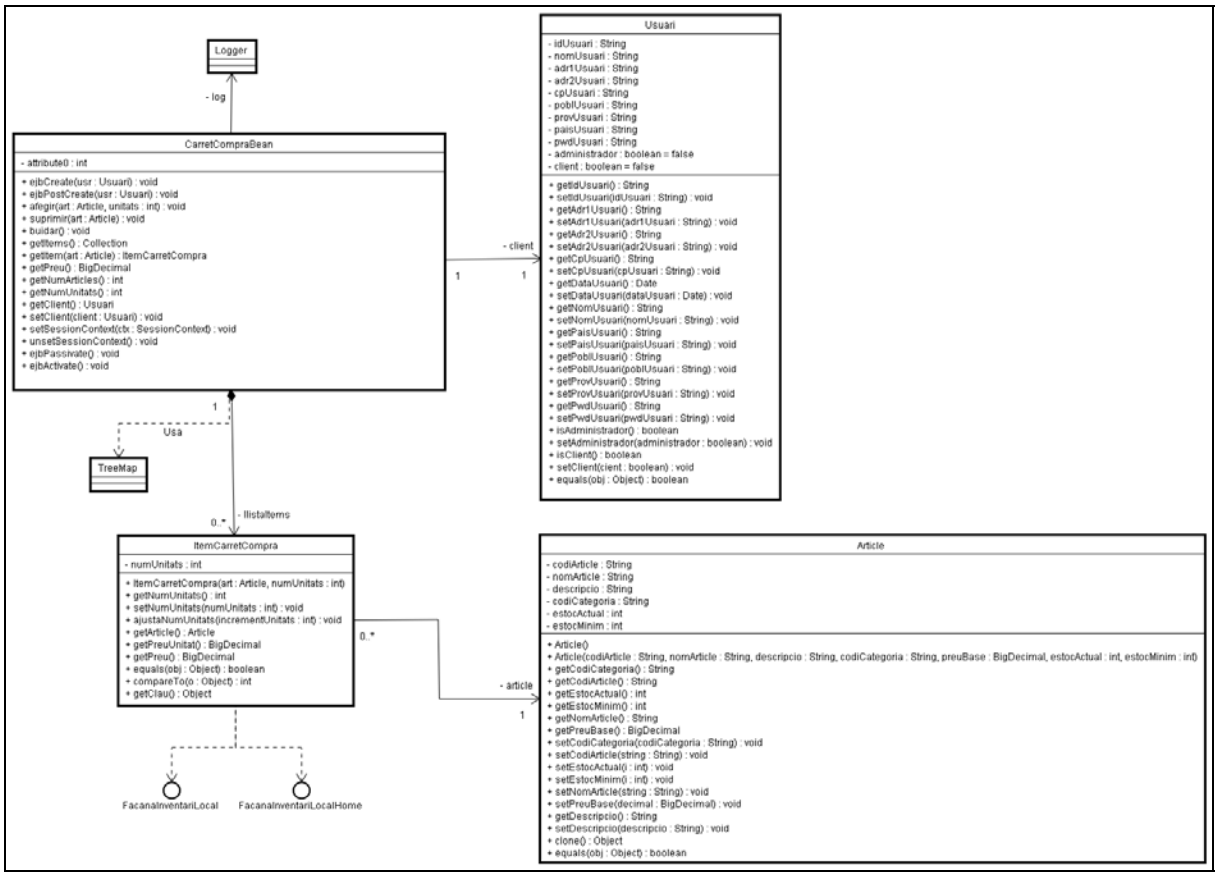

Figura 15 - Carret de Compra i classes relacionades

4.4. Objectes-Valor.

En poques paraules, el patró dels objectes-valor, ara anomenat *objectes de transferència*, consisteix en passar la informació corresponent als atributs dels objectes persistents entre les diferents capes de l'aplicació en forma d'instàncies d'objectes creats especialment a aquest efecte (els *objectes-valor*), en lloc d'usar referències remotes i accedir als atributs un a un. Això permet estalviar tràfic de xarxa i separar més clarament la capa de persistència de les capes de lògica de negoci i de presentació.

Els objectes-valor que s'han definit en l'aplicació són els següents:

- **Article**. És la representació d'una instància de l'EJB ArticleBean.
- **Categoria**. És la representació d'una instància de l'EJB CategoriaBean.
- **Comanda**. És la representació d'una instància de l'EJB Comanda.
- **ItemCarretCompra**. Representa una línia del carret de la compra. Aquesta classe no es correspon a un objecte persistent (els ItemCarretCompra pengen d'una instància de l'EJB de sessió CarretCompra, que és un objecte volàtil lligat a una sessió web). Estrictament, no és, doncs, un objecte-valor, però per similitud amb la resta de classes que representen objectes, s'ha agrupat a aquest apartat i s'ha definit al mateix paquet dels Objectes-Valor.
- **LiniaComanda**. És la representació d'una instància de l'EJB LiniaComanda.
- **Usuari**. Aquesta classe agrupa la informació d'usuaris i rols, tal com està definida al diagrama de classes d'entitats del sistema. Les instàncies d'aquesta classe estan suportades per un DAO, que gestiona la persistència i implementa les operacions necessàries per crear, buscar, modificar i eliminar usuaris i privilegis.

