TFC – Bustamante.

Alumne: Àngel Bustamante Maldonado.
ETIG.

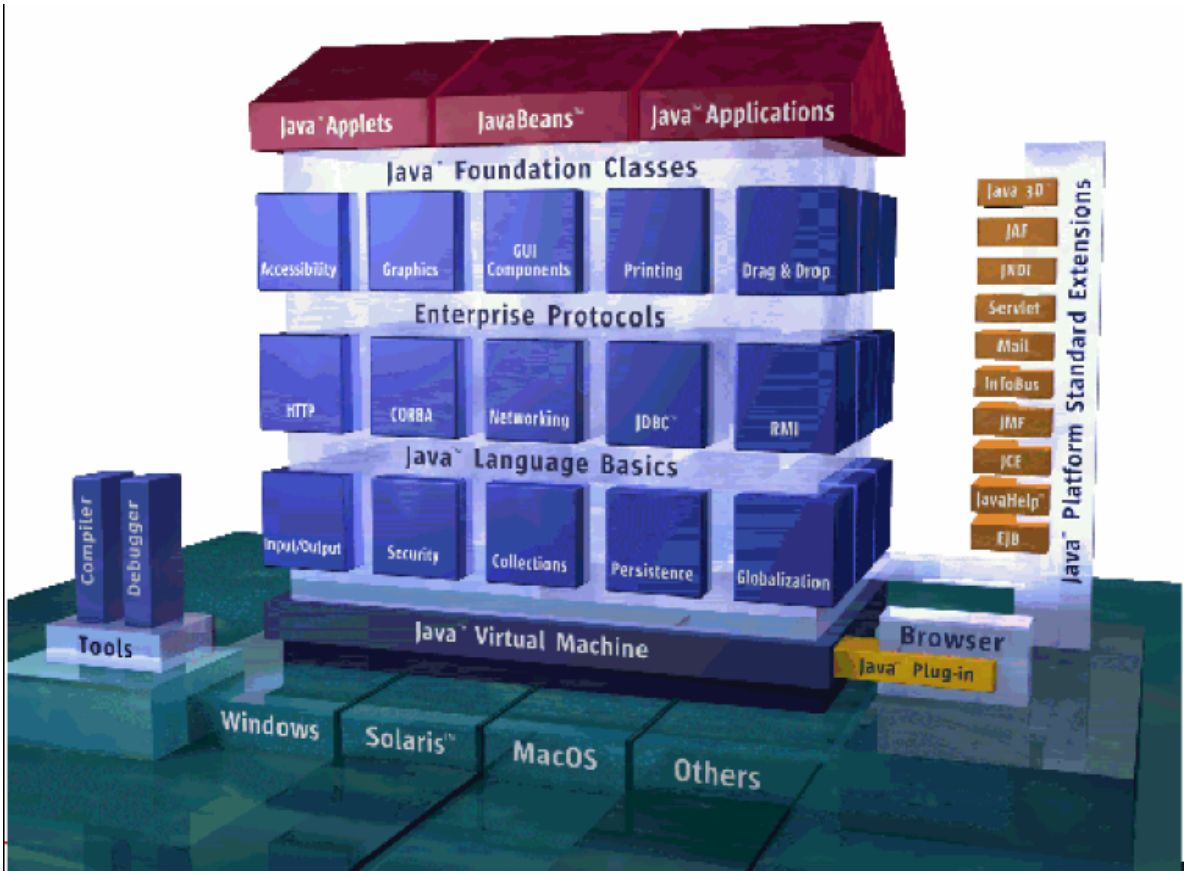
Consultor: Albert Grau Perisé

Data Lliurament: 25/06/2008.

1. Resum.

El treball fi de carrera (TFC) pretén ser un treball de sintesi dels coneixements adquirits en diferents assignatures de la carrera d'ETIG.
Entre les diferents àrees que es podien triar per a fer aquest treball, vaig triar la de J2EE per ser una tecnologia que s'està imposant en el món de les aplicacions empresarials.

Sun Microsystems va dissenyar un grup d'especificacions que permeten la creació d'aplicacions empresarials. Aquestes especificacions formen la plataforma Java 2 Enterprise Edition (J2EE), que defineix l'estàndar per al desenvolupament d'aplicacions multicapa a nivell empresa basades en components modulars i estandaritzats oferint un complert conjunt de serveis per als esmentats components i gestionant automàticament molts detalls del comportament de l'aplicació, sense una programació complexa. D'aquesta manera, facilita les tasques d'accés a base de dates (JDBC), a mètodes rèmots (RMI), la utilització de directoris distribuits (JNDI), aplicacions web (JSP, Servlets), etc...



