TREBALL FINAL DE CARRERA

J2EE

AUTOR: Baldomero Baranco Sánchez (ETIG)
CONSULTOR: Salvador Campo Mazarico

Document que recull la Memòria del Treball Final de Carrera del projecte basat en la tecnologia J2EE.

MEMÒRIA

INDEX

LLICENCIA

INTRODUCCIÓ

Aquesta memòria presenta les línies generals que s'han seguit per tal d'implementar de manera funcional una aplicació de gestió de la compra/venta de cotxes en una empresa del sector automobilístic. En aquest document es recull de forma precisa les bases del nostre projecte utilitzant el llenguatge de programació JAVA i fent servir diferents *frameworks* per les diferents capes de la nostra lògica de negoci com Struts, Struts Tiles o Hibernate.

D'una banda, s'intenta definir el disseny de l'aplicació recollint les funcions més importants i relacionant-les en un diagrama de casos d'ús, tant entre elles mateixes com amb els actors que intervindran en el nostre sistema. Un cop tenim aquest llistat de funcions (o casos d'ús) decidirem els que són realment importants, i que implementarem com a màxima prioritat. Aquests casos d'ús seleccionats els mostrarem amb detall mitjançant fitxes de casos d'ús. Per acabar amb la fase de disseny mostrarem varis prototipus de finestres, que seran la cara de la nostra aplicació.

Un cop finalitzat el disseny, donarem pas a la seva especificació tècnica, relacionant tots els objectes que a priori intervindran en el sistema (ja siguin actors o usuaris del sistema com objectes que formen part de l'aplicació, com poden ser els cotxes) mitjançant un diagrama de classes. Un cop tenim el nostre diagrama de classes construït només caldrà transformar-lo en un diagrama relacional mitjançant el model entitat/relació, que originarà la construcció de la nostra futura base de dades.

Per acabar intentarem donar les claus de requeriment i configuració per a la implantació del nostre projecte anomenat **UNO**, fent servir la tecnologia J2EE. Especificarem l'estructura adoptada en el projecte fent servir pàgines WEB en format JSP per a la presentació visual (amb ajuda de Struts) i diferenciant la part de persistència i model de negoci (Fent servir Hibernate).

DESCRIPCIÓ DEL TFC

Aquest projecte final de carrera sorgeix de la necessitat d'una nova empresa del sector automobilístic de gestionar les seves vendes i adquisicions, així com portar un control sobre les activitats diàries dels venedors per un major control de la qualitat del servei.

L'empresa està orientada a la compra i venta de cotxes de segona mà, i el seu esquema de treball és el següent: L'empresa té un estoc, compost pels cotxes que tenen disponibles en cada moment per vendre. Quan un client arriba a la nostra empresa preguntant per algun dels cotxes en estoc, el que fem és donar-li una informació, que està composta de fins a tres ofertes. Aquestes ofertes poden o no ser del mateix cotxe.

Quan un client, un cop rebuda la informació, es decideix a comprar un cotxe tenim una comanda. Les comandes poden (o no) tenir un factor important en la nostra empresa, que és la reducció del preu de la comanda mitjançant un intercanvi de cotxes. Un client, si actualment té un cotxe, ens el pot entregar per reduir el preu del cotxe que comprarà. La única cosa que estem fent amb aquest cotxe es comprar-li i deduint-li el seu cost al nou cotxe que ell ens està comprant.

D'aquesta forma l'estoc s'anirà emplenant a mesura que també s'està venent. Quan es produeix un intercanvi de cotxes, el primer que necessitem sobre aquest cotxe és una taxació, que ens donarà les dades més importants sobre el vehicle, com la seva matrícula o el seu estat, però encara no estarà al nostre estoc. Quan una comanda es faci efectiva, aquest cotxe, del qual tenim prèviament una taxació, sí que formarà part del nostre estoc.

Els usuaris del sistema en principi seran de tres tipus: Administratius, comptables i venedors (L'administrador el deixem a part ja que sempre hi haurà un grup). Els administratius són els encarregats de donar el vist i plau a totes les transaccions, tant de compra com de venta. Els comptables són els encarregats de controlar totes les despeses i guanys que tindrà la empresa. Els venedors estan encarregats de la creació de la informació i tràmit de comandes, així com de les taxacions. En una futura ampliació podríem contemplar la opció per fer que els clients siguin usuaris del sistema, millorant així el servei ja que es podria consultar tot l'estoc disponible a casa.

ESTRUCTURACIÓ DEL PROJECTE

El projecte es dividirà en tres subsistemes, cadascun dels quals estarà composat per diferents mòduls agrupant així les funcionalitats que podrà realitzar cada usuari. Aquests sistemes a més, estaran relacionats entre sí per treballar les dades en comú.

Subsistema de ventes: Utilitzat pels venedors, tindran la opció de afegir, buscar i modificar clients. Podran crear ofertes per presentar-les al client, així com realitzar taxacions. A més, podran donar d'alta una comanda perquè sigui aprovada per la administració.

Subsistema d'administració: Utilitzat pels administratius, tindran la opció de afegir, buscar i modificar clients. Podran consultar les taxacions realitzades pels venedors. A més, podran donar d'alta o aprovar una comanda. En aquest subsistema també portaran el control de les nòmines dels treballadors.

Subsistema de comptabilitat: Utilitzat pels comptables, només tindran la opció per portar la comptabilitat, que això afectarà al accés a les nòmines dels treballadors així com totes les transaccions de compra/venta de la companyia, a més de portar un control de despeses com pot ser la llum o el lloguer del local.

OBJECTIUS GENERALS I ESPECÍFICS

L'administració i comptabilitat són subsistemes amb poca prioritat en quant a la seva implementació, per la qual cosa primera i exclusivament ens centrarem en el subsistema de ventes, que és el que ens permetrà començar a treballar amb la companyia. Això ens obligarà a alterar la funcionalitat de la nostra aplicació en un principi, deixant així que els propis venedors puguin acceptar informacions com a comandes vàlides i puguin afegir cotxes en estoc. (Això sí, sempre guiant-nos per les condicions exposades anteriorment)

Els objectius generals que es volen assolir amb aquest projecte són els següents:

- ✓ Familiaritzar-se amb la metodologia de treball de la tecnologia J2EE.
- ✓ Consolidar els coneixements apresos durant els estudis, tant de programació en JAVA com de Base de Dades.
- ✓ Realitzar una eina basada en estàndards i que sigui el màxim de reutilitzable.
- ✓ Aprendre a fer servir patrons de disseny per a les aplicacions.

Els objectius específics que es volen assolir concretament sobre el projecte són els següents:

- ✓ Dotar a l'empresa d'una gestió de dades ràpida i concisa.
- ✓ Centralització de dades, accessibles a tothom en qualsevol moment.
- ✓ Dotar al programa d'una interfície amigable i fàcil d'utilitzar.
- ✓ Afegir la opció de multi-llenguatge fent servir els estàndards i18n.

RELACIÓ DE LES DADES

⇒ Un client passa per dos estats diferents; Possible comprador, quan demana informació a la nostra empresa, sense que hagi comprat abans (podem tenir o no enregistrades les seves dades personals, ja que pot haver vingut abans a consultar altres ofertes, però si mai ens ha comprat cap cotxe no tindrem enregistrat el seu DNI/NIF). Comprador, quan ens ha comprat anteriorment un vehicle (Hem enregistrat a la base de dades el seu DNI per fer la factura).

⇒ Al ser una empresa que treballa amb cotxes de segona mà, existeixen convenis amb altres empreses del sector per els quals els venedors d'aquestes entitats exteriors ens poden facilitar vehicles, per la qual cosa no sempre coincideix que un venedor al mateix temps està autoritzat a taxar un vehicle. Únicament els venedors de la nostra pròpia empresa podran taxar els cotxes.

⇒ Un cotxe, abans d'arribar a l'estoc de la companyia haurà de passar per dos estats; El primer estat, anomenat taxació, on s'avalua el vehicle i es prenen les seves dades. No necessàriament tots els cotxes taxats arribaran al nostre estoc, ja que hi haurà taxacions fetes a possibles compradors que al final no siguin clients, o que al final no es decideixin per intercanviar el seu vehicle encara que si en comprin un de nou. El segon estat, anomenat cotxe ofertat, on sí estarà en el nostre estoc, producte d'haver consumat una comanda i un intercanvi amb un client.

⇒ Una informació està composta per entre una i tres ofertes. Una oferta fa referència a un cotxe disponible al nostre estoc, que sempre tindrà un mateix equipament i preu, però podran variar altres aspectes que variïn el preu final com pot ser més o menys mesos de garantia, o si el client vol intercanviar un cotxe per abaratir costos.

⇒ S'ha de tenir en compte que una matrícula no és necessàriament un identificador únic per segons quins aspectes de la nostra aplicació. Per exemple, podem tenir vàries taxacions d'un mateix cotxe (mateixa matrícula), però amb sis mesos de diferència, per la qual cosa el seu valor final ha disminuït.