A continuació es pot trobar un diagrama de classes amb les relacions (lògiques) entre els objectes valor "autèntics" definits (no hi ha ItemCarretCompra, pel motiu exposat més amunt).

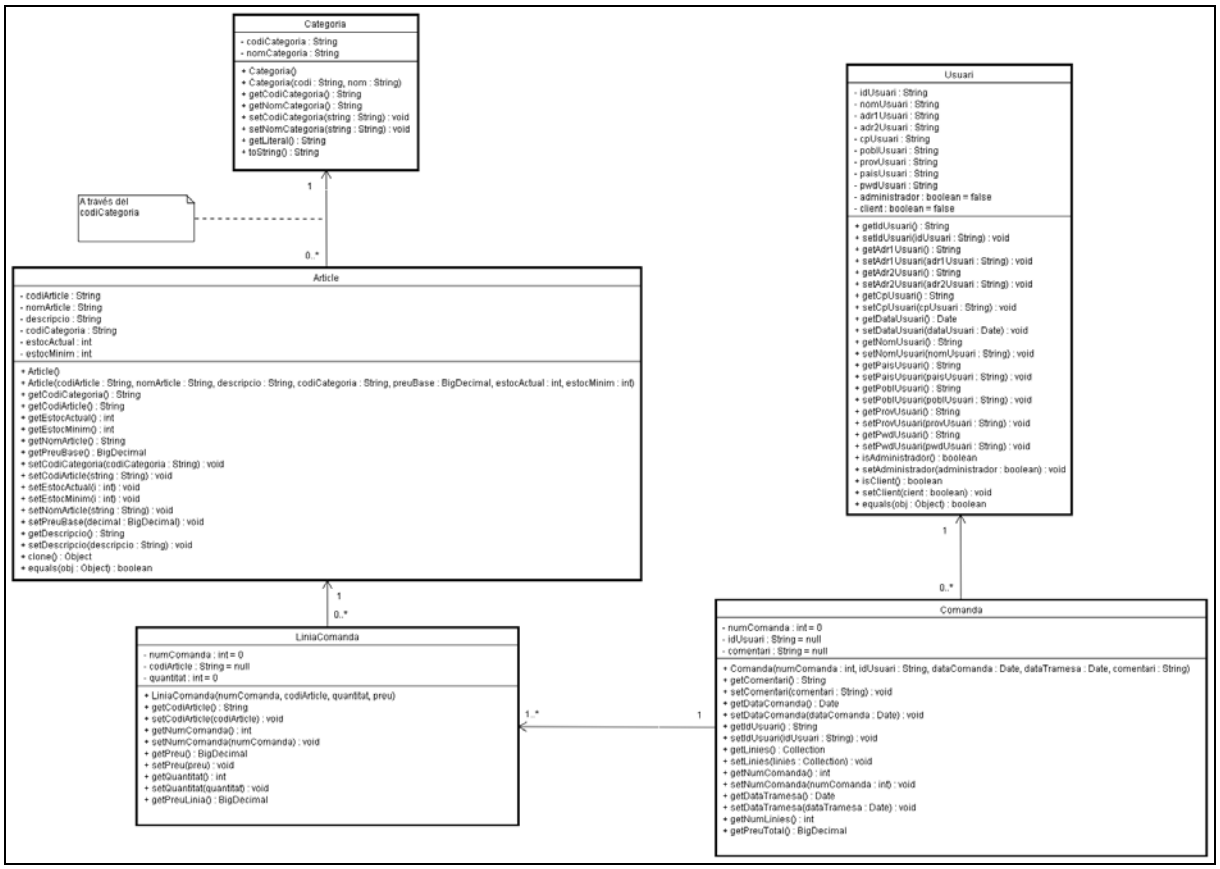

Figura 16 - Relacions entre els Objectes-Valor

4.5. Altres classes de suport.

4.5.1. ControladorLlista i ControladorLlistaList.

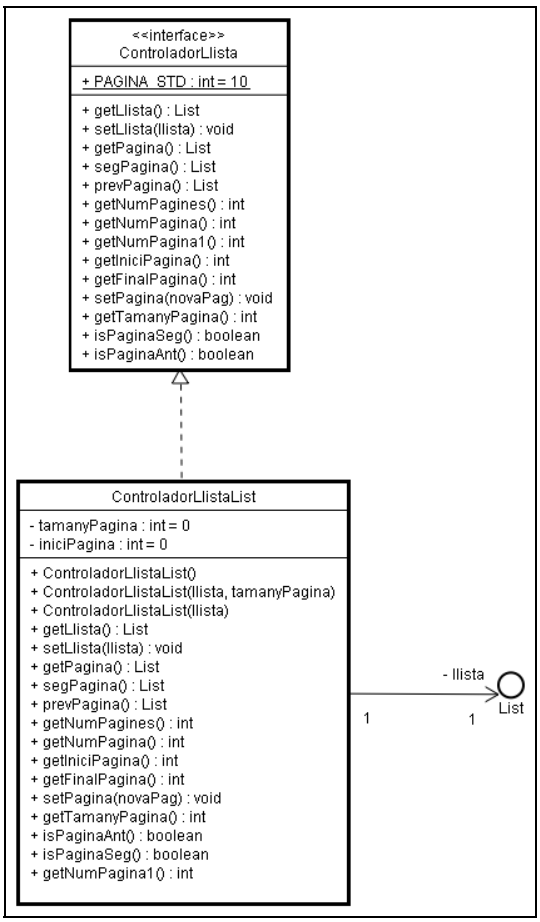

Figura 17 - ControladorLlista i ControladorLlistaList

La paginació de llistes de valors ("pantalles" de llistes de dades) s'ha resolt definint una interfície especialitzada en gestionar dades per blocs. Aquesta interfície es **ControladorLlista**, i defineix una sèrie d'operacions genèriques necessàries per paginar dades. A grans trets, la idea és mantenir una llista a través de la qual es pot accedir a tots els valors a paginar, i disposar d'operacions que ens permetin obtenir dades sobre el paginador (nombre total de línies i pàgines, moure la pàgina actual (avançant o retrocedint un nombre prefixat de línies) i obtenir la pàgina actual en forma de col·lecció.

Per desenvolupar el projecte s'ha definit una implementació d'aquesta interfície basada en una llista de dades en memòria. Aquesta solució no és ni de bon tros escalable, ja que el ControladorLlista s'ha de desar a la sessió web, i si la llista és molt gran i hi ha moltes sessions concurrents pot produir-se un gran consum de memòria *heap* de la màquina virtual on corre el servidor web.

Si es volés fer un desenvolupament escalable, es podrien prendre les mesures següents:

- Definir una implementació de ControladorLlista basada en suport en base de dades. Això suposaria haver de fer accessos JDBC des de la capa web, cosa que viola el principi de la separació de capes.
- Posar a la llista únicament les claus dels objectes a presentar, i recuperar-los des de la capa web en el moment en que calgui mostrar-los. Això provocaria un augment del tràfic de xarxa entre la capa web i la capa EJB, o multitud d'accessos a la capa EJB mitjançant les interfases locals.
- Implementar el ControladorLlista en forma d'EJB de sessió amb estat, de forma que la despesa de memòria es faria a la capa EJB (i es suposa que el contenidor passivitzaria les instàncies necessàries abans de que això fos un problema). En aquest supòsit, la sessió web només tindria una referència a una interfase local o remota al EJB corresponent.