Per aprofundir en el funcionament d'aquesta tecnologia, la utilitzem per a implementar una aplicació web que hem construit utilitzant la metodologia Rational Unified Process.

A més d'estudiar la tecnologia J2EE, aquest treball aplica molts coneixements adquirits en diferents assignatures com podrien ser Enginyeria del Programari, totes les de l'àrea de Bases de Dades, Programació Orientada a l'Objecte, Enginyeria del Programari, Tècniques de Desenvolupament del Programari, etc...

Considero doncs que el gran esforç que ha significat fer aquest treball m'ha permés de consolidar molts coneixements adquirits de la carrera a més d'incorporar-ne molts de nous.

2. Índex de continguts.

3. Cos de la memòria.

3.1 Introducció.

3.1.1 Justificació del TFC i context en el qual es desenvolupa: punt de partida i aportació del TFC.

Al començament del curs el consultor ens va animar a que proposessim nosaltres mateixos un problema que es pogués solucionar mitjançant una aplicació que fes servir la tecnologia J2EE. Tenint en compte la finalitat de sintesi i investigació del Treball Fi de Carrera, vaig decidir proposar una típica aplicació web empresarial d'un banc qualsevol amb les funcionalitats més habituals.
És important assenyalar que per a mi l'aplicació ha sigut un mitjà per coneixer el món J2EE, i no una finalitat en sí mateixa, per lo que m'he detingut en implementar unes poques funcionalitats incorporant això sí el màxim de conceptes i utilitats que es fan servir avui dia en aquest tipus d'aplicacions.
D'aquesta manera he hagut d'aprofundir en conceptes com Struts, Hibernate, Patrons, Servlets, JSP, HTML, Javascript, XML, Servidor d'Aplicacions, EJB, Oracle, Enginyeria del Programari, etc...

3.1.2 Objectius del TFC.

- Construcció d'una aplicació web a partir de les especificacions d'un client imaginari.

- Estudi i utilització del mètode Rational Unified Process, el més habitual actualment en la construcció de software.
- Disseny d'una base de dades i implementació del model lògic mitjançant un SGBD punter al mercat com Oracle.

- Estudi i utilització de patrons de disseny.

- Estudi i utilització del framework Struts.

- Estudi i utilització d'un Object Relational Mapping com és Hibernate.

3.1.3 Enfocament i mètode seguit.

Bàsicament es pot dividir la construcció d'aquest projecte en dues parts ben diferenciades: una primera centrada en l'estudi del mètode Rational Unified Process (RUP) a la vegada que l'anava aplicant al nostre cas concret, i una segona part centrada en l'estudi d'aspectes concrets de la tecnologia J2EE a la vegada que anava implementant l'aplicació. A la primera part se li ha d'afegir tota la feina de tria i prova del programari a utilitzar en la implementació.

3.1.4 Planificació del projecte.

Fase	N.Tasca	Data	Durada
Recollida i documentació de requisits			
Identificar casos d'us i actors.	1	15/03/2008	2
Detallar casos d'us.	2	17/03/2008	7
Interfaces d'usuaris	3	24/03/2008	7
Anàlisi.			
Anàlisi de l'arquitectura	4	31/03/2008	2
Anàlisi casos d'us.	5	03/04/2008	3
Anàlisi de classes.	6	07/04/2008	7
Anàlisi de paquets.	7	14/04/2008	2
Disseny			
Disseny d'arquitectura.	8	16/04/2008	5
Disseny casos d'us.	9	21/04/2008	7
Disseny de classes.	10	28/04/2008	7
Disseny de la persistència.	11	05/05/2008	4
Implementació.	12	09/05/2008	34
Proves	13	12/06/2008	4
Solució incidències	14	16/06/2008	4
Memòria	15	20/06/2008	3
Presentació.	16	23/06/2008	2
Lliurament.		25/06/2008	

Com podem veure al quadre anterior la planificació del projecte es va fer dividint el temps del que disposavem en tasques d'una determinada durada (en dies de treball). He sigut molt més precís en tota l'etapa de RUP que en la d'implementació, ja que en la primera les diferents tasques estan més clarament definides que en la segona.
He descartat incloure cap diagrama de Gantt o similar al considerar que al tractar-se d'un treball individual no tenia massa sentit.

3.1.5 Productes obtinguts.

Com a productes obtinguts en la primera etapa tenim la documentació generada, fent servir bàsicament la notació UML.
En quant a la etapa d'implementació obtenim un codi per a la seva avaluació, un fitxer EAR que al ser desplegat desplegat mostra el funcionament dels casos d'us de Login i Clients, una guia de proves per a facilitar el testing de l' EAR, uns scripts sql per crear i poblar la base de dades, documentació explicativa de cada uns dels mòduls de codi i documentació explicativa de la instal.lació.