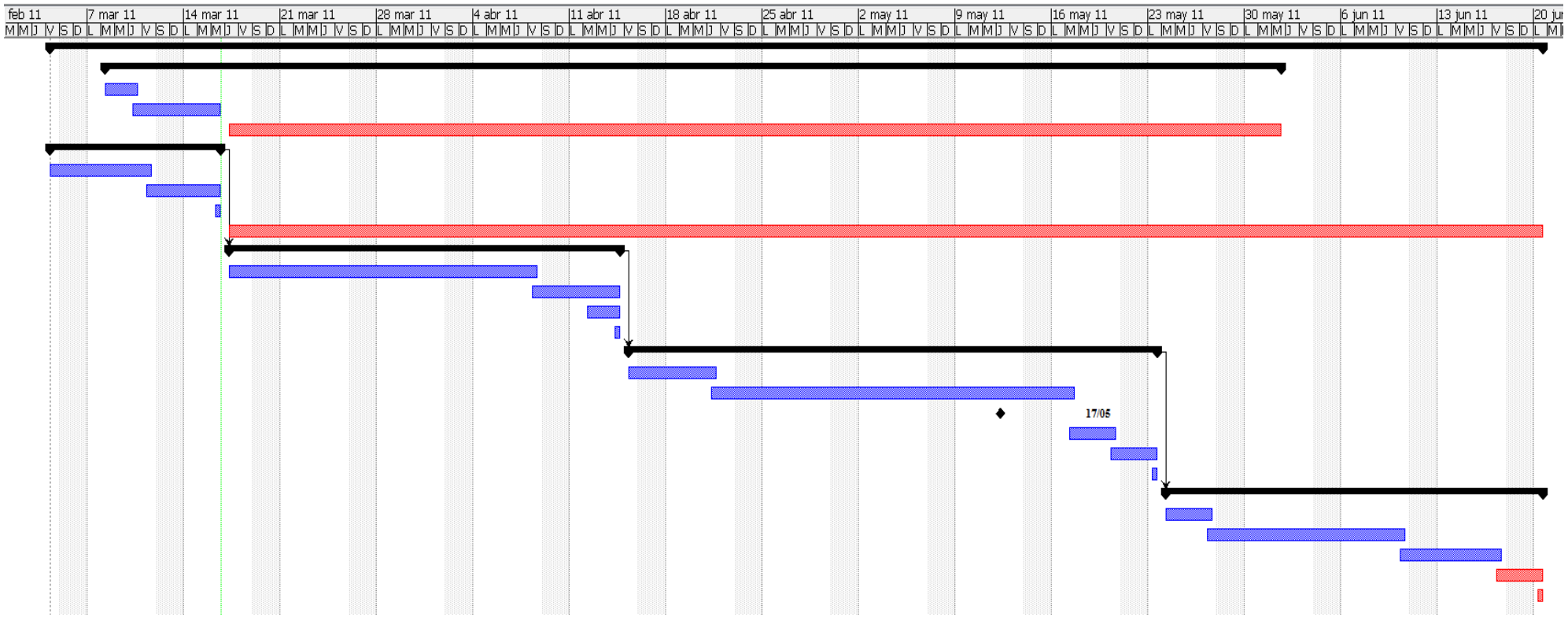
PLANIFICACIÓ I TEMPORALITZACIÓ

TASQUES A REALITZAR

Nombre	Duración	Inicio	Terminado
⊟TFC / J2EE	**77 days?**	**4/03/11 8:00**	**20/06/11 17:00**
⊟Estudi de la tecnologia J2EE	**62 days?**	**8/03/11 8:00**	**1/06/11 17:00**
Descripció J2EE	3 days?	8/03/11 8:00	10/03/11 17:00
Anàlisis de les eines de treball	5 days?	10/03/11 8:00	16/03/11 17:00
Aprofundiment tecnologia J2EE	55 days?	17/03/11 8:00	1/06/11 17:00
⊟PAC1	**9 days?**	**4/03/11 8:00**	**16/03/11 17:00**
Estudi del cas a proposar	6 days?	4/03/11 8:00	11/03/11 17:00
Redacció del Pla de Treball	4 days?	11/03/11 8:00	16/03/11 17:00
Entrega Pla de Treball	1 day?	16/03/11 8:00	16/03/11 17:00
Redacció Memòria	68 days?	17/03/11 8:00	20/06/11 17:00
⊟PAC2	**21 days?**	**17/03/11 8:00**	**14/04/11 17:00**
Disseny del Projecte	17 days?	17/03/11 8:00	8/04/11 17:00
Redacció del document	5 days?	8/04/11 8:00	14/04/11 17:00
Instal·lació de les eines necessàries	3 days?	12/04/11 8:00	14/04/11 17:00
Entrega Disseny del Projecte	1 day?	14/04/11 8:00	14/04/11 17:00
⊟PAC3	**27 days?**	**15/04/11 8:00**	**23/05/11 17:00**
Creació BDD	5 days?	15/04/11 8:00	21/04/11 17:00
Implementació JAVA subsistema Ventes	19 days?	21/04/11 8:00	17/05/11 17:00
Implantació JAVA altre subsistema ?	4 days?	12/05/11 8:00	17/05/11 17:00
Proves	4 days?	17/05/11 8:00	20/05/11 17:00
Redacció Manual Usuari	2 days?	20/05/11 8:00	23/05/11 17:00
Entrega PAC3	1 day?	23/05/11 8:00	23/05/11 17:00
⊟PAC4	**20 days?**	**24/05/11 8:00**	**20/06/11 17:00**
Resolució d'imprevistos trobats	4 days?	24/05/11 8:00	27/05/11 17:00
Compactament/redacció final Memòria	11 days?	27/05/11 8:00	10/06/11 17:00
Presentació	6 days?	10/06/11 8:00	17/06/11 17:00
Resolució final	2 days?	17/06/11 8:00	20/06/11 17:00
Entrega Memòria + Presentació	1 day?	20/06/11 8:00	20/06/11 17:00

Fent servir OpenProj: http://openproj.org/

DIAGRAMA DE GANTT



FUNCIONALITATS DEL PROJECTE

Com hem descrit anteriorment, la idea del projecte consta de tres seccions treballant cadascuna de forma independent, però únicament s'implementarà la primera i més important: La secció de ventes. El projecte implementa una forma fàcil i còmoda per administrar les ventes per part dels treballadors. Hem de tenir en compte que en cap moment és un sistema per interactuar amb l'usuari, sinó un mitjà pel qual el treballador pot buscar i modificar la informació necessària per presentar-la al client.

Per utilitzar l'aplicació serà necessari tenir un usuari i una contrasenya donats d'alta a la base de dades. Cada usuari tindrà, de forma independent, una sèrie de permisos per utilitzar les diferents funcionalitats del programa. Entre aquestes funcionalitats es troben la gestió de taxacions, la gestió d'usuaris, la gestió de cotxes en estoc i la gestió d'informació.

- La gestió de taxacions consisteix en la possibilitat de modificar una taxació enregistrada anteriorment al sistema, crear una nova taxació en blanc, o crear una segona taxació. Una segona taxació consisteix en crear una nova taxació a partir d'una taxació existent al sistema, seleccionant les dades més importants. A més, permet llistar totes les taxacions gravades al sistema i filtrar-les segons uns paràmetres determinats com són per matrícula i marca del cotxe taxat. Per altra banda, aquests filtres són additius, la qual cosa vol dir que es poden fer servir cap, un o dos filtres alhora.

- La gestió d'usuaris consisteix en la possibilitat de modificar o crear un nou usuari al sistema. Existiran dos tipus de nivell de seguretat per gestionar els usuaris: Un primer nivell, on es podran modificar únicament una sèrie de dades bàsiques com podran ser el nom o la direcció de l'usuari, i un segon nivell on es podrà modificar qualsevol dada com poden ser els seus permisos o el seu *username*. A més, permet llistar tots els usuaris donats d'alta al sistema i filtrar les dades segons els cognoms dels usuaris.

- La gestió de cotxes en estoc consisteix en la possibilitat de modificar o crear un nou cotxe al sistema. Aquests cotxes són els que esta disponibles per a la venta o estan a l'espera de poder ser venuts. A més, permet llistar tots els cotxes existents al sistema i filtrar-los segons el seu tipus d'estoc.

- La gestió d'informació consisteix en la possibilitat de modificar o crea una nova informació al sistema. Les informacions consisteixen en un llistat de fins a tres ofertes que es presenten a un client per vendre un cotxe. A més, permet llistar les informacions segons els cognoms dels clients que han demanat alguna oferta en algun moment.

DISSENY

ACTORS DEL SISTEMA

Primerament hem d'identificar els actors del nostre sistema descrivint cadascuna de les accions que podran realitzar, definint així el seu rol. Aquestes accions s'identificaran com els casos d'ús que s'implementaran en la nostra aplicació i es relacionaran en un diagrama.

⇒ ***Venedor***: És necessària la seva identificació[1] en el sistema abans de poder realitzar cap acció. Les seves funcions són les de mantenir el llistat de clients (afegint nous clients o modificant dades dels ja existents), crear ofertes[2] i donar d'alta comandes[2].

⇒ ***Taxador***: És una extensió de l'actor Venedor, així que a més de poder realitzar totes les accions d'un venedor, a més podrà afegir taxacions[2].

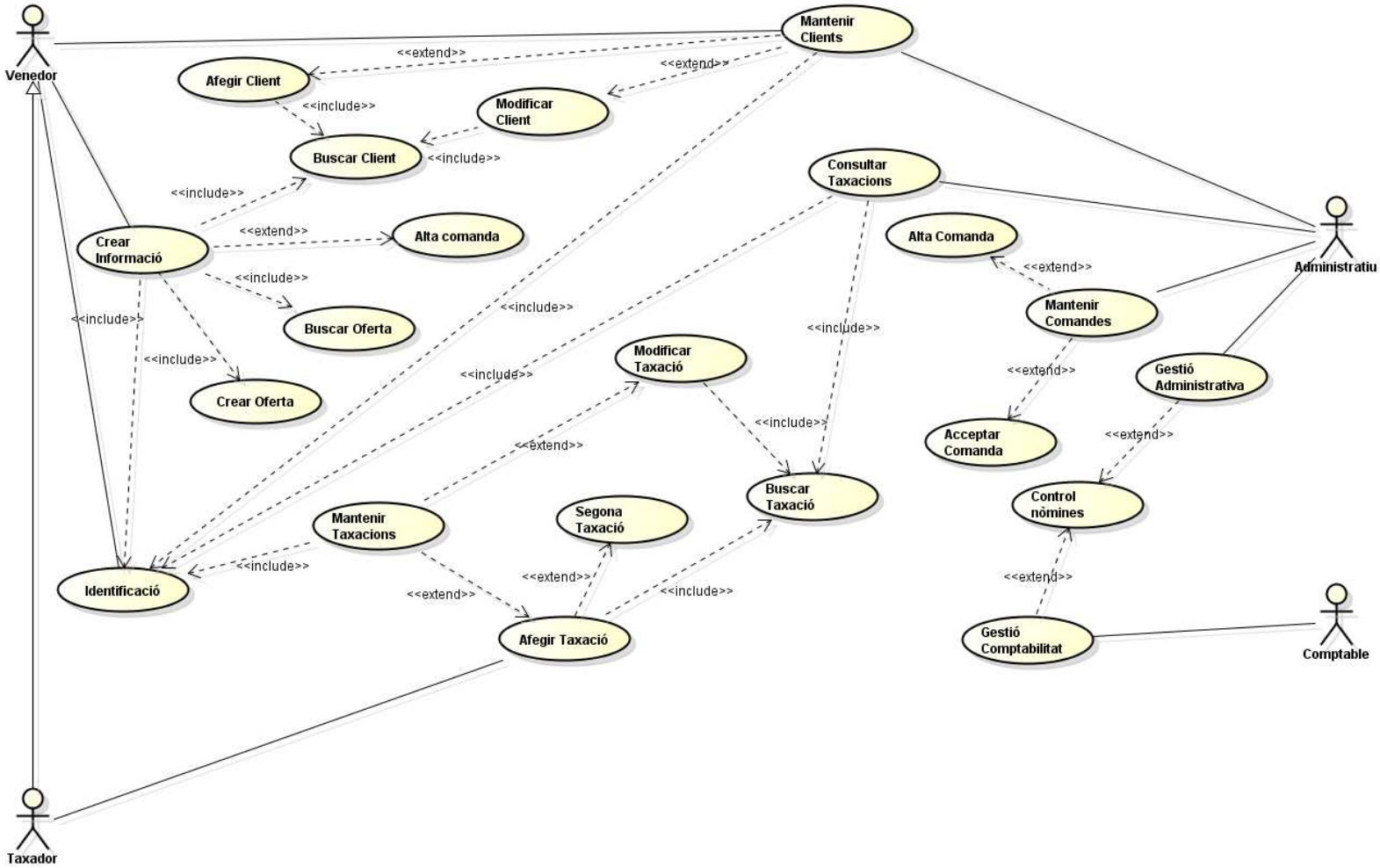
⇒ ***Administratiu***: És necessària la seva identificació en el sistema abans de poder realitzar cap acció. Les seves funcions són les de mantenir el llistat de comandes, ja sigui creant una nova comanda o acceptant una comanda ja donada d'alta al sistema, també poden mantenir el llistat de clients (afegint nous clients o modificant dades dels ja existents), consultar taxacions ja existents al sistema i realitzar tot tipus de gestions administratives.

⇒ ***Comptable***: És necessària la seva identificació en el sistema abans de poder realitzar cap acció. Les seves funcions principals són les de gestionar la comptabilitat.

⇒ ***Administrador***[3]: És necessària la seva identificació en el sistema abans de poder realitzar cap acció. Podrà realitzar qualsevol acció implementada en el sistema. A més, tindrà una sèrie de casos d'ús complementaris als que es mostren en el diagrama de la pàgina següent, necessaris per a la administració de la aplicació com la eliminació de clients o taxacions.

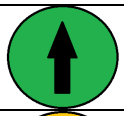
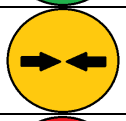
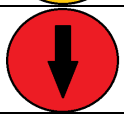
[1] Encara que la identificació és obligatòria per tots els casos d'ús que exposarem en aquest document, podria ser possible que en un futur existís un actor anomenat Invitat que pugui consultar de forma restrictiva alguna informació.
[2] Consultar Glossari al final del document.
[3] Encara que no estigui present en els nostres casos d'ús sí que estarà en el nostre sistema.

CASOS D'ÚS


Venedor
Mantenir Clients
Afegir Client
Modificar Client
Buscar Client
Consultar Taxacions
Crear Informació
Alta comanda
Alta Comanda
Administratiu
Buscar Oferta
Crear Oferta
Modificar Taxació
Mantenir Comandes
Gestió Administrativa
Acceptar Comanda
Control nòmines
Buscar Taxació
Mantenir Taxacions
Segona Taxació
Identificació
Afegir Taxació
Gestió Comptabilitat
Comptable
Taxador
<<extend>>
<<include>>

Un cop hem identificat els actors del sistema, creem el diagrama de casos d'ús que es mostra en la pàgina anterior, el qual ens servirà com a base per a llistar tots els casos d'ús i assignar a cadascun d'ells una prioritat d'implementació, que ens indicarà la seva importància dins el sistema i ens servirà per decidir quines són les funcions a implementar de forma immediata i quines es poden deixar de banda en un principi.

PRIORITAT IMPLEMENTACIÓ DE CASOS D'ÚS

Icona	***Descripció***
	Prioritat alta, s'implementarà en el nostre sistema inicial ja que és una funció bàsica del nostre sistema.
	Prioritat mitjana, es podria implementar en el nostre sistema encara que no és cap funció crítica.
	Prioritat baixa, a menys que s'hagin implementat tots els casos d'ús de prioritat alta i mitjana no s'implementaran.

Prioritat	***Cas d'ús***
	**Identificació**
	**Crear Informació** → (*Buscar Oferta* / *Crear Oferta*)
	(Mantenir Clients) ↔ **Afegir Client** → (*Buscar Client*)
	(Mantenir Taxacions) ↔ **Afegir Taxació** → (*Buscar Taxació*)
	**Segona Taxació**
	**Modificar Client**
	**Modificar Taxació**
	**Alta Comanda**
	(Mantenir Comandes) ↔ **Acceptar Comanda**
	**Gestió Comptabilitat**
	**Gestió Administrativa**
	**Control Nòmines**

Com podem comprovar, hi ha alguns casos d'ús que són dependents d'altres, com per exemple el cas d'ús de prioritat alta *Afegir Client*, que depèn del cas d'ús *Buscar Client*, per la qual cosa ambdós han de ser implementats al mateix temps.