Òbviament es podria optar per combinar més d'una estratègia. Per exemple, es podria basar el ControladorLlista en un EJB de sessió amb Estat que guardés claus i no objectes. Caldria estudiar amb cura els guanys i beneficis de cada solució.

4.5.2. MailBean.

La classe MailBean té per únic objecte la tramesa de missatges de correu electrònic usant els serveis javamail.

El seu ús és molt senzill. La classe té un únic constructor i un únic mètode. El constructor rep com a paràmetres una adreça de correu origen, que serà el contingut del cap From: del missatge generat, i una adreça del servidor SMTP a utilitzar. El mètode enviarMissatge rep tres paràmetres: l'adreça del destinatari, el Subject: del correu i el contingut del missatge en forma de String.

L'aplicació usa referències d'entorn JNDI per obtenir l'adreça origen i el servidor SMTP. Concretament, les variables buscades són `tfc/correu-from` i `tfc/correu-smtp`. Cal tenir present que al buscar aquestes referències s'ha d'usar el prefix `java:comp/env`.

4.6. Altres detalls d'implementació.

4.6.1. Logging.

Per fer el logging de seguiment i depuració s'ha usat log4j. Aquesta infraestructura permet inserir lliurement instruccions de logging en el codi, que després s'activen o no en funció del *nivell de logging* actiu, definit a un arxiu de configuració del JBoss, i del nivell de logging de cada instrucció.

Els noms dels *Loggers* usats són en tots els casos el nom complet qualificat de cadascuna de les classes que usa log4j. Això permetria, si fos necessari, activar i desactivar el logging a nivell de paquet o a nivell de classe específica.

L'ús dels *Loggers* no és homogeni en totes les classes. En algunes d'elles s'usen instàncies dels *Loggers* específiques a nivell de mètode (el *Logger* és una variable local), mentre que en altres casos es construeix un *Logger* a nivell d'instància en el constructor de la classe o en el mètode de creació (cas dels EJBs). Això no té més importància, i si hi hagués temps suficient seria molt senzill d'estandarditzar.