3.2 Recollida i documentació de requissits.

A continuació enunciem el problema que pretenem resoldre amb la nostra aplicació i especifiquem tots els casos d'us.

Descripció del problema. Primera aproximació.

Un important banc espanyol disposa en l'actualitat d'una aplicació informàtica per a les seves sucursals, que per haver-se fet malament des d'un començament no deixa de donar problemes. Ens encarreguen construir una aplicació web que substitueixi a aquesta aplicació, i implantar-la en una oficina pilot. L'aplicació bàsicament ha de permetre introduir nous clients amb totes les seves dades, obrir i tancar comptes corrents, fer reintegraments, contractar productes, etc, a més de permetre portar el control del personal i la contabilitat de la sucursal. Aquestes operacions s'introduiran mitjançant una interficie gràfica a les oficines que l'utilitzin. Ens demanen que hi hagin dos rols, un de director i l'altre d'empleat. El banc te un departament d'arquitectura tecnològica que dona unes directrius que s'han de complir.

Descripció del problema. Segona aproximació.

L'aplicació, haurà de tenir bàsicament tres blocs funcionals, un d'activitat bancària, un altre de gestió de personal i un darrer de contabilitat:

- Gestió bancària. L'aplicació ens haurà de permetre donar d'alta/baixa/consulta/modificar clients, operacions i productes. Dels clients ens interessarà tenir quantes més dades millor. Les operacions bàsiques són ingressar o reintegrar diners. Els productes principals són comptes corrents, tarjetes, prèstecs i dipòsits.
- Gestió de personal. L'aplicació haurà de permetre donar d'alta/baixa/consulta/modificar els empleats de la sucursal, així com controlar l'absentisme.
- Gestió contable. Al final del dia ha de quedar reflectit en el balanç tots els moviments contables de la sucursal. Cada empleat disposa d'una caixa (que obrirà i tancarà cada dia el director) amb una quantitat inicial en metàlic que al final del dia haurà de quadrar. Un cop quadrat s'incorporaran les seves dades a un balanç general controlat pel director.

A partir d'aquestes dades bàsiques i d'altres que inferirem podem identificar els actors i els casos d'ús de l'aplicació:

Identificar actors.

L'aplicació en una primera fase, només disposarà de dos actors:

- Director d'oficina: tindrà accés a totes les funcionalitas.
- Empleat: Només tindrà accés a les funcionalitats puraments bancàries.

Identificar casos d'us.

Per a la gestió bancària:

- CU1: Alta / Baixa / Consulta / Modifica client.
- CU2: Alta / Baixa/ Consulta/ Modifica compte bancari.
- CU3: Alta / Baixa/ Consulta/ Modifica tatjeta.
- CU4: Alta / Baixa/ Consulta /Modifica solicitud prèstec.
- CU5: Alta / Baixa / Consulta /Modifica contractació dipòsit.
- CU6: Ingressar diners.
- CU7: Reintegrar diners.

Per a la gestió de personal:

- CU8: Alta /Baixa/ Consulta/ Modifica empleat.
- CU9: Alta /Consulta/ Modifica absentisme.

Per a la gestio contable:

- CU10: Obrir caixa central balanç.
- CU11: Obrir caixes
- CU12: Enviar diners caixa.
- CU13: Retirar diners caixa.
- CU14: Tancar caixes.
- CU15: Tancar caixa central.
- CU16: Generar balanç.
- CU17: Tancar balanç.
- CU18: Consultar balanç.

Anem ara a detallar tots aquets casos d'us:

Detallar casos d'us.

CU1: Alta / Baixa / Consulta / Modifica client.

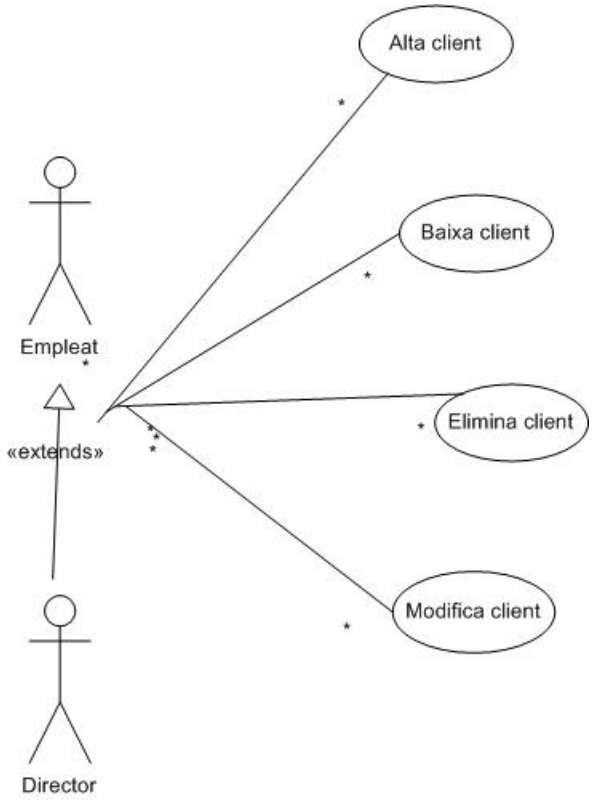
Resum funcionalitat general: introdueix, modifica, esborra i/o consulta un client a la base de dades.