Hi ha casos d'ús independents, que formen part d'un altre cas d'ús la funció del qual és la suma de diferents casos d'ús, com per exemple en el cas d'ús de prioritat baixa *Acceptar Comanda*, que pertany al cas d'ús *Mantenir Comandes* però pot treballar de forma independent.

FITXES DELS CASOS D'ÚS MÉS SIGNIFICATIUS

Un cop identificats els casos d'ús més importants seleccionem els més rellevants i mostrem de forma detallada algunes de les accions més importants. Aquesta llista està composta dels casos d'ús considerats crítics pel nostre sistema. Si ens fixem, hi ha casos d'ús als quals no se'ls atorga una importància rellevant al sistema a priori, com són els referents a la modificació de dades. Això és així perquè en aquest cas treballem amb unes dades molt *estàtiques* i poc propenses a canviar, a més la duplicació de la majoria de dades no és del tot rellevant.

Cas d'ús: Crear Informació	
Resum:	Usat pels venedors per mostrar les ofertes als clients.
Actors:	Venedor
Precondició:	La informació no existeix al sistema i l'usuari està identificat
Postcondició:	Nova informació al sistema
Casos d'ús relacionats:	**Buscar Client**, **Buscar oferta**, **Crear Oferta**
Flux Principal:	1. Es busca el client al que va destinada la Informació. 2. Es busquen les ofertes que compondran la Informació. 3. Es genera la Informació amb les ofertes escollides. 4. S'imprimeix la Informació per al client.
Flux Alternatiu:	1.1 Si no existeix el client al sistema es crea un de nou. 2.1 Si no existeix la oferta desitjada es crea una de nova.

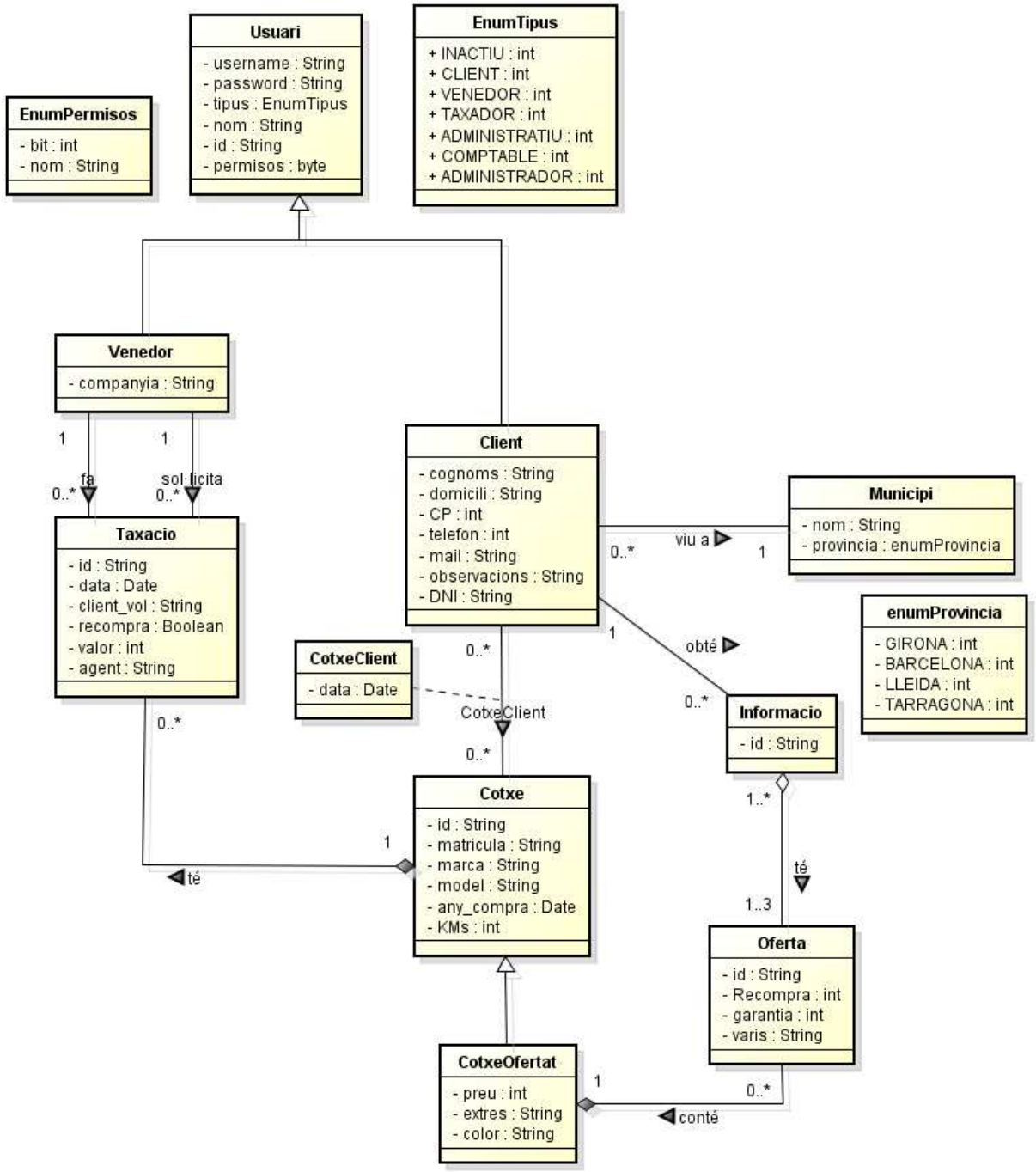
Cas d'ús: Afegir Client	
Resum:	Usat pels venedors per donar d'alta un nou possible client.
Actors:	Venedor, Administratiu
Precondició:	Usuari identificat
Postcondició:	Existeix un nou client al sistema
Casos d'ús relacionats:	**Buscar Client**
Flux Principal:	1. S'introdueixen les dades del client. 2. Es comprova que el client no existeixi. 3. Es dóna d'alta el nou client amb les dades introduïdes.
Flux Alternatiu:	2.1 Si ja existeix el client es permeten modificar les seves dades.

Cas d'ús: Afegir Taxació	
Resum:	Usat per taxadors per donar d'alta una nova taxació.
Actors:	Taxador
Precondició:	Usuari identificat
Postcondició:	Existeix una nova taxació al sistema
Casos d'ús relacionats:	**Buscar Client, Buscar Taxació, Segona Taxació**
Flux Principal:	1. Es comprova que no existeixi la taxació al sistema. 2. S'introdueixen les dades del cotxe necessàries per a la taxació.
Flux Alternatiu:	1.1 Si ja existeix la taxació es realitza una **Segona Taxació**.

Cas d'ús: Segona Taxació	
Resum:	Usat per taxadors per donar d'alta una segona taxació.
Actors:	Taxador
Precondició:	Existeix la Taxació i l'usuari està identificat
Postcondició:	Existeix una nova taxació al sistema
Casos d'ús relacionats:	**Afegir Taxació**
Flux Principal:	1. Es crea una nova taxació amb les mateixes dades que la taxació donada. 2. Es pot modificar el valor de la taxació.
Flux Alternatiu:	

ESPECIFICACIÓ

DIAGRAMA DE CLASSES



Tal com podem veure en el dibuix de la pàgina anterior que ens mostra el diagrama de classes es composa d'una estructura estàndard, encara que hi ha diverses restriccions no presents en aquest diagrama UML que sí s'hauran d'implementar en el nostre sistema final com per exemple que únicament es podran assignar en noves taxacions venedors com a taxadors únicament si són del tipus TAXADOR.

També hem de destacar la relació entre una ***Taxació*** i un ***Cotxe***, i una ***Oferta*** i un ***CotxeOfertat***. En ambdós casos és una relació de composició ja que, en el primer cas no té cap sentit tenir una taxació sense cotxe o, en el segon cas, una oferta sense cap cotxe a vendre.

Per finalitzar fixem-nos en la relació entre ***Oferta*** i ***Informació***, que pertany a una relació d'agregació i, a més, amb un cardinal atípic de 1..3.

DIAGRAMA RELACIONAL

Primerament obtindrem el disseny estàtic de les taules de la base de dades mitjançant el diagrama de classes proposat en la pàgina anterior, tot eliminant l'herència i reflectint les associacions entre classes.

Companyia(companyia)

enumTipus(idTipus, nom)

Provincia(provincia)

Municipi(municipi, provincia)

{provincia és clau forana de Provincia}

Usuari(idUsuari, companyia, municipi, *username*, password, nom, tipus, permisos, cognoms, domicili, CP, telefon, mail, observacions, DNI)

{companyia és clau forana de Companyia}
{municipi és clau forana de Municipi}
{tipus és clau forana de enumTipus}

Agent(agent)

Marca(marca)

Extres(idExtres, extra)

Color(color)

Cotxe(idCotxe, color, marca, matricula, model, any_compra, KMs, preu, extres, estoc)

{color és clau forana de Color}
{marca és clau forana de Marca}

Taxacio(idTaxacio, cotxe, venedor, taxador, agent, data, client_vol, recompra, valor)

{cotxe és clau forana de Cotxe}
{venedor i taxador són clau forana de Usuari}
{agent és clau forana de Agent}

CotxeClient(idUsuari, idCotxe, data)

{idUsuari és clau forana de Usuari}
{idCotxe és clau forana de Cotxe}

Oferta(idOferta, cotxe, recompra, garantia, varis)

{cotxe és clau forana de Cotxe}

Informacio(idInformacio, client, oferta1, oferta2, oferta3)

{client és clau forana de Usuari}
{oferta1, oferta2, oferta3 són clau forana de Oferta}

Un cop hem definit les taules amb les entitats i relacions podem obtenir el model Entitat/Relació que ens conduirà finalment al diagrama relacional que es mostra en la pàgina següent. Per limitacions amb la generació del diagrama no es mostren de forma correcta vàries relacions i atributs, per la qual cosa especificarem els més importants:

⇒ En la taula Informació oferta2 i oferta3 poden obtenir el valor nul, no així oferta1.

⇒ Pels atributs del tipus BIT (permisos a Usuari; extres a Cotxe) la idea és aplicar el concepte de *flag*. Per exemple, si existeix un extra amb *idExtres* igual a 1, que per exemple és cinc portes, i un segon extra amb *idExtres* igual a 2, que per exemple és clau a distància, si un cotxe té un valor de 3 en extres sabrem quin llistat d'extres té, sense haver de fer taules secundàries.

⇒ L'atribut *username* a la taula Usuari és una clau secundària, ja que sempre serà única, el que passa és que pot no ser fixa ja que en un futur ens podria interessar que els propis usuaris puguin canviar-la.

⇒ Com podem veure hi ha moltes taules simples d'un o dos atributs. Això és especialment important ja que la major part del temps es treballa amb les

mateixes dades fixes que no canvien, i és important tenir els valors unificats. A més cada cert temps es fa una ampliació d'aquestes, com per exemple a la taula Companyia aparegui un nou proveïdor de cotxes.

⇒ Un venedor pot ser al mateix temps taxador, per la qual cosa es pot donar el cas que al mateix que un venedor sol·licita una taxació, ell mateix és el taxador.

En la pàgina següent mostrem el diagrama relacional amb els atributs de totes les taules, les seves relacions i els seus tipus de dades.