4.6.2. Internacionalització.

S'ha intentat que aquesta aplicació sigui fàcilment internacionalitzable. Tal com es lliura, suporta els idiomes català i castellà. Els recursos que s'han usat per facilitar la internacionalització són:

- El suport d'internacionalització que incorpora la infraestructura Struts.
- El mecanisme de ResourceBoundle que implementa el propi java.

L'aplicació usa dos *ResourceBundle*, situats dins del paquet edu.uoc.jguillaumes.tfc.resources:

- Tfc.properties / Tfc_es.properties, que conté els literals de l'aplicació que s'han de mostrar per pantalla (tant els basats en Struts com els textos de les Exceptions[3], etc…).
- correu.properties i correu_es.properties, que conté els literals dels missatges que s'han d'enviar per correu electrònic.

En tots els casos l'idioma dels missatges s'escull en funció del que l'usuari tingui definit com a preferit al seu navegador. L'idioma per defecte és el català.

El suport d'internacionalització de Struts té un problema empipador. El validador de Struts genera codi javascript per fer validacions en el propi navegador del client. El problema és que si els missatges d'error han de ser visibles **com a text de la pàgina** s'han d'usar entitats HTML al escriure'ls (&agrave, &ccedil i tota aquesta sèrie de noms per caràcters internacionals). Tanmateix, el validador usa el text *tal com està* per generar el codi javascript. La conseqüència és que en les finestres d'error javascript apareixen les entitats sense traduir. Òbviament, això no és precisament molt *user-friendly*, per la qual cosa hi ha tres alternatives:

- Arreglar el validador de Struts.
- No usar javascript ni validació local.
- No usar caràcters internacionals en els missatges que pugui generar el validador.

Descartada la primera opció per falta de temps, he optat per la tercera: mantenir el javascript però usar missatges ortogràficament incorrectes.

[3] Tal com s'explicarà en l'apartat corresponent, la internacionalització dels missatges de les excepcions no funciona, ja que es busca el missatge en l'idioma corresponent al *Locale* per defecte de la JVM, en lloc de l'idioma de la sessió del client (el *Locale* del *request*).

5. Desplegament sota JBoss i PostgreSQL.

Aquest apartat és una guia d'instal·lació de l'aplicació sota el servidor JBoss i el SGBD Postgresql.

5.1. Requisits per a la instal·lació.

Aquests són els nivells de programari sota els que s'ha provat l'aplicació. És possible que es pugui instal·lar i executar en versions anteriors, però no s'ha provat ni hi ha cap garantia de que sigui així.

- **Màquina virtual:** JVM de Sun Microsystems, versió 1.4.2.
- **Servidor JBoss**. Versió 3.2.3, amb Jakarta Tomcat integrat.
- **Servidor PostgreSQL**. Versió 7.3 ó 7.4. **Ull!** Cal usar la versió del driver JDBC corresponent a la versió de PostgreSQL que s'hagi d'usar. El driver de la versió 7.3 **no funciona correctament** amb la 7.4.
- **Navegadors**. S'ha provat extensament amb els navegadors Safari 1.1 i Mozilla 1.7. S'ha provat amb menys exhaustivitat amb Internet Explorer 6.1.
- **Sistema Operatiu**. Teòricament el S.O. no hauria de tenir importància al ser una aplicació 100% java. El desenvolupament s'ha fet sota MacOS X 10.3 i Windows XP Home. L'aplicació s'ha desplegat sota Linux (nucli 2.4.26, distribució Red Hat 8 molt parxejada).

5.2. Tasques prèvies al desplegament.

Abans de desplegar l'aplicació caldrà fer de forma manual les tasques següents:

- **Crear l'usuari PostgreSQL** que serà propietari de les dades:
  - `$ createuser –A –P tfc`

  Aquest comandament demanarà un password, que ha de ser precisament `tfc`. Si s'usa un altre password caldrà modificar l'arxiu de desplegament del DataSource, tal com, s'explica més endavant. El comandament s'haurà d'executar des d'un usuari que tingui privilegis d'administrador de PostgreSQL.
- **Crear la base de dades**.
  - `$ createdb –O tfc tfc`

  Novament, aquest comandament s'haurà de llençar des d'un usuari que tingui privilegis d'administrador de PostgreSQL.
- **Crear les taules i inicialitzar la base de dades.**
  - `$ psql –U tfc –f tfc-jguillaumes.sql`

  L'arxiu tfc-jguillaumes.sql forma part del paquet ZIP lliurat amb l'aplicació.

  Aquest comandament s'hauria d'executar també sota l'usuari administrador PostgreSQL. Si el comandament finalitza sense errors tindrem la base de dades definida, amb un administrador per defecte creat. L'administrador té l'identificador "admin" i la paraula-clau "password".
- **Inserir la definició del domini de seguretat a JBoss.** Els dominis de seguretat no es poden desplegar igual que les aplicacions o els datasources. Definir un domini de seguretat requereix modificar un arxiu de configuració XML i reiniciar el JBoss.