Actors: director, empleat.

Precondició: el client no és a la base de dades en cas de creació. El client hi ha de ser per la resta de funcions.

Postcondició: el client està incorporat a la BBDD en cas de creació. En cas de modificació, eliminació i consulta, s'efectua l'operació o s'emet un missatge explicatiu de les raons per les quals no s'ha pogut portar a terme.

Descripció: en cas de creació, s'ha de introduir el dni, nom, cognoms, adreça, telèfon, ocupació del client. En la resta d'operacions s'ha de introduir el dni per a accedir al client.



CU2: Alta / Baixa/ Consulta/ Modifica compte bancari.

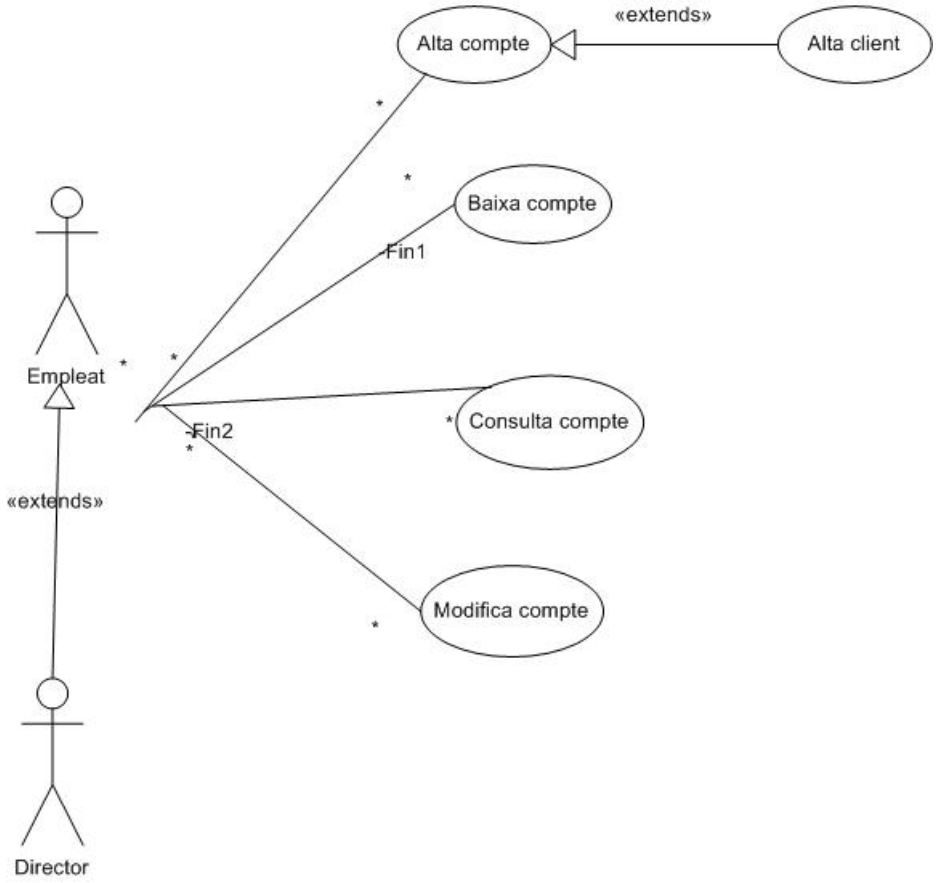
Resum funcionalitat general: introdueix, modifica, esborra i/o consulta un compte corrent a la base de dades. Sempre ha de anar associat a un client.

Actors: director, empleat.

Precondició: el compte no és a la base de dades en cas de creació.. El compte hi ha de ser per la resta de funcions a la base de dades.

Postcondició: el compte està incorporat a la BBDD en cas de creació. Aquest compte s'autogenera aleatoriament pel sistema excepte els 4 primers dígits que sempre són els mateixos per a identificar l'entitat bancària En cas de modificació, eliminació i consulta, s'efectua l'operació o s'emet un missatge explicatiu de les raons per les quals no s'ha pogut portar a terme.