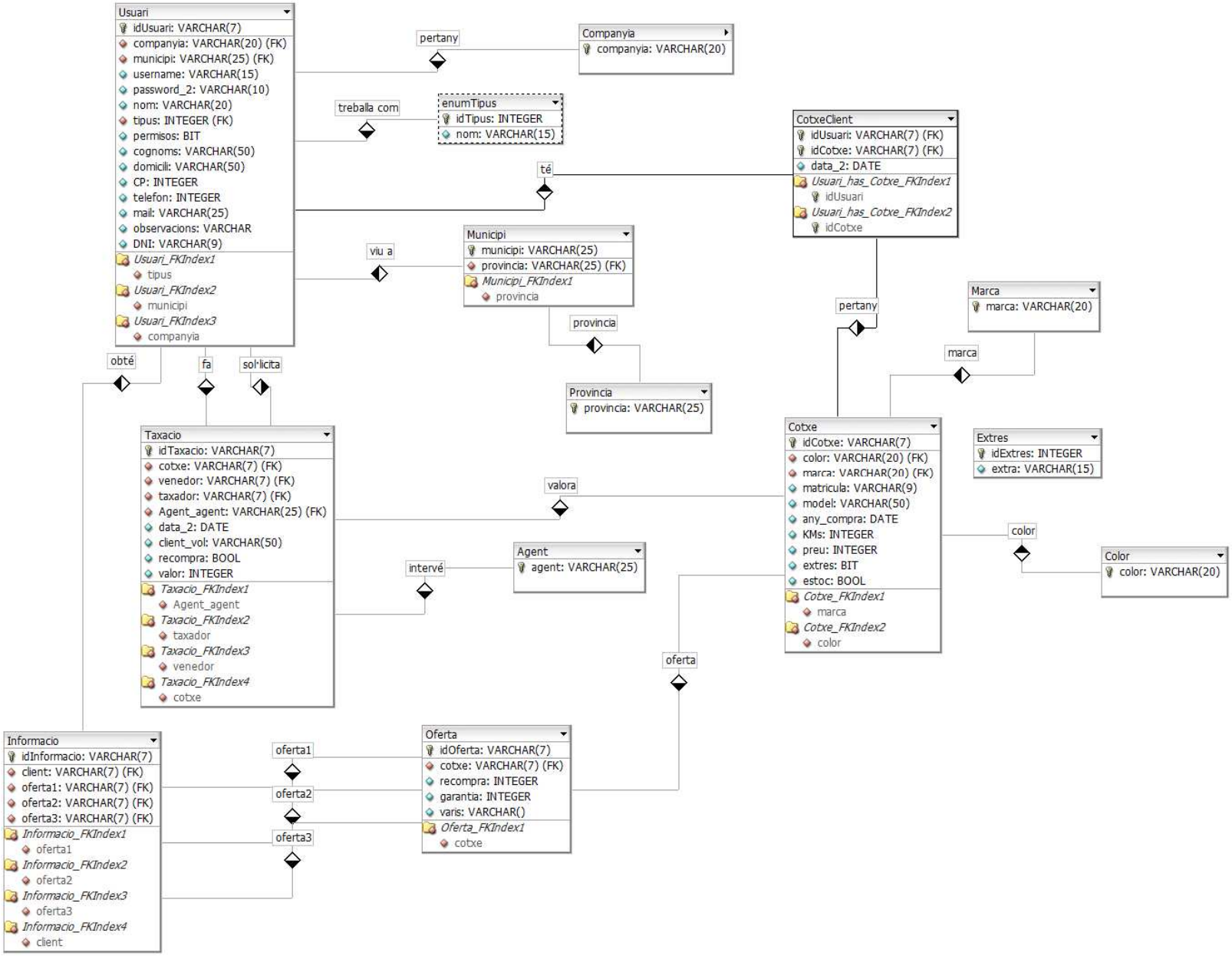
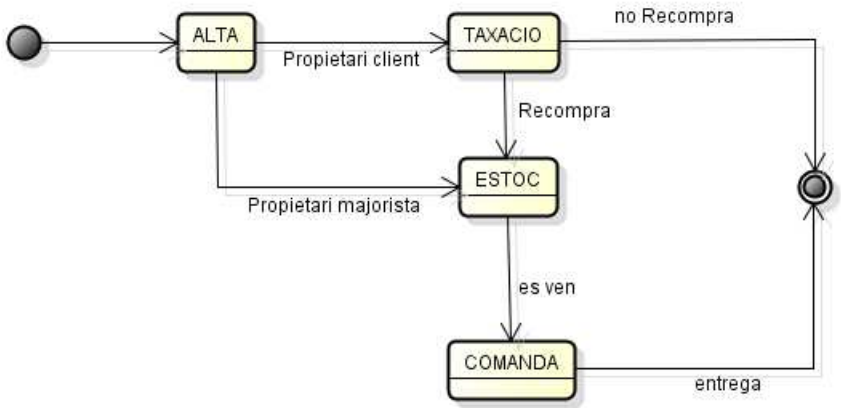

Usuari
idUsuari: VARCHAR(7)
companyia: VARCHAR(20) (FK)
municipi: VARCHAR(25) (FK)
username: VARCHAR(15)
password_2: VARCHAR(10)
nom: VARCHAR(20)
tipus: INTEGER (FK)
permisos: BIT
cognoms: VARCHAR(50)
domicili: VARCHAR(50)
CP: INTEGER
telefon: INTEGER
mail: VARCHAR(25)
observacions: VARCHAR
DNI: VARCHAR(9)
Usuari_FKIndex1
tipus
Usuari_FKIndex2
municipi
Usuari_FKIndex3
companyia
pertany
Companyia
companyia: VARCHAR(20)
treballa com
enumTipus
idTipus: INTEGER
nom: VARCHAR(15)
té
CotxeClient
idUsuari: VARCHAR(7) (FK)
idCotxe: VARCHAR(7) (FK)
data_2: DATE
Usuari_has_Cotxe_FKIndex1
idUsuari
Usuari_has_Cotxe_FKIndex2
idCotxe
viu a
Municipi
municipi: VARCHAR(25)
provincia: VARCHAR(25) (FK)
Municipi_FKIndex1
provincia
provincia
Provincia
provincia: VARCHAR(25)
pertany
Marca
marca: VARCHAR(20)
marca
obté
fa
sol·licita
Taxacio
idTaxacio: VARCHAR(7)
cotxe: VARCHAR(7) (FK)
venedor: VARCHAR(7) (FK)
taxador: VARCHAR(7) (FK)
Agent_agent: VARCHAR(25) (FK)
data_2: DATE
client_vol: VARCHAR(50)
recompra: BOOL
valor: INTEGER
Taxacio_FKIndex1
Agent_agent
Taxacio_FKIndex2
taxador
Taxacio_FKIndex3
venedor
Taxacio_FKIndex4
cotxe
valora
Cotxe
idCotxe: VARCHAR(7)
color: VARCHAR(20) (FK)
marca: VARCHAR(20) (FK)
matricula: VARCHAR(9)
model: VARCHAR(50)
any_compra: DATE
KMs: INTEGER
preu: INTEGER
extres: BIT
estoc: BOOL
Cotxe_FKIndex1
marca
Cotxe_FKIndex2
color
Extres
idExtres: INTEGER
extra: VARCHAR(15)
color
Color
color: VARCHAR(20)
intervé
Agent
agent: VARCHAR(25)
oferta
Informacio
idInformacio: VARCHAR(7)
client: VARCHAR(7) (FK)
oferta1: VARCHAR(7) (FK)
oferta2: VARCHAR(7) (FK)
oferta3: VARCHAR(7) (FK)
Informacio_FKIndex1
oferta1
Informacio_FKIndex2
oferta2
Informacio_FKIndex3
oferta3
Informacio_FKIndex4
client
oferta1
oferta2
oferta3
Oferta
idOferta: VARCHAR(7)
cotxe: VARCHAR(7) (FK)
recompra: INTEGER
garantia: INTEGER
varis: VARCHAR()
Oferta_FKIndex1
cotxe

DIAGRAMA D'ESTATS D'UN COTXE

A continuació mostrem els diferents estats pels que passa la classe clau del nostre projecte: Un cotxe.

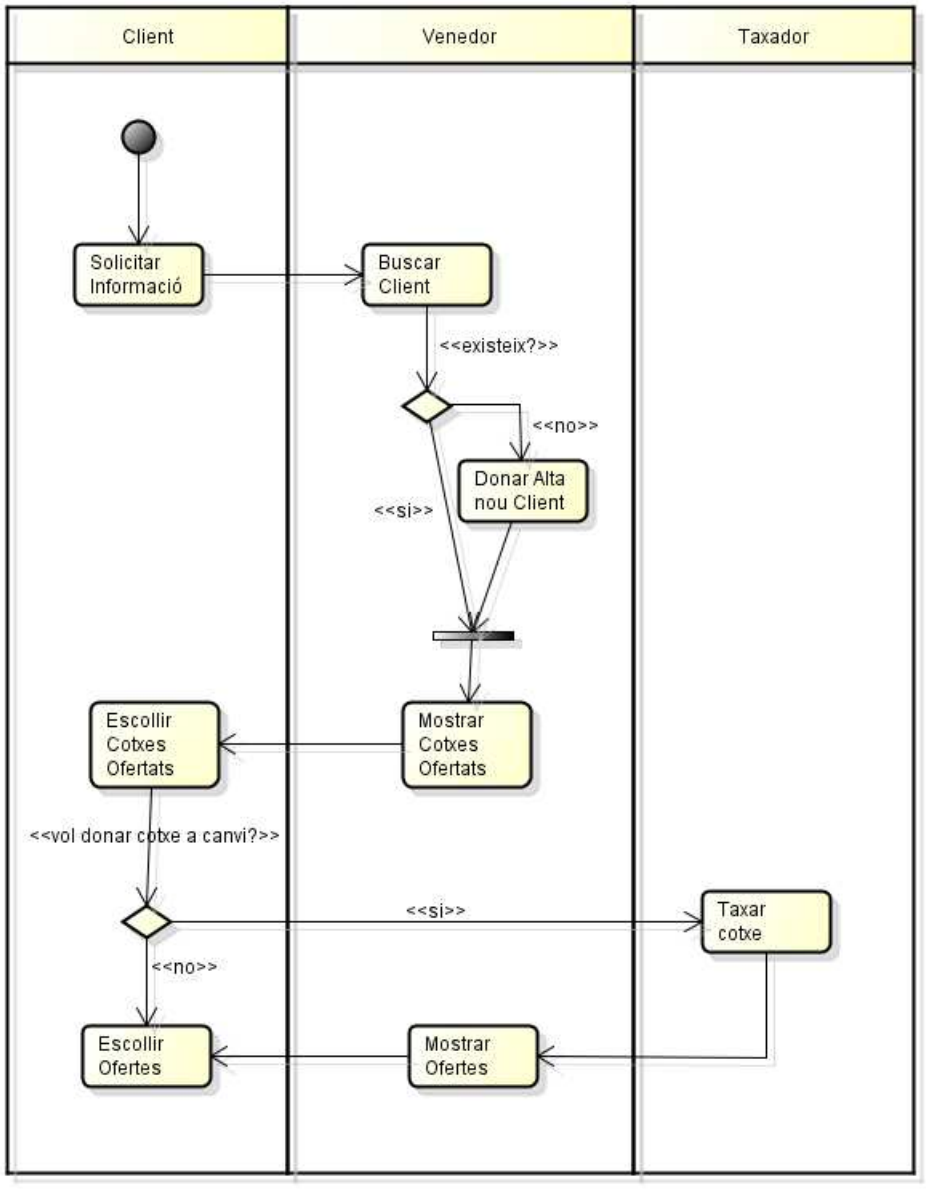


Quan donem d'alta un cotxe al nostre sistema poden passar dues coses, que el propietari del cotxe sigui un client, per la qual cosa primer haurem de fer una taxació per valorar aquest cotxe, o que el propietari sigui un majorista, per la qual cosa el podríem introduir directament a estoc. En el cas que el propietari sigui un client, si hi ha una recompra per part nostra (el client ens compra un cotxe i fa un intercanvi per abaratir costos) aquest cotxe passa al nostre estoc. Quan un cotxe del nostre estoc té un comprador, aquest passa a una comanda a l'espera de ser entregat i s'elimina del nostre estoc.