  L'arxiu de configuració es troba al subdirectori `conf` de l'arbre del servidor que estem usant (per exemple, `$JBOSS_HOME/server/default/conf`), i s'anomena `login-config.xml`. Aquest document XML està format per un element arrel <policy> del que pengen diversos elements <application-policy>, un per cada domini de seguretat. Per inserir la definició que usa el tfc només cal enganxar dins del document el fragment XML que es trobarà a l'arxiu `jguillaumes-tfc-login.xml`, que forma part del ZIP lliurat amb el projecte. Cal tenir cura d'inserir el fragment al lloc correcte (per exemple, després del darrer tag </application-policy>), respectant l'estructura del document XML.

5.3. Desplegament de l'aplicació.

El desplegament de l'aplicació consisteix en copiar al subdirectori deploy del servidor JBoss els elements necessaris: el driver JDBC, la definició del DataSource i l'arxiu EAR de l'aplicació pròpiament dita.

- El **driver JDBC** ha de correspondre a la versió exacta de PostgreSQL que s'estigui usant. En cas contrari ens trobarem amb errors estranys i aparentment absurds, i amb comportaments irregulars de l'aplicació.

- La **definició del datasource** es troba a l'arxiu `jguillaumes-tfc-ds.xml`.
- L'**arxiu EAR de l'aplicació** és `jguillaumes-tfc.ear`.

Un cop feta la còpia dels arxius al directori corresponent, cal esperar un parell de minuts a que JBoss desplegui els components abans de provar d'accedir-hi.

5.4. Iniciar l'aplicació.

Per iniciar l'aplicació cal dirigir el navegador a l'adreça `http://servidor-jboss:port-jboss:/tfc`. Suposant un navegador configurat per admetre l'idioma català com a preferit, s'haurà d'obtenir la pantalla següent:

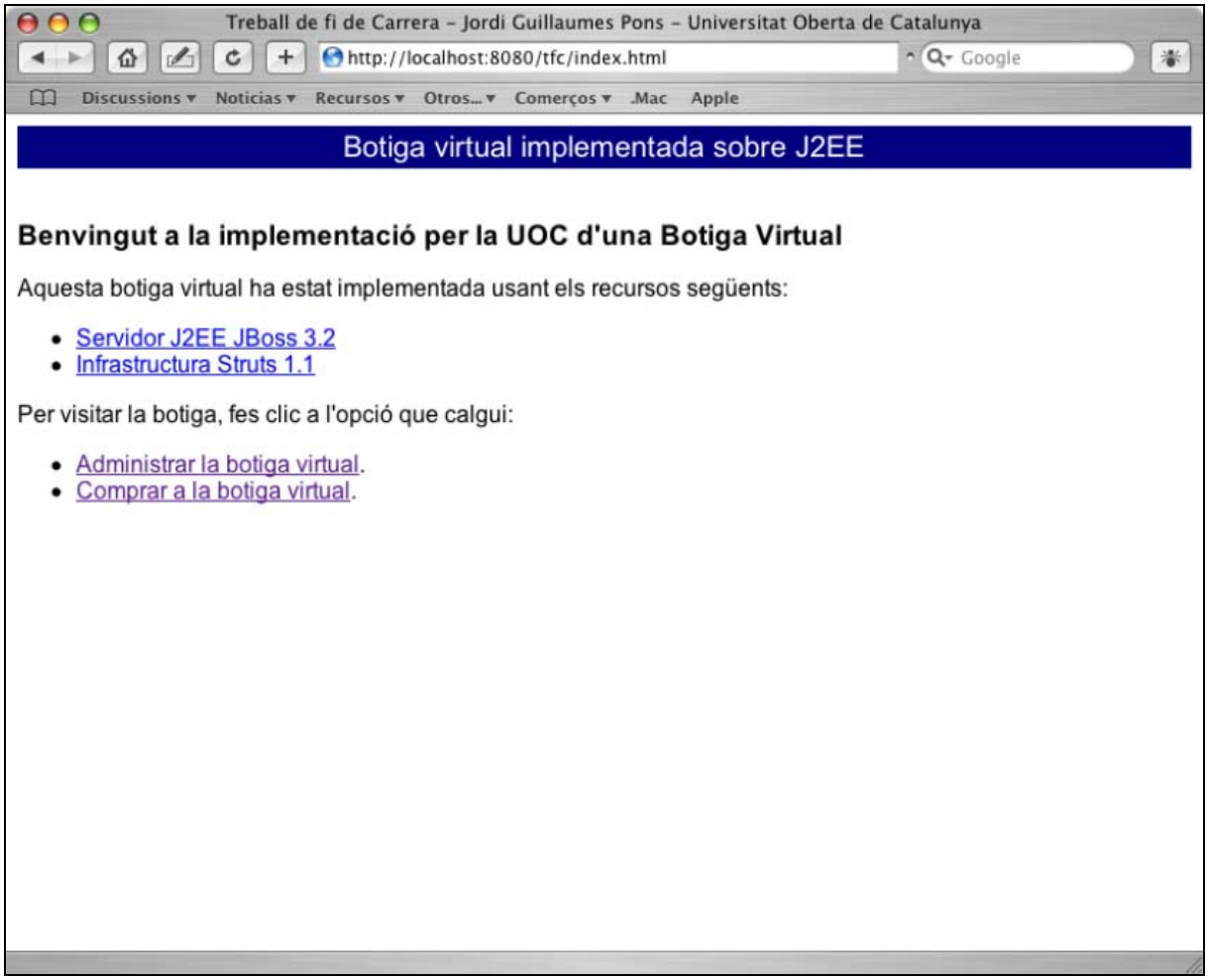

Figura 18 - Pantalla inicial de l'aplicació

Tot el text de la pantalla està generat usant els serveis d'internacionalització de Struts, per la qual cosa si es pot visualitzar aquesta pàgina podem tenir la seguretat de que el motor JSP i el suport Struts funcionen correctament.

Els primers passos després d'iniciar l'aplicació podrien ser:

- Entrar com a Administrador, amb identificador "admin" i paraula-clau "password", accedir l'administració d'usuaris i canviar la paraula-clau de l'administrador.
- Accedir l'administració de categories i definir-ne algunes.
- Accedir l'administració d'articles i definir-ne alguns, tenint cura de deixar l'estoc diferent de zero.
- Entrar com a client ("Comprar a la botiga virtual") i registrar un nou usuari.
- Entrar com a client amb l'usuari tot just registrat, fullejar l'inventari i fer alguna compra.

5.5. Configuració de la tramesa de correu electrònic.

L'aplicació usa el correu electrònic per notificar l'usuari després de fer l'alta. Per generar i enviar els missatges de correu l'aplicació usa la classe MailBean, comentada al capítol anterior. Aquesta classe necessita saber el nom de l'originador del correu (from) i del servidor SMTP a usar per enviar-lo.

Aquestes dades s'obtenen a partir d'entrades d'entorn JNDI (*environment references*), i es declaren al descriptor de desplegament dels EJB que en fan ús (en aquest cas, FacanaUsuaris). Si es seguís el cicle de desenvolupament J2EE proposat per Sun Microsystems la definició dels valors d'aquestes entrades d'entorn correspondria al *packager* i en última instància al *deployer* de l'aplicació, que haurien de disposar de les eines adients per donar el valor correcte a aquestes entrades.

Malauradament, l'entorn JBoss no disposa de cap d'aquestes eines. La construcció dels descriptors de desplegament és manual o –com a molt- assistida. Xdoclet, en particular, construeix els descriptors a partir de comentaris tipus javadoc inserits en el codi font de l'aplicació. Si bé això és molt còmode per desenvolupar, té e problema de lligar dos rols que haurien d'estat separats, i de fer dependre una qüestió a establir en el moment del desplegament d'un codi font creat per un programador en temps de desenvolupament.

En la versió de l'aplicació que es lliura, la definició d'aquestes entrades d'entorn és la següent:

java:comp/env/tfc/correu-from	administrador@botiga-virtual.com
java:comp/env/tfc/correu-smtp	127.0.0.1

Això significa que perquè funcioni la tramesa de correu hi ha d'haver un servidor SMTP engegat al host on es desplegui l'aplicació. En cas contrari no s'enviarà missatge de correu, i apareixerà una entrada de tipus WARNING a log del servidor.

Si es vol modificar aquests valors per defecte es pot optar per:

- Modificar els valors als comentaris Xdoclet a la classe FacanaUsuari, i tornar a fer els processos de generació xdoclet i empaquetat de l'aplicació. Si es vol optar per aquesta via, el més recomanable és importar el paquet font, jguillaumes-tfc-font.jar, dins d'un espai de treball d'Eclipse i usar JBosside per fer la generació.
- Modificar els valors directament al descriptor de desplegament, ejb-jar.xml. Per fer això cal:
  - Descomprimir el EAR d'aplicació per extraure el paquet d'EJBs, Tfc-EJB.jar.
  - Descomprimir el JAR d'EJBs, Tfc-EJB.jar.
  - Editar el descriptor de desplegament, ejb-jar.xml, al subdirectori META-INF del JAR descomprimit, canviant el valor de les entrades d'entorn que es troben en forma d'elements <env-entry> dins de la descripció del EJB FacanaUsuari.
  - Tornar a construir els paquets Tfc-EJB.jar i jguillaumes-tfc.ear, cuidant que tot quedi al mateix lloc.
- Finalment, a mi no em costa gens modificar aquests valors i tornar a generar l'aplicació. Només cal que se'm demani.

6. Conclusions.

És el moment de fer balanç sobre el que ha estat el desenvolupament d'aquest projecte. Malauradament, les limitacions de temps (tres mesos i mig) i recursos (una sola persona, que a més havia d'aprendre l'ús de la tecnologia a l'hora que feia el desenvolupament) han impossibilitat que s'assolissin tots els objectius plantejats a l'anàlisi i al disseny de l'aplicació. El temps que ha faltat ha estat poc: probablement amb dues setmanes més s'hagués aconseguit implementar i depurar totes les funcionalitats que s'havien marcat com a objectiu.