Descripció: en cas de creació, s'ha de introduir el dni del client . En la resta d'operacions s'ha de introduir el dni o el número de compte per a accedir al compte. En la consulta de compte es poden veure els moviments entre dues dates.



CU3: Alta / Baixa/ Consulta/ Modifica tatjeta.

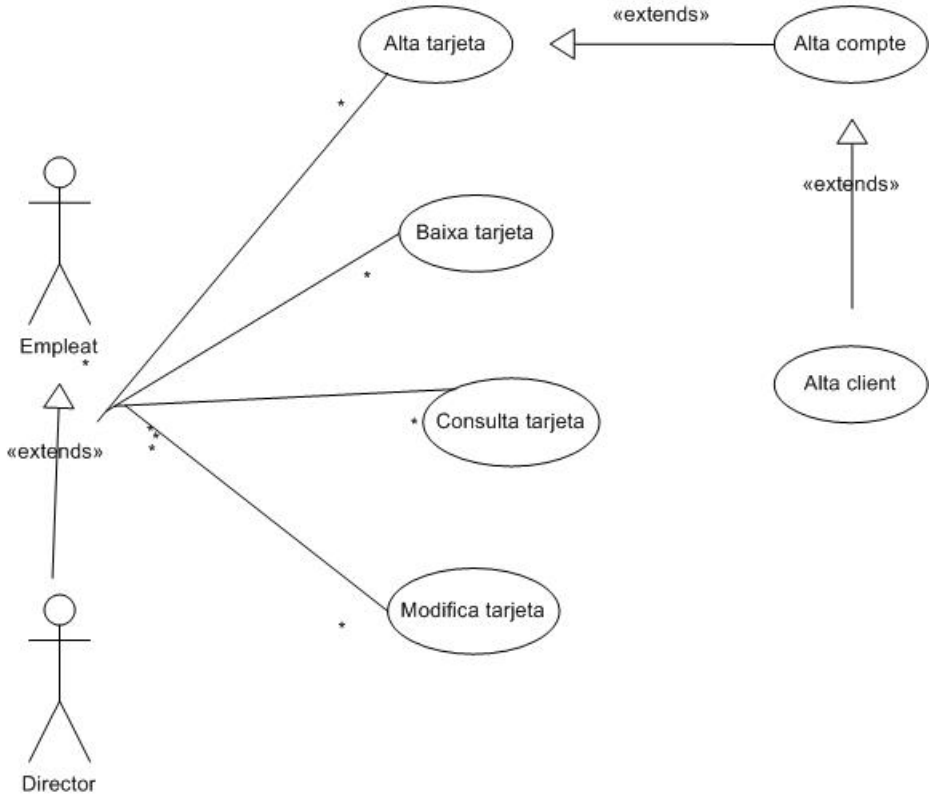
Resum funcionalitat general: dona d'alta, modifica, dona de baixa i/o consulta una tarjeta a la base de dades. Sempre ha de anar associada a un compte corrent i a un client.

Actors: director, empleat.

Precondició: la tarjeta no és a la base de dades en cas de creació. El client que la demana ha de tenir un compte. La tarjeta hi ha de ser per la resta de funcions a la base de dades.

Postcondició: la tarjeta està incorporada a la BBDD en cas de creació. En cas de modificació, eliminació i consulta, s'efectua l'operació o s'emet un missatge explicatiu de les raons per les quals no s'ha pogut portar a terme.

Descripció: en cas de creació, s'ha de introduir el dni del client . S'ha de triar un dels comptes si s'escau per associar la tarjeta. El número de la tarjeta es generarà automàticament. S'ha d'indicar el límit diari que es pot treure i si és de dèbit o crèdit, En la resta d'operacions s'ha de introduir el dni o el número de tarjeta per a accedir a al tarjeta. En el cas de consulta es poden veure els moviments entre dues dates.



CU4: Alta / Baixa/ Consulta /Modifica solicitud prèstec.