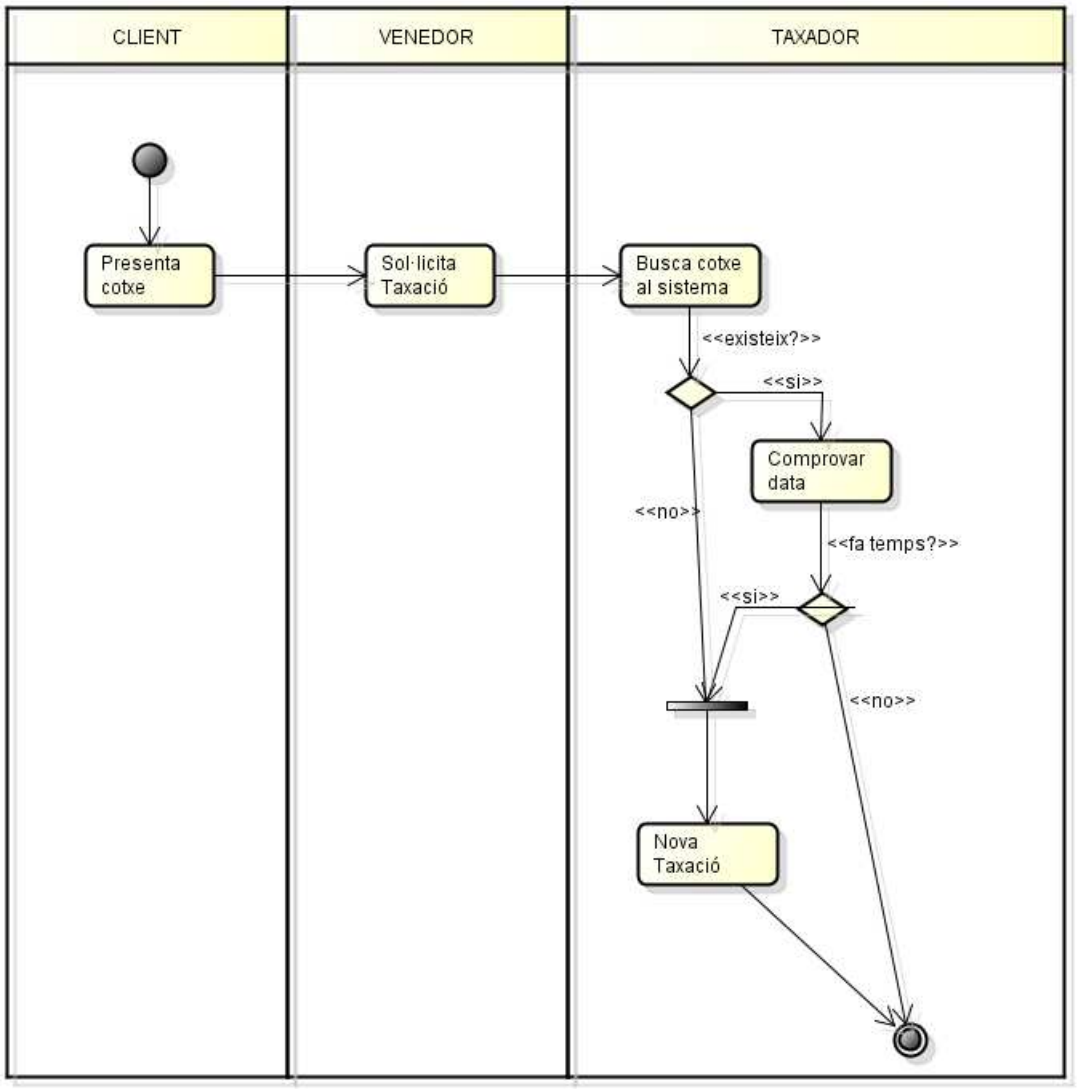
DIAGRAMA D'ACTIVITAT

A mode de referència mostrem dos diagrames d'activitat de dos casos d'ús més representatius com són la creació d'una informació pel client i la creació d'una taxació.

CREAR INFORMACIÓ

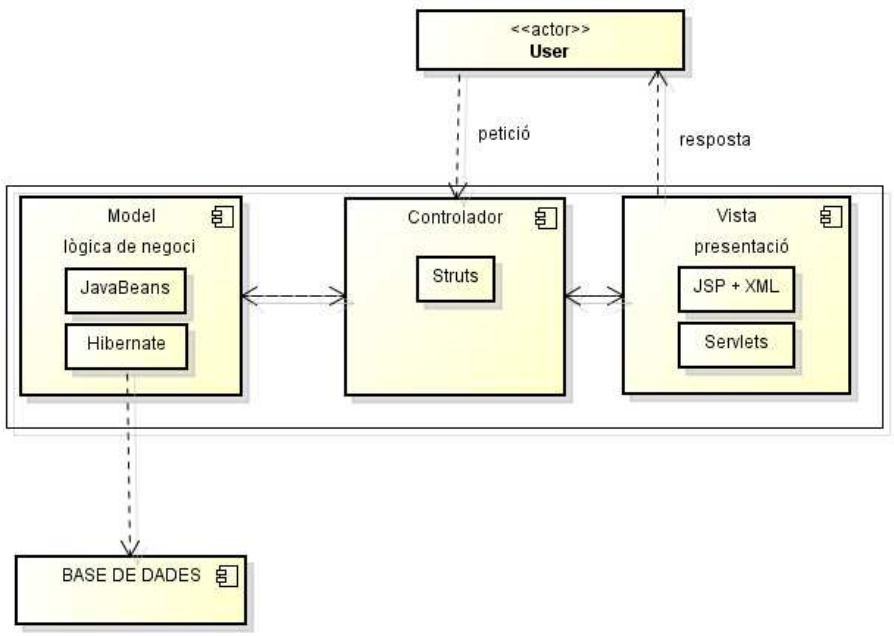


AFEGIR TAXACIÓ



ARQUITECTURA DE L'APLICACIÓ

La nostra és una aplicació del tipus client/servidor, per la qual cosa farem diverses divisions segons el tipus de cada recurs emprat. Ja que fem servir la tecnologia J2EE, afavorint així la implementació de patrons del tipus Model-Vista-Controlador (MVC), agruparem els diferents mètodes en capes, de forma que tots els recursos es distribueixin de forma coherent per poder tenir accés mitjançant una intranet (o Internet).



- **Struts** ens proporciona de forma inherent els controladors necessaris per a la nostra aplicació, creant així de forma natural el patró d'implementació MVC. La part del controlador és la responsable de rebre i distribuir les diferents accions sol·licitades.

- **Hivernate** és un *framework*[4] utilitzat per a la realització de la persistència d'objectes relacionats amb bases de dades.

- **JavaBeans** (o en aquest cas Enterprise JavaBeans o EJB) són les API que donen les pautes per a la utilització d'objectes per part dels servidors. Juntament amb Hivernate conformen el model del patró, encarregat de la gestió de la base de dades (la seva comunicació) i la lògica de negoci (la seva estructura).

[4] Consultar Glossari al final del document.

- **JSP/XML/Servlets** conformen el conjunt final del model MVC, referent a la vista, el qual fa referència a l'aparença (pantalles) per a la comunicació amb l'usuari de l'aplicació. En aquest apartat hem de fer menció de Tomcat com a contenidor de Servlets que farem servir.

- **Tiles** és un *framework* que ens serveix per crear plantilles que ens ajudaran a donar una aparença uniforme a totes les pàgines de la nostra aplicació. Tiles és un complement molt útil que es pot fer servir juntament amb les possibilitats que ens aporta Struts en la creació de temes.

REQUERIMENTS DE SOFTWARE

Els requeriments de software d'aquest projecte són bàsicament dos: Un programa de BDD (base de dades) que en el nostre cas serà Mysql. Un programa de contenidor de servlets que en el nostre cas serà Tomcat.

De forma addicional definirem la tecnologia amb la que hem treballat per a realitzar el projecte, encara que no és necessària cap instal·lació afegida ni cap requeriment complementari i a la distribució final seran presents totes les llibreries necessàries per fer funcionar el TFC. El projecte es basa en el framework Hibernate per treballar amb la persistència i comunicació amb la BDD (Mysql) i el framework Struts2 per treballar amb el servlets i les pàgines jsp.

Mysql com a Base de Dades

Com ja hem comentat hem escollit Mysql com a gestor de la nostra base de dades. La versió utilitzada per a la realització de la implantació de tot el sistema ha sigut la 5.1.54, instal·lada sobre la plataforma Linux Ubuntu (Extreta directament dels repositoris per defecte de la distribució). En un document adjunt (anomenat bdd_script.sql) estan totes les comandes per a la creació de totes les taules que es faran servir al projecte, així com les definicions de totes les seves claus (primàries i foranes, o claus úniques) i les mesures a executar en cas de modificació/eliminiació. En un altre document (anomenat inserts.sql) hi ha una petita mostra amb diferents exemples de dades per introduir-los a la BDD.

¡**Important!:** Per a la implementació de la persistència es fa servir Hibernate amb la tecnologia @Annotations per a la especificació, i aquesta fa referència de forma concreta al nom del nostre esquema creat a la BDD (En el nostre cas l'esquema té el nom TFC) per la qual cosa és imperatiu crear un esquema amb el mateix nom per evitar possibles incompatibilitats. A més a més, Hibernate fa referència a un nom d'usuari i contrasenya de la nostra BDD de forma directa als seus arxius de

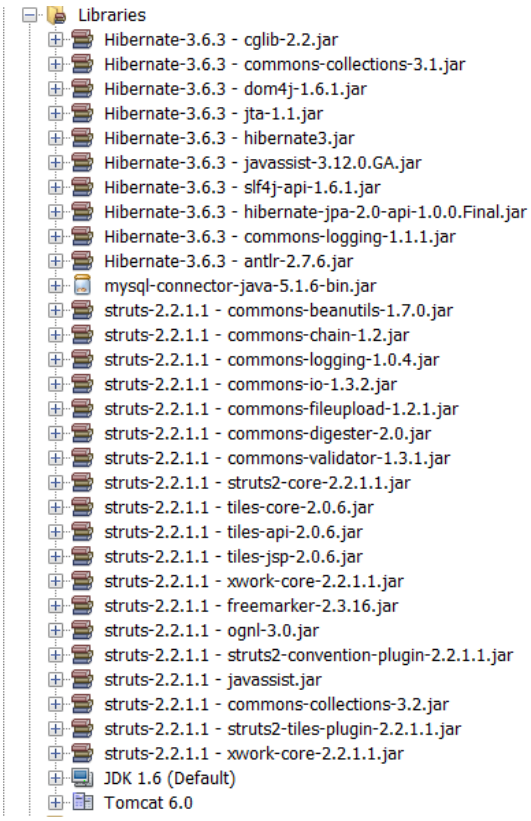
configuració (En el nostre cas l'usuari és 'TFC' i la contrasenya 'lalala') per la qual cosa necessitarem un usuari i contrasenya iguals en la nostra BDD o, en el seu defecte, haurem de modificar la configuració. Més endavant explicarem com modificar aquestes dades en el cas en què vulguem modificar-les.

Tomcat com a contenidor de Servlets

Tomcat ha sigut l'aplicació escollida com a servidor per emmagatzemar i poder tenir accés a les nostres pàgines web escrites en JSP. La versió utilitzada en el nostre cas és la 6.0.32 instal·lada sobre Windows7. No s'ha de fer cap configuració especial pel nostre projecte, únicament la configuració estàndard de posada en marxa d'un servidor Tomcat indicant les rutes de Java i directori arrel de Tomcat, així com un usuari i contrasenya inicial per un administrador de Tomcat.

El projecte es distribuirà en un arxiu WAR, que Tomcat serà capaç d'interpretar. Per a la instal·lació del projecte al servidor únicament hem de copiar aquest arxiu WAR (en el nostre cas anomenat 'S-car.war') en el directori 'webapp' de Tomcat i ell automàticament farà la resta. Si tot ha sortit de forma correcta, en el mateix directori on hem copiat l'arxiu WAR tindrem una carpeta anomenada S-car amb el nostre projecte.

Per tal que l'aplicació funcioni de forma correcta ens hem d'assegurar que tenim instal·lades totes les llibreries necessàries al directori '\S-car\WEB-INF\lib'. A continuació detallem una llista amb les llibreries necessàries per executar el projecte sense problemes.



Framework Hibernate

Com ja hem indicat anteriorment, Hibernate (la versió utilitzada en el projecte ha sigut la 3.6.3) és un Framework per a la persistència d'objectes amb la nostra BDD. La seva configuració es pot trobar a la carpeta 'S-car\WEB-INF\classes' amb el nom 'hibernate.cfg.xml'. Per adaptar la nostra aplicació al nostre entorn segurament haurem de modificar alguna de les línies d'aquest arxiu:

- `name="hibernate.connection.url">jdbc:mysql://192.168.1.102:3306/TFC`

  Aquesta línia ens indica la direcció del nostre servidor on es troba la nostra BDD de Mysql i el port per on connectar-se. En aquest cas el servidor Mysql es troba en una màquina d'una intranet amb la direcció 192.168.1.102 fent servir el port 3306 (port per defecte de Mysql). De forma general s'haurà de canviar aquesta direcció per la local 127.0.0.1 (I canviar el número de port en cas que s'hagi modificat la configuració per defecte de Mysql) 'TFC' fa referència al nom de l'esquema al que es connectarà per defecte.

- `<property name="hibernate.connection.username">TFC</property>`

  Aquesta línia ens indica el nom d'usuari que es farà servir per connectar amb la BDD de Mysql. De forma general no és recomanable fer servir els usuaris root o administrador per aquest tipus de connexions ja que la contrasenya no està encriptada a la configuració, per la qual cosa seria convenient crear un nou usuari. En el cas en que ja tinguem un usuari creat només haurem de canviar 'TFC' pel nom del nostre usuari.

- `<property name="hibernate.connection.password">lalala</property>`

  Línia que ens indica la contrasenya de l'usuari de la BDD de Mysql.

- `<property name="hibernate.show_sql">true</property>`

  Aquesta línia no és realment obligatòria. Amb aquesta sentència i un valor a 'true' estem indicant a Tomcat que mostri als seus arxius de registre que mostri les sentències SQL fetes per Hibernate. En principi no hi ha cap problema en tenir activades aquestes sentències, ja que totes les variables es criden de forma referencial, de tal forma que mai es mostraran les dades introduïdes per l'usuari, i aquesta informació és útil mentre es desenvolupa l'aplicació, però sempre es pot desactivar canviant el valor a 'false'.

La resta d'opcions de configuració s'haurien de deixar intactes pel bon funcionament de la aplicació.