6.1. Objectius assolits.

En primer lloc, i tot i que no és un objectiu quantificable, es pot dir que s'han assolit els objectius respecte a l'aprenentatge i familiarització amb la tecnologia J2EE. Tot i que la corba d'aprenentatge comença tenint una pendent considerable, arriba un moment a partir del qual la productivitat en quant al desenvolupament es dispara, fins al punt en el que es pot codificar i preparar una funció de negoci sencera en un parell de dies de treball. A més, amb els coneixements adquirits també es podria accelerar substancialment el treball de disseny.

En el que respecta a les funcionalitats desenvolupades, podem considerar com a aconseguits els següents objectius, amb les salvedats que s'indica en cada cas:

- **Gestió d'usuaris** per part de l'administrador.
- **Auto-registre i manteniment de dades pròpies** per part dels clients.
- **Connexió i desconnexió** al sistema.
- **Gestió de categories** per part de l'administrador.
- **Gestió d'articles** per part de l'administrador, amb l'excepció del filtratge per nom parcial d'article, que no s'ha arribat a implementar: el camp corresponent a les pantalles està desactivat.
- **Navegació per la botiga** per part del client, amb dues salvedats:
  - No s'ha implementat el filtratge per nom parcial d'article.
  - No s'ha implementat una consulta de detall dels articles. El codi d'article hauria de ser clicable, i hauria de mostrar el detall de les característiques de l'article en una nova finestra.
- **Gestió del carret de la compra** per part del client, amb la salvedat esmentada de l'absència d'una consulta de detall d'article.
- **Finalització de la compra i gravació de comanda**, amb la salvedat de que no es notifica per correu electrònic el client sobre la comanda efectuada.

6.2. Objectius no assolits.

La relació d'objectius inicials que no s'han pogut assolir per falta de temps són:

- **Gestió de comandes**. El que s'ha implementat és molt bàsic. Per poder parlar d'una gestió de comandes amb "cara i ulls" hauria de tenir (i no té):
  - Selecció de comandes per usuari.
  - Selecció de comandes per estat (facturada / no facturada).
  - Detall de comanda (ara mateix només es presenta un llistat en el que, tot i ser clicable, el número de comanda no porta enlloc).
  - Canvi d'estat d'una comanda (marcar-la o desmarcar-la com a facturada).

  Aquestes mancances justifiquen el considerar com "no feta" la gestió de comandes.
- **Llistat d'incidències d'estoc**. No s'ha implementat, tot i que part de la infraestructura està al seu lloc: existeix un mètode buscador d'articles amb incidència al EJB ArticleBean, que té la seva corresponent rèplica a la façana FacanaInventari.
- **Llistat de comandes del dia**. No s'ha implementat, tot i que la consulta de comandes pot filtrar per dates i, per tant, dóna part d'aquesta funcionalitat.
- **Consulta de comandes per part de l'usuari**. No s'ha implementat aquesta funcionalitat.

Altres possibles funcionalitats (com ara la gestió de trameses, despeses associades, etc...) ja havien estat explícitament excloses al plantejament inicial.

6.3. Errors i incidències al programari.

Tot i que –òbviament- s'ha intentat depurar en el possible el programari que es lliura, l'excusa habitual (falta de temps) fa que hi hagi algunes parts en les que és probable que es pugui trobar errors. En general, tot el programari s'ha provat exhaustivament. Tot i així, hi ha els següents errors detectats i no corregits a la versió final:

- El formulari del *carret de la compra* permet entrar dades no numèriques com a quantitat d'unitats a comprar. El programari pren com a *zero* qualsevol quantitat no numèrica que s'entri.
- La navegació entre el carret de la compra i la confirmació de la mateixa presenta problemes. Aquests problemes poden portar a que apareguin a pantalla excepcions java. Si es torna a entrar a la botiga, però, el carret de la compra es manté i la situació es normalitza.
- Els textos d'error lligats a Excepcions apareixen sempre en l'idioma corresponent al Locale per defecte de la màquina virtual a la que corre JBoss. Això és així perquè s'usa getLocalizedMessage() per obtenir els textos. Aquest mètode usa el Locale per defecte, mentre que s'hauria d'usar el Locale corresponent a la petició (request) http. La solució seria definir un mètode getLocalizedMessage(Locale loc) que admetés un Locale explícit. Amb això es solucionarien els textos corresponents a Exceptions definides per l'aplicació (descendents de TfcException).

6.4. Hipotètica evolució futura de l'aplicació.

Anotarem a continuació quina seria l'evolució de l'aplicació si no hi hagués una data d'entrega fixada.