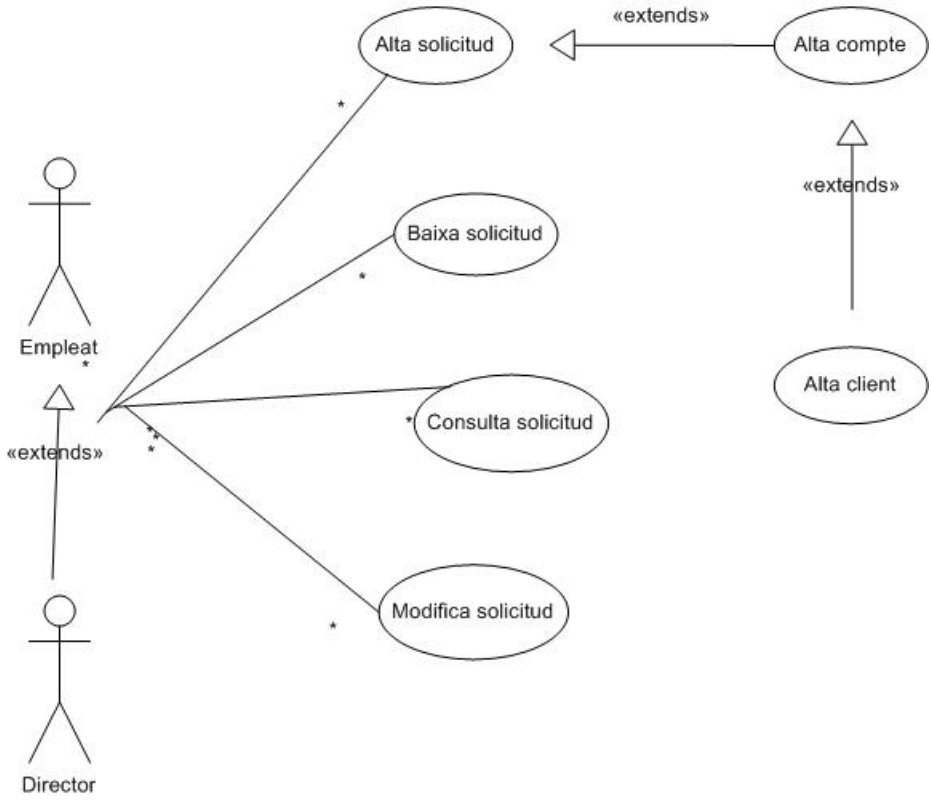
Resum funcionalitat general: dona d'alta, modifica, dona de baixa i/o consulta una solicitud de prèstec a la base de dades. Sempre ha de anar associada a un compte corrent i a un client.

Actors: director, empleat.

Precondició: la solicitud no és a la base de dades en cas de creació. El client que la demana ha de tenir un compte. La solicitud hi ha de ser per la resta de funcions a la base de dades.

Postcondició: la solicitud està incorporada a la BBDD en cas d'alta. En cas de modificació, eliminació i consulta, s'efectua l'operació o s'emet un missatge explicatiu de les raons per les quals no s'ha pogut portar a terme.

Descripció: en cas d'alta, s'ha de introduir el dni del client . S'ha de triar un dels comptes si s'escau per associar la solicitud. El número de solicitud es generarà automàticament. S'ha d'introduir la quantitat demanada i el plaç d'amortització desitjat. En la resta d'operacions s'ha de introduir el dni o el número de solicitud per a accedir a la solicitud.



CU5: Alta / Baixa / Consulta /Modifica contractació dipòsit.