Per a la implantació de la persistència amb Hibernate hem fet servir un tipus especial de semàntica a la programació Java anomenat @Annotations. Aquest tipus d'anotacions ens permeten indicar paràmetres des de fora dels mètodes i de les classes, evitant així carregar massa el codi i podent accedir de forma directa a tot tipus d'informació.

Hibernate treballa amb la persistència d'objectes mitjançant un mapatge d'un objecte en Java i relacionant-lo amb una taula de la BDD. Aquest mapatge es pot definir de dues formes: Amb una estructura XML o amb anotacions. En essència, amb ambdues solucions tindrem els mateixos resultats, l'únic que canvia és la sintaxis d'una i altra solució.

Framework Struts

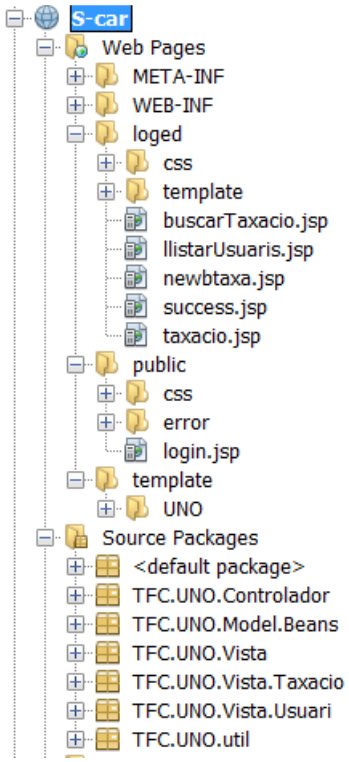
Com hem dit anteriorment, Struts és un Framework per a la creació d'aplicacions basades en la web (concretament servlets i pàgines JSP). En el nostre cas hem fet servir la versió 2.2.1.1 de Struts2. Encara que en essència Struts i Struts2 són iguals, a l'hora de treballar amb una versió o una altra es fa de forma molt diferent.

Struts es fa servir per la capa de Vista (l'aparença del producte final) i controla el flux del programa amb 'Interceptors' i 'Actions'. La seva configuració es troba a 'S-car\WEB-INF\classes' encara que a priori no és necessària cap modificació d'aquest arxiu.

Al mateix temps que Struts es fa servir un altre Framework completament integrat amb el primer, anomenat Tiles, per a la homogeneïtat de les pàgines JSP. Tiles ens proporciona la creació de plantilles estructurades per donar a la nostra aplicació una aparença constant.

ESTRUCTURA DEL PROJECTE

A continuació mostrarem la estructura general del projecte i explicarem en detall les diferents parts:



- A 'Web Pages' trobem tots els arxius i carpetes relacionats amb la configuració i vista de la nostra aplicació. Les carpetes 'META-INF' i 'WEB-INF' són predeterminades per fer-les servir i guardar la informació amb els servidors Tomcat. Respecte a l'aplicació en si, podem veure que s'han dividit les pàgines JSP en dues branques, una anomenada 'loged' i l'altra anomenada 'public'. A la carpeta 'loged' tenim un estil personalitzat de les pàgines (CSS) i una carpeta anomenada 'template', on es guarda la estructura utilitzada pel Framework Tiles. A la carpeta 'public' únicament tenim la pàgina 'login.jsp' i una carpeta 'error' per guardar els missatges d'error de servidor personalitzats.

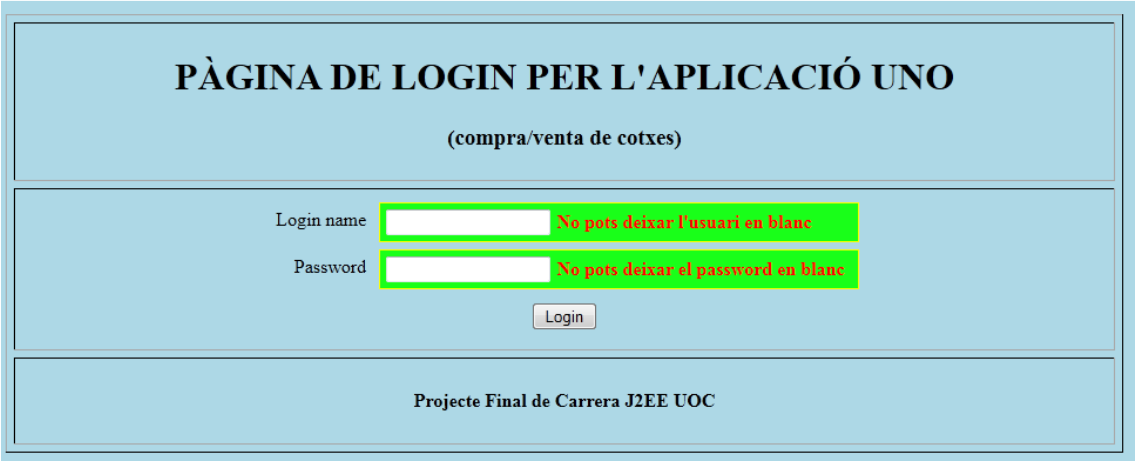
- A 'Source packages' trobem el codi font de la aplicació. Està dividit en tres branques; La primera, referent als controladors, on trobem tot el codi (implementacions i interfícies) per comunicar-se amb la BDD. La segona, referent a les vistes de l'aplicació, on es troba el codi (actions/interceptors/validates) referent a Struts i que fan servir de forma directa les pàgines JSP. La tercera branca, referent al model de negoci, on es declaren tots el beans (objectes) de l'aplicació.

NIVELL D'IMPLANTACIÓ

A continuació analitzarem el nivell d'implantació dels diferents casos d'ús descrits anteriorment amb prioritat alta.

Identificació

S'ha descartat la utilització dels anomenats 'Realms' de Tomcat per a la gestió d'usuaris i accessos, implementant una solució de forma manual. S'han fet servir les prestacions de la BDD Mysql per a l'emmagatzematge de contrasenyes mitjançant cadenes BLOB[5]. Existeix un interceptor de Login per evitar que algú pugui accedir a una pàgina sense estar identificat al sistema i un segon interceptor implementat en totes les accions disponibles per comprovar els permisos d'un usuari, permetent així la restricció d'usuaris identificats al sistema però que no tenen els suficients permisos per fer servir una acció en concret.

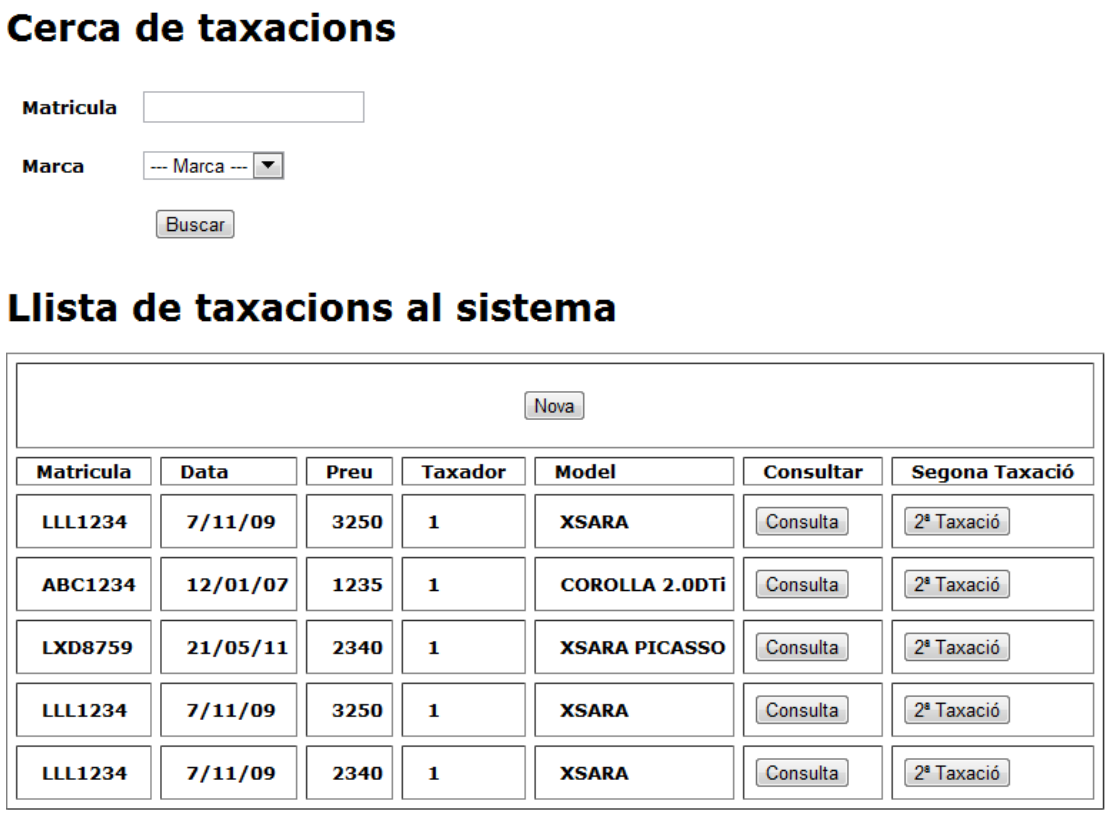


Pàgina de login

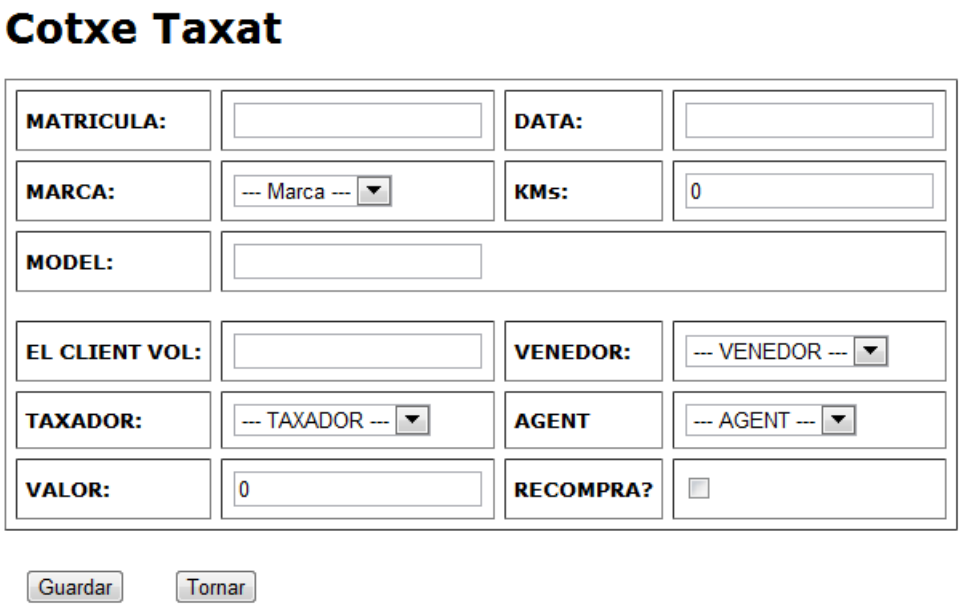
[5] La solució BLOB fa servir una cadena de caràcters com a clau per encriptar les contrasenyes. Aquesta clau és necessària tant per encriptar (es fa servir la funció de xifrat AES_ENCRYPT) com desencriptar (es fa servir la funció de xifrat AES_DECRYPT) les contrasenyes.

Taxacions (CRUD, 2ª Taxació, filtres)

Totes les funcions referents a les taxacions estan completament implementades. La funcionalitat CRUD (Create, Read, Update, Delete) està completa i funcionant de forma correcta (La funció d'eliminar taxacions no s'implantarà al sistema a priori). Es pot crear una segona taxació a partir d'una taxació anterior (evitant així introduir dades repetitives com podria ser la matrícula o marca/model de cotxe). A més es pot filtrar la llista de taxacions segons la 'Marca' o la 'Matrícula' (En aquest segon cas, no es comprova una concurrència exacta, sinó parcial; Per exemple, es podria introduir el número 1234 per trobar totes les matrícules coincidents amb aquests dígits, tinguin la lletra que tinguin; Es fa servir la sentència '%LIKE%') A més, aquests filtres són acumulatius.