6.4.1. Implementació de funcionalitats que falten.

- La implementació de la pantalla de **detall d'article** seria força trivial. Caldria convertir en enllaços els codis d'article, fent-los referir a una Action a la que es passaria com a paràmetre el codi de l'article a mostrar. La Action en qüestió invocaria un JSP que mostraria els detalls de l'article i tindria un simple botó de "tancar". A ser possible aquesta pàgina JSP contindria codi javascript per dimensionar-se correctament.
- La implementació de la pantalla de **detall de comanda** seria força equivalent, amb l'afegit de que hauria de preveure la modificació de l'estat de la comanda. Aquesta modificació es faria modificant l'atribut dataTramesa de la comanda. Es suposa que si dataTramesa és nul·la la comanda encara no ha estat facturada.
- L'**afegir filtres per usuari i per estat a la pantalla de llista de comandes** seria força senzill: només caldria afegir mètodes *finder* a l'EJB ComandaBean i reflectir-los en FacanaComanda. Un cop fet això s'haurien d'afegir els camps de filtre a l'ActionForm corresponent i a la pàgina JSP, i tractar-los adequadament a la Action respectiva.
- La **consulta de comandes per part de l'usuari** també seria força trivial, ja que es podria basar en els filtres creats per la consulta de l'administrador. Òbviament, la pantalla de consulta de detall de comanda per usuari no hauria de permetre cap modificació.
- El **llistat d'articles amb incidència d'estoc** s'hauria de basar en el *finder* ja existent. La implementació podria fer-se de dues maneres:
  - En forma d'aplicació client *stand-alone***,** que no correria dins del servidor J2EE sinó en la seva pròpia JVM. Aquesta aplicació invocaria la FacanaComanda usant la interfase remota corresponent, i hauria de ser planificada mitjançant una eina d'execució periòdica tipus *cron*.
  - En forma de Message-Driven EJB, que es dispararia per una acció de l'Administrador que enviaria un missatge a la cua corresponent. L'EJB construiria el llistat i el trametria per correu electrònic a l'administrador que hagués cursat la petició.
  - Combinació de les dues formes: un client *stand-alone* que només enviaria un missatge que seria processat pel MDB. Aquesta opció permetria l'ús d'una eina tipus *cron* per planificar el treball, mantenint força senzill el client i minimitzant el tràfic de xarxa.
- Per resoldre el **llistat de comandes del dia** es podria usar tècniques equivalents a les del llistat d'incidències.

6.4.2. Correcció de problemes i millores generals.

- Després de l'auto-registre l'usuari ha de fer logon. Hagués estat millor fer el logon automàticament a partir de les dades tot just entrades. Per fer això caldria crear un gestor de logins que proporcionés el Principal ja validat.

- Hagués estat molt positiu guardar imatges dels articles. Això s'hauria de fer amb un camp de tipus LOB a la taula corresponent, i un servlet encarregat de servir les imatges als clients. La pantalla de llista d'articles contindria un enllaç que activaria aquest servlet i mostraria una imatge en una finestra de navegador nova.
- El missatge de correu de confirmació d'alta és molt pobre (text senzill). S'hagués hagut d'usar un missatge HTML encapsulat dins d'un correu MIME. Això és relativament senzill de fer amb la interfase javamail.
- En termes generals, caldria reconsiderar l'opció adoptada d'usar l'adreça de correu com a identificador d'usuari. Si bé te avantatges evidents, presenta el problema de que els usuaris poden canviar d'adreça, i tal com s'ha muntat un usuari que canviés d'adreça hauria de donar-se d'alta de nou. Això no seria acceptable en un entorn de producció real.

7. Bibliografia i referències.

- UML y Patrones (2ª Edición) – Craig Larman – Pearson/Prentice Hall, 2003 – ISBN: 0-13-092569.1
- JDBC API Tutorial and Reference, Second Edition – White/Fisher/Cattell/Hamilton/Hapner – Addison-Wesley, 1999 – ISBN: 0-201-43328-1
- Programación Java Server con J2EE Edición 1.3 – Allamajaru/Beust/Davies/Jewell/Johnson/Longshaw/Nagappan/Sarang/Toussaint/Tyagi/Watson/Wilcox/Williamson/O'Connor – Wrox/Anaya Multimedia, 2002 – ISBN: 84-415-1358-9
- Mastering Enterprise JavaBeans, Second Edition – Roman/Ambler/Jewell – Wiley Computer Publishing, 2002 – ISBN: 0-471-41711-4 (eBook)
- Jakarta Struts Live – Rick Hightower – SourceBeat LLC, 2004 – ISBN: 097-48843-08
- JBoss administration and development Third Edition – Scott Stark & The JBoss Group – JBoss Group LLC, 2003 – (eBook).
- http://www.theserverside.com
- http://java.sun.com/blueprints/patterns/index.html (i pàgines enllaçades)
- http://www.ibm.com/developerworks/websphere
- Material UOC de l'assignatura Enginyeria del Programari.

I no pot faltar:

- http://www.google.com, l'autèntica Enciclopedia Universal.