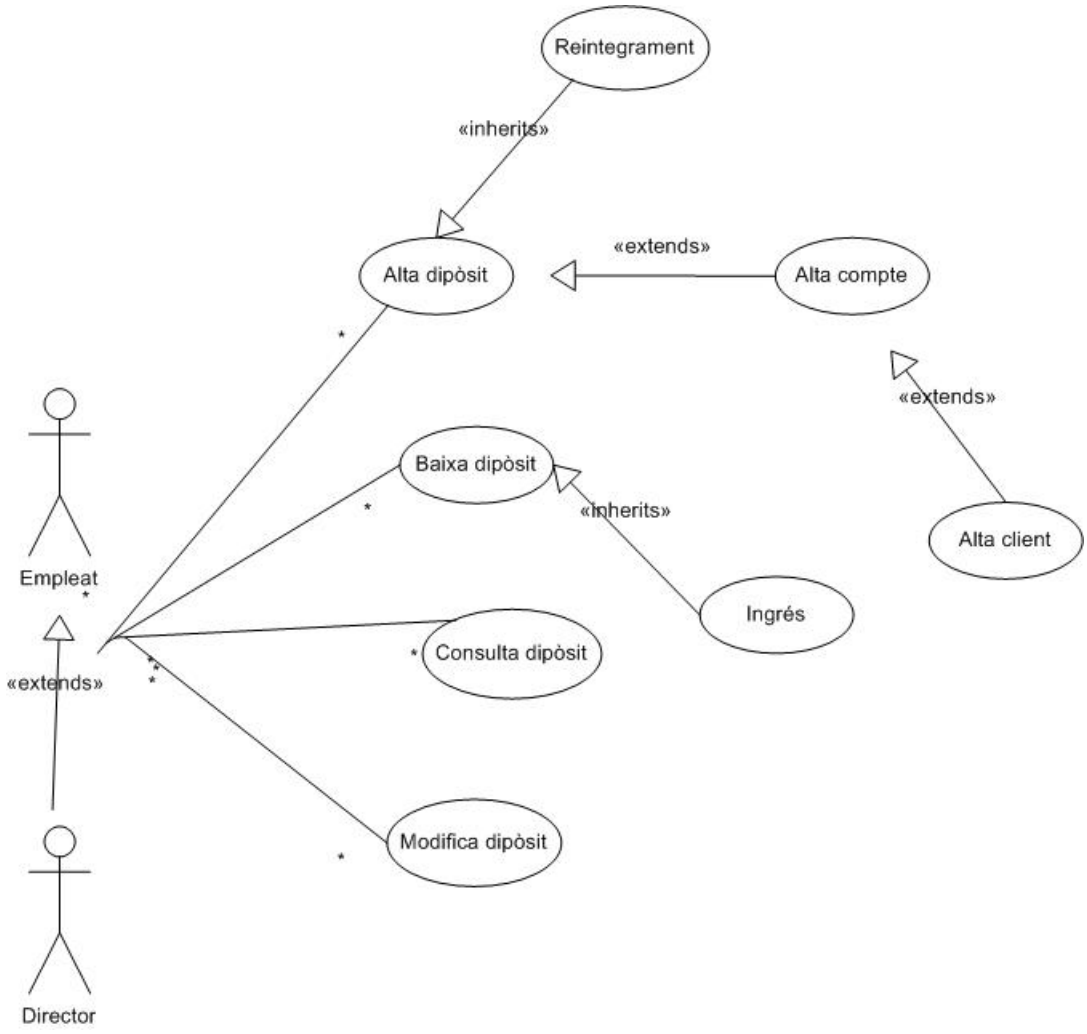
Resum funcionalitat general: dona d'alta, modifica, dona de baixa i/o consulta una contractació de dipòsit a la base de dades. Sempre ha de anar associada a un compte corrent i a un client.

Actors: director, empleat.

Precondició: la contractació no és a la base de dades en cas d'alta. El client que el contracta ha de tenir un compte. La contractació hi ha de ser per la resta de funcions a la base de dades.

Postcondició: la contractació està incorporada a la BBDD en cas d'alta. En cas de modificació, eliminació i consulta, s'efectua l'operació o s'emet un missatge explicatiu de les raons per les quals no s'ha pogut portar a terme.

Descripció: en cas d'alta, s'ha de introduir el dni del client . S'ha de triar un dels comptes per associar a la contractació. El número de contractació es generarà automàticament. S'ha d'introduir la quantitat dipositada, el plaç del dipòsit i tipus de dipòsit. El compte farà un reintegrament amb destinació el dipòsit. En la resta d'operacions s'ha de introduir el dni o el número de contractació per a accedir al dipòsit.
En cas de baixa el compte associat farà un ingrés provinent del dipòsit. La quantitat d'un dipòsit no es podrà modificar mai.



CU6: Ingressar diners.

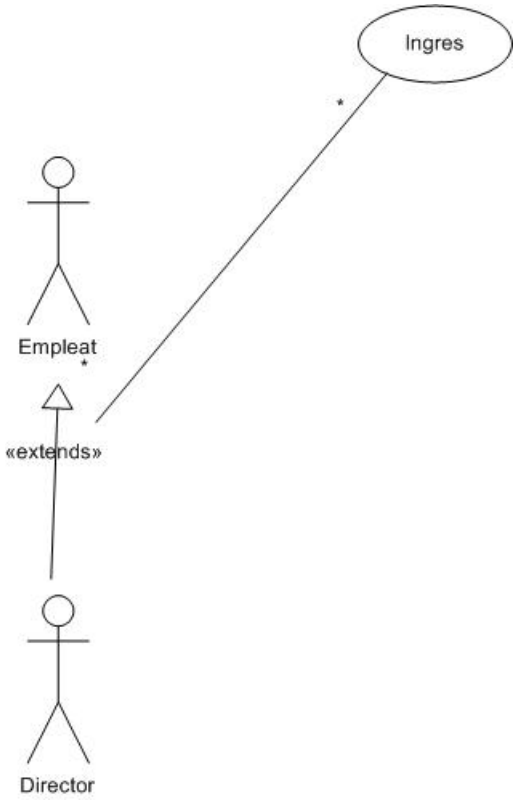
Resum funcionalitat general: ingressa una quantitat en un compte corrent.