Llistat de taxacions (Es pot fer un filtre de la llista segons Matrícula/Marca, crear una de nova, consultar una existent o crear una 2ª Taxació a partir d'una existent)



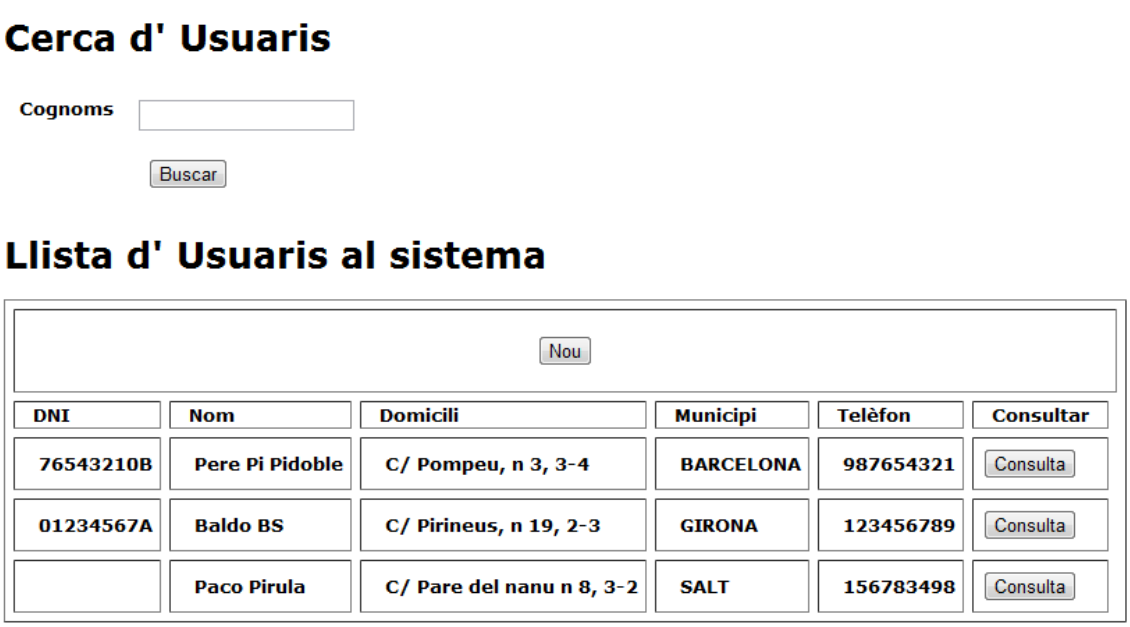
Pàgina per a la consulta/modificació de taxacions existents al sistema.

Usuaris (CRUD, filtres)

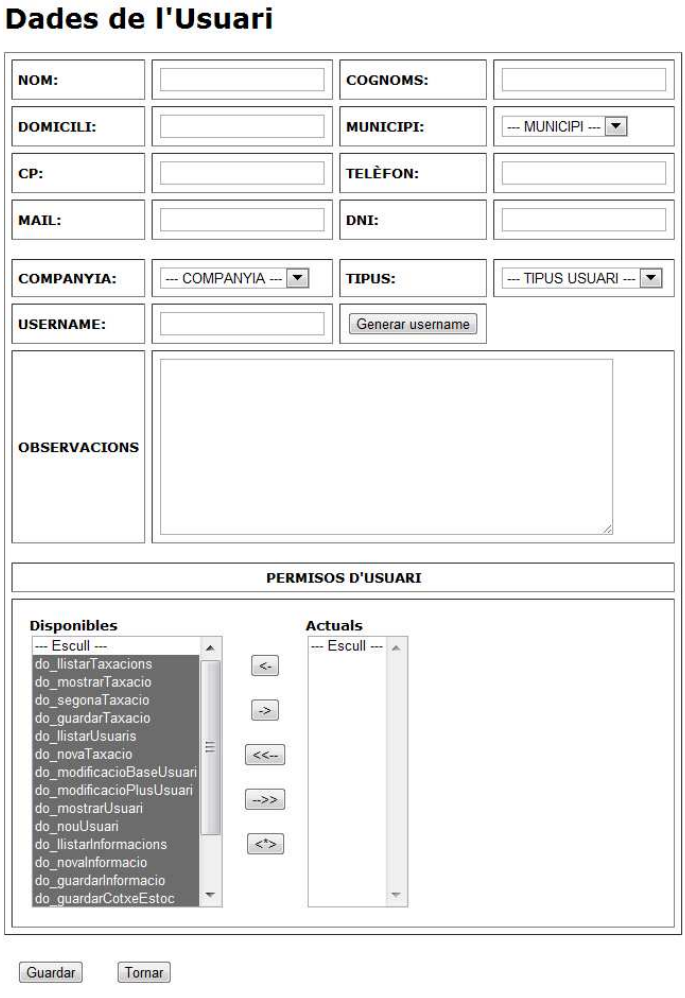
Totes les funcionalitats referents als usuaris estan implementades en el sistema. Existeixen dos nivells de modificació d'un usuari. Un primer nivell on es poden modificar únicament un número de dades determinades, i un segon nivell on es pot modificar qualsevol dada d'un usuari, excepte la seva contrasenya. Actualment no està disponible el registre de nous usuaris pròpiament dits al sistema, per la qual cosa no és possible modificar la contrasenya.

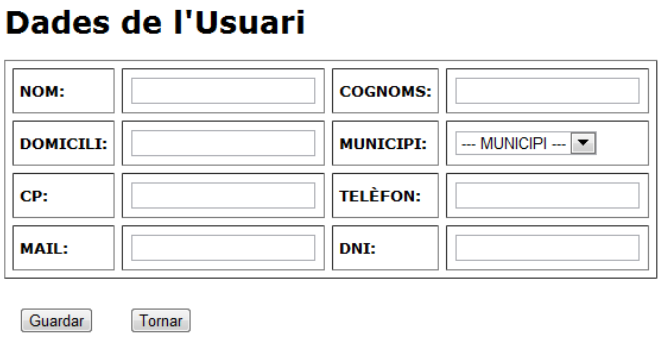
Hem de tenir en compte que a nivell lògic de la base de dades tothom és 'usuari', ja sigui administrador, venedor, taxador o client, però únicament els administradors, venedors i taxadors poden fer ús del sistema. És per això que no és necessària la modificació de contrasenyes ja que suposem que tots els usuaris afegits al sistema seran clients, sense cap accés a la aplicació.

A banda de tot això també tenim disponible el llistat de tots els usuaris existents al sistema i un filtre capaç de seleccionar únicament els usuaris amb un cognom donat, fent servir la sentència LIKE%.



Llistat d'usuaris (Es pot fer un filtre de la llista segons els cognoms, crear un nou usuari o consultar/modificar una existent)





Modificació d'un usuari de primer nivell

Modificació d'un usuari de segon nivell

Cotxes en Estoc (CRUD, filtres)

Aquesta funcionalitat ens permet crear, modificar i consultar qualsevol cotxe existent a la nostra base de dades que tinguem en estoc i que està (o estarà en un curt període de temps) disponible per a la venta de cara al client final. Les seves opcions són les mateixes implementades en funcionalitats anteriors, com el llistat complet de cotxes i la possibilitat de filtrar-los segons el seu tipus d'estoc (Si estan actualment en estoc, si estan en espera per afegir-los a l'estoc o si estan en tràmits de vendre'l per exemple) o la modificació o creació de nous cotxes al sistema.

Cotxes en Estoc (CRUD, filtres)

Aquesta funcionalitat ens permet crear, modificar i consultar qualsevol cotxe existent a la nostra base de dades que tinguem en estoc i que està (o estarà en un curt període de temps) disponible per a la venta de cara al client final. Les seves opcions són les mateixes implementades en funcionalitats anteriors, com el llistat complet de cotxes i la possibilitat de filtrar-los segons el seu tipus d'estoc (Si estan actualment en estoc, si estan en espera per afegir-los a l'estoc o si estan en tràmits de vendre'l per exemple) o la modificació o creació de nous cotxes al sistema.

Informacions (CRUD, filtres)

Aquesta funcionalitat ens permet crear, modificar i consultar qualsevol informació existent a la nostra base de dades que hàgim presentat als clients. Les seves opcions són les mateixes exposades anteriorment, com el llistat complet d'informacions i la possibilitat de filtrar-los segons el cognom.

CONCLUSIONS

El treball amb la tecnologia J2EE i la utilització de *frameworks* com Hibernate, Struts o Tiles i la capacitat d'aquests per adaptar-se a qualsevol situació i requeriment necessari ens ha mostrat les infinites possibilitats de les que es pot disposar fent servir aquests elements.

El treball amb la tecnologia J2EE és un requeriment cada cop més estès en el si de les empreses que volen un programari client-servidor, dividit per capes i mòduls diferenciats i on la necessitat de compartir la informació des de qualsevol lloc i en qualsevol moment imperen sobre la resta d'opcions. La capacitat per utilitzar la xarxa d'Internet com a medi de treball és clau per qualsevol projecte d'aquest tipus.

La utilització de Hibernate per administrar la base de dades mitjançant JAVA ens mostra la multitud de possibilitats que es poden fer servir de forma pràctica i senzilla. L'adaptació per part d'aquest *framework* de les anotacions de JAVA ens mostra el grau d'adaptació que aquesta eina pot arribar a tenir, i estem segurs que per la nostra part només hem fet servir una petita part de les seves possibilitats.

El *framework* Struts per la seva part ha sigut capaç de diferenciar de forma clara i concisa totes les capes de la nostra lògica de negoci, fent així possible la capacitat de expandir o modificar les funcionalitats del nostre projecte sense variar massa l'estil emprat i sense posar en risc la implementació actual.

Tiles per la seva banda ens ha aportat una homogeneïtat en totes les pantalles del nostre projecte mitjançant una única plantilla, encara que es podrien haver fet servir moltes més combinacions. D'aquesta forma es poden dotar als projectes d'aparences completament iguals sense cap tipus de problema, o donar una aparença única segons les opcions escollides.

Personalment el treball amb aquesta tecnologia i els diferents *frameworks* ha sigut una experiència enriquidora que m'ha mostrat una part fins al moment desconeguda de la programació. El que més destacaria és la utilització de Hibernate i l'ús de les seves fent així possible la comunicació amb la base de dades d'una forma molt senzilla i transparent. Per contra, la major dificultat trobada a l'hora d'implementar el projecte ha sigut Struts. És evident que les seves possibilitats són moltes però el seu nivell d'aprenentatge és massa alt i el requeriment de temps és massa excessiu per poder tenir complet un projecte sense cap base en menys de quatre mesos.

ANNEXOS

Script de creació de la Base de Dades

```
CREATE TABLE IF NOT EXISTS Companyia (
  companyia VARCHAR(20) NOT NULL,
  PRIMARY KEY (companyia)
) DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS tipusUsuari (
  nom VARCHAR(15) NOT NULL,
  PRIMARY KEY (nom)
) DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Provincia (
  provincia VARCHAR(25) NOT NULL,
  PRIMARY KEY (provincia)
) DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Municipi (
  municipi VARCHAR(25) NOT NULL,
  provincia VARCHAR(25) NOT NULL,
  PRIMARY KEY (municipi),
  KEY (provincia)
) DEFAULT CHARSET=utf8;

ALTER TABLE Municipi
ADD CONSTRAINT Municipi_ibfk_1 FOREIGN KEY (provincia) REFERENCES
Provincia (provincia) on DELETE SET NULL ON UPDATE CASCADE;


CREATE TABLE IF NOT EXISTS Usuari (
  idUsuari INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  companyia VARCHAR(20) default NULL,
  municipi VARCHAR(25) NOT NULL,
  username VARCHAR(15) NOT NULL,
  password BLOB,
  nom VARCHAR(20) NOT NULL,
  tipus VARCHAR(15) NOT NULL default 'CLIENT',
  permisos INTEGER UNSIGNED NOT NULL default 0,
  cognoms VARCHAR(50) default NULL,
  domicili VARCHAR(50) default NULL,
  CP INTEGER UNSIGNED default NULL,
  telefon INTEGER UNSIGNED default NULL,
  mail VARCHAR(25) default NULL,
  observacions LONGTEXT default NULL,
  DNI VARCHAR(9) default NULL,
  PRIMARY KEY (idUsuari),
  KEY (tipus),
  KEY (municipi),
  KEY (companyia),
  UNIQUE KEY (username),
  KEY (nom, cognoms)
) DEFAULT CHARSET=utf8;
```

```
ALTER TABLE Usuari
ADD CONSTRAINT Usuari_ibfk_1 FOREIGN KEY (tipus) REFERENCES
tipusUsuari (nom) ON DELETE SET DEFAULT,
ADD CONSTRAINT Usuari_ibfk_2 FOREIGN KEY (municipi) REFERENCES
Municipi (municipi) ON DELETE SET NULL ON UPDATE CASCADE,
ADD CONSTRAINT Usuari_ibfk_3 FOREIGN KEY (companyia) REFERENCES
Companyia (companyia) ON DELETE SET NULL ON UPDATE CASCADE;


CREATE TABLE IF NOT EXISTS Color (
  color VARCHAR(20) NOT NULL,
  PRIMARY KEY (color)
) DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Marca (
 marca VARCHAR(20) NOT NULL,
 PRIMARY KEY (marca)
) DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS tipusEstoc (
  nom VARCHAR(15) NOT NULL,
  PRIMARY KEY (nom)
) DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Cotxe (
  idCotxe INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  color VARCHAR(20) NOT NULL,
  marca VARCHAR(20) NOT NULL,
  matricula VARCHAR(9) NOT NULL,
  model VARCHAR(50) NOT NULL,
  any_compra DATE NOT NULL,
  KMs INTEGER UNSIGNED NOT NULL default 0,
  preu INTEGER UNSIGNED NOT NULL default 0,
  extres INTEGER UNSIGNED NOT NULL default 0,
  estoc VARCHAR(15) NOT NULL default 'NO_ESTOC',
  recompra BOOL NOT NULL default FALSE,
  PRIMARY KEY (idCotxe),
  KEY (marca),
  KEY (color),
  KEY (matricula)
) DEFAULT CHARSET=utf8;

ALTER TABLE Cotxe
ADD CONSTRAINT Cotxe_ibfk_1 FOREIGN KEY (marca) REFERENCES Marca
(marca) ON DELETE SET NULL ON UPDATE CASCADE,
ADD CONSTRAINT Cotxe_ibfk_2 FOREIGN KEY (color) REFERENCES Color
(color) ON DELETE SET NULL ON UPDATE CASCADE,
ADD CONSTRAINT Cotxe_ibfk_3 FOREIGN KEY (estoc) REFERENCES tipusEstoc
(nom) ON DELETE SET DEFAULT;
```

```
CREATE TABLE IF NOT EXISTS CotxeClient (
  idUsuari INTEGER UNSIGNED NOT NULL,
  idCotxe INTEGER UNSIGNED NOT NULL,
  data DATE NOT NULL,
  PRIMARY KEY (idUsuari, idCotxe),
  KEY (idUsuari),
  KEY (idCotxe)
) DEFAULT CHARSET=utf8;

ALTER TABLE CotxeClient
ADD CONSTRAINT CotxeClient_ibfk_1 FOREIGN KEY (idUsuari) REFERENCES
idUsuari (Usuari) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT CotxeClient_ibfk_2 FOREIGN KEY (idCotxe) REFERENCES
idCotxe (Cotxe) ON DELETE CASCADE ON UPDATE CASCADE;


CREATE TABLE IF NOT EXISTS Agent (
  agent VARCHAR(25) NOT NULL,
  PRIMARY KEY (agent)
) DEFAULT CHARSET=utf8;


CREATE TABLE IF NOT EXISTS Taxacio (
  idTaxacio INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  marca VARCHAR(20) NOT NULL,
  matricula VARCHAR(9) NOT NULL,
  model VARCHAR(50) NOT NULL,
  KMs INTEGER UNSIGNED NOT NULL default 0,
  venedor INTEGER UNSIGNED NOT NULL,
  taxador INTEGER UNSIGNED NOT NULL,
  agent VARCHAR(25) default NULL,
  data DATE NOT NULL,
  client_vol VARCHAR(50) NOT NULL,
  recompra BOOL NOT NULL default FALSE,
  valor INTEGER UNSIGNED NOT NULL default 0,
  PRIMARY KEY (idTaxacio),
  KEY (agent),
  KEY (taxador),
  KEY (venedor)
);

ALTER TABLE Taxacio
ADD CONSTRAINT Taxacio_ibfk_1 FOREIGN KEY (agent) REFERENCES agent
(Agent) ON DELETE SET NULL ON UPDATE CASCADE,
ADD CONSTRAINT Taxacio_ibfk_2 FOREIGN KEY (taxador) REFERENCES
idUsuari (Usuari) ON DELETE SET NULL ON UPDATE CASCADE,
ADD CONSTRAINT Taxacio_ibfk_3 FOREIGN KEY (venedor) REFERENCES
idUsuari (Usuari) ON DELETE SET NULL ON UPDATE CASCADE;
```

```
CREATE TABLE IF NOT EXISTS Oferta (
  idOferta INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  cotxe INTEGER UNSIGNED NOT NULL,
  recompra INTEGER UNSIGNED default 0,
  garantia INTEGER UNSIGNED default 0,
  varis VARCHAR(255) default NULL,
  PRIMARY KEY (idOferta),
  KEY (cotxe)
);

ALTER TABLE Oferta
ADD CONSTRAINT Oferta_ibfk_1 FOREIGN KEY (cotxe) REFERENCES idCotxe
(Cotxe) ON DELETE CASCADE ON UPDATE CASCADE;


CREATE TABLE Informacio (
  idInformacio INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  client INTEGER UNSIGNED NOT NULL,
  oferta1 INTEGER UNSIGNED NOT NULL,
  oferta2 INTEGER UNSIGNED default NULL,
  oferta3 INTEGER UNSIGNED default NULL,
  PRIMARY KEY(idInformacio),
  KEY (oferta1),
  KEY (oferta2),
  KEY (oferta3),
  KEY (client)
);

ALTER TABLE Informacio
ADD CONSTRAINT Informacio_ibfk_1 FOREIGN KEY (oferta1) REFERENCES
idOferta (Oferta) ON DELETE SET NULL ON UPDATE CASCADE,
ADD CONSTRAINT Informacio_ibfk_2 FOREIGN KEY (oferta2) REFERENCES
idOferta (Oferta) ON DELETE SET NULL ON UPDATE CASCADE,
ADD CONSTRAINT Informacio_ibfk_3 FOREIGN KEY (oferta3) REFERENCES
idOferta (Oferta) ON DELETE SET NULL ON UPDATE CASCADE,
ADD CONSTRAINT Informacio_ibfk_4 FOREIGN KEY (client) REFERENCES
idUsuari (Usuari) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE IF NOT EXISTS Permissions (
  idPermission INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  accio VARCHAR(25),
  base VARCHAR(15) NOT NULL default 'CLIENT',
  PRIMARY KEY(idPermission),
  UNIQUE KEY (accio)
) DEFAULT CHARSET=utf8 AUTO_INCREMENT=0;

DELIMITER |

CREATE TRIGGER assignId BEFORE INSERT on Permissions
FOR EACH ROW
BEGIN
DECLARE col INT;

SELECT MAX(idPermission) INTO col FROM Permissions;

IF col <> 0 THEN
SET NEW.idPermission = 2 * col;
ELSE
SET NEW.idPermission = 1;
END IF;

END
```

Les següents informacions s'han d'introduir obligatòriament al sistema abans de fer servir l'aplicació, així com a mínim un usuari per poder fer-lo servir amb un permís igual a l'id de *do_modificacioPlusUsuari*.

```
INSERT INTO Permissions (accio) VALUES("do_llistarTaxacions");
INSERT INTO Permissions (accio) VALUES("do_mostrarTaxacio");
INSERT INTO Permissions (accio) VALUES("do_segonaTaxacio");
INSERT INTO Permissions (accio) VALUES("do_guardarTaxacio");
INSERT INTO Permissions (accio) VALUES("do_llistarUsuaris");
INSERT INTO Permissions (accio) VALUES("do_novaTaxacio");
INSERT INTO Permissions (accio, base)
VALUES("do_modificacioBaseUsuari", "VENEDOR");
INSERT INTO Permissions (accio, base)
VALUES("do_modificacioPlusUsuari", "ADMINISTRADOR");
INSERT INTO Permissions (accio, base) VALUES("do_mostrarUsuari",
"VENEDOR");
INSERT INTO Permissions (accio, base) VALUES("do_nouUsuari",
"ADMINISTRADOR");
INSERT INTO Permissions (accio) VALUES("do_llistarInformacions");
INSERT INTO Permissions (accio) VALUES("do_novaInformacio");
INSERT INTO Permissions (accio) VALUES("do_guardarInformacio");
INSERT INTO Permissions (accio) VALUES("do_guardarCotxeEstoc");
INSERT INTO Permissions (accio) VALUES("do_llistarCotxeEstoc");
INSERT INTO Permissions (accio) VALUES("do_mostrarCotxeEstoc");
INSERT INTO Permissions (accio) VALUES("do_nouCotxeEstoc");
INSERT INTO Permissions (accio) VALUES("do_mostrarInformacio");

INSERT INTO tipusUsuari (nom) VALUES("CLIENT");
INSERT INTO tipusUsuari (nom) VALUES("VENEDOR");
INSERT INTO tipusUsuari (nom) VALUES("TAXADOR");
INSERT INTO tipusUsuari (nom) VALUES("ADMINISTRATIU");
INSERT INTO tipusUsuari (nom) VALUES("ADMINISTRADOR");
```

Instal·lació de l'aplicació

En aquesta secció volem donar una sèrie de pautes per a la instal·lació de l'aplicació. Ometrem la instal·lació del servidor Tomcat així com la Base de dades Mysql per la qual cosa si és necessària alguna indicació us remetem a que mireu la bibliografia al final del document on podreu trobar diferents direccions útils per a la seva instal·lació.

Tenint en compte que tenim els servidors de Tomcat i de la base de dades instal·lats de forma correcta, el primer pas serà executar el script per a la creació de la nostra base de dades, encara que prèviament haurem de crear un esquema anomenat TFC. Volem fer notar especialment que al final del script de creació de la base de dades hi ha un disparador per afegir permisos a la base de dades. La lògica de programació dels permisos es basa en els bits, per la qual cosa el seu valor numèric serà una potència de 2. Això s'ha de tenir molt en compte ja que sinó l'aplicació podria treballar de forma incorrecta i ens podríem trobar amb problemes de seguretat.

Un cop hem creat la base de dades el següent pas és emplenar-la. Podreu trobar un arxiu anomenat inserts.sql amb exemples per a la introducció de dades a la BDD però hem de tenir especial cura en la introducció de dades claus com són els permisos, que estan presents en la implementació del projecte. Si no existeixen aquests permisos a la base de dades la nostra aplicació no funcionarà de forma correcta. A més, és necessari introduir com a mínim un usuari al sistema amb permisos de modificació d'usuari per poder assignar-se tots els permisos possibles.

Finalment si hem seguit els passos anteriors podrem fer servir de forma completa la nostra aplicació.

BIBLIOGRAFIA

Material docent UOC, Benet Campderrich Falgueras. Enginyeria del Programari.

Pàgina oficial distribució Linux-Ubuntu
http://www.ubuntu.com/

Pàgina oficial Servidor i contenidor de Servlets Tomcat
http://tomcat.apache.org/

Instal·lació i configuració Tomcat a Linux-Ubuntu
http://joeljil.wordpress.com/2010/07/07/apache-tomcat-7-ubuntu-10-04/
http://albertoabian.wordpress.com/2010/07/20/instalacion-de-tomcat-7-en-gnulinux/

Pàgina oficial BDD Mysql
http://www.mysql.com/

Pàgina oficial patró MVC Struts
http://struts.apache.org/

Pàgina oficial i documentació model de persistència Hibernate
http://www.hibernate.org/docs

Pàgina oficial entorn de treball de programació Netbeans
http://netbeans.org/

Exemples Struts + Hibernate
http://www.roseindia.net/

GLOSSARI

- **API**: És un conjunt de funcions i procediments per a la utilització de tercers de forma estàndard, aconseguint així un nivell d'abstracció en el codi d'una aplicació.
- **Client**: Persona física, la qual en algun moment ha tramitat una comanda.
- **Comanda**: Acció per la qual un client es compromet a la compra d'un cotxe.
- **Estoc**: Conjunt de vehicles disponibles a l'empresa per vendre'ls.
- **Framework**: És un conjunt conceptes i regles per a la estandardització d'aplicacions, amb mètodes propis que serveixen per a la creació d'una estructura per a la resolució de problemes semblants.
- **Hivernate**: Framework per treballar amb la persistència de bases de dades.
- **Informació**: Llistat d'ofertes entregades a un futurible client.
- **JavaBeans**: API que defineix les línies principals d'utilització d'objectes per part del servidor.
- **MVC**: Patró les sigles del qual volen dir Model/Vista/Controlador. Basat en el Model v2 serveix per dissenyar una aplicació en capes, cadascuna és depenent de l'altra pel seu funcionament però al mateix temps cadascuna és independent de l'altre per a la seva formació.
- **Oferta**: Principi de comanda que s'estableix entre un venedor i un futurible client, amb condicions úniques sobre un cotxe en comú.
- **Recompra**: Acció per la qual un venedor accepta el cotxe d'un client al tramitar una comanda. (El client dóna el seu cotxe antic a canvi del nou)
- **Struts**: Framework per implantar el patró MVC.
- **Taxació**: Acció per la qual un taxador recull les dades d'un cotxe que pot interessar per fer una recompra, assignant-li un valor.
- **Taxador**: Persona física (sempre serà un venedor) que pot realitzar l'acció de taxació sobre un cotxe.
- **Tomcat**: Aplicació servidor que fa de contenidor pels nostres Servlets.
- **Venedor**: Persona física (no necessàriament és un taxador) que proporciona informació a futuribles clients i dóna d'alta els tràmits per realitzar comandes.