Actors: director, empleat.

Precondició: l'ingrés no és a la base de dades.

Postcondició: l'ingrés està incorporat a la BBDD. Si el compte de destí no existeix dona error. Si el destinatari no coincideix amb el titular dona error. El saldo del compte s'incrementa pel valor de l'ingrés.

Descripció: Es pot especificar qui fa l'ingrés. Un ingrés pot ser una entrada en caixa en efectiu , venir de la baixa d'un dipòsit o d'un enviament de diners del director (en els dos darrers casos es fa automaticament).



CU7: Reintegrar diners.

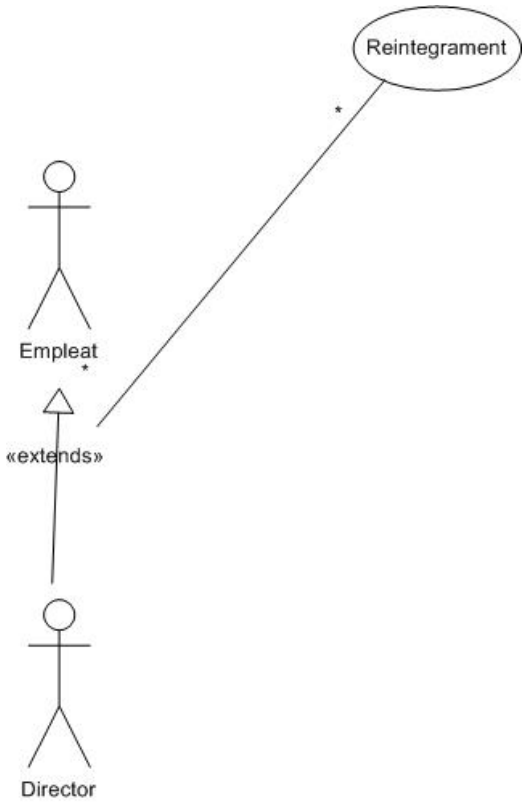
Resum funcionalitat general: reintegra una quantitat d'un compte corrent.

Actors: director, empleat.

Precondició: el reintegrament no és a la base de dades.

Postcondició: el reintegrament està incorporat a la BBDD. Si el compte d'orígen no existeix dona error. El saldo del compte pel valor del reintegrament.

Descripció: Un reintegrament pot ser una sortida de caixa en efectiu pel titular del compte, anar a la contractació d'un dipòsit a anar a una retirada de diners per part del director (en els dos darrers casos es fa automaticament).



CU8: Alta /Baixa/ Consulta/ Modifica empleat.

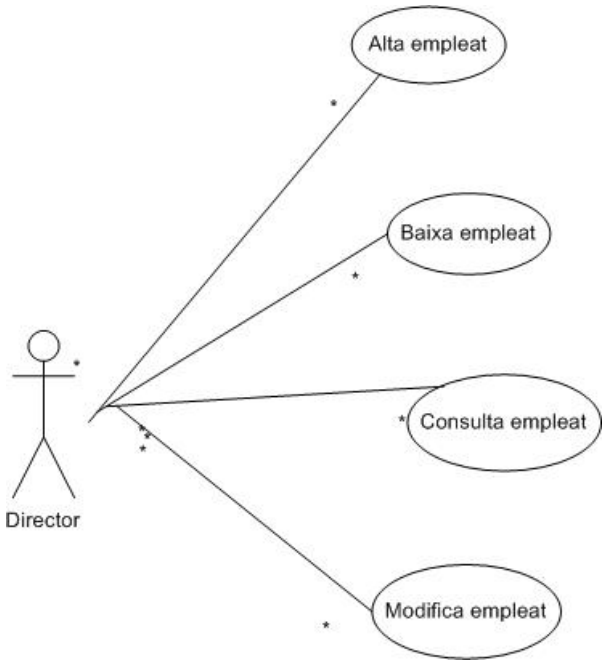
Resum funcionalitat general: introdueix, modifica, esborra i/o consulta un empleat a la base de dades.

Actors: director.

Precondició: l'empleat no és a la base de dades en cas d' alta. L'empleat hi ha de ser per la resta de funcions.

Postcondició: l'empleat està incorporat a la BBDD en cas d'alta. En cas de modificació, eliminació i consulta, s'efectua l'operació o s'emet un missatge explicatiu de les raons per les quals no s'ha pogut portar a terme.

Descripció: en cas de creació, s'ha de introduir el dni, nom, cognoms, adreça, telèfon, antiguetat a l'empresa, titulació acadèmica. En la resta d'operacions s'ha de introduir el dni per a accedir al client.



CU9: Alta /Consulta/ Modifica absentisme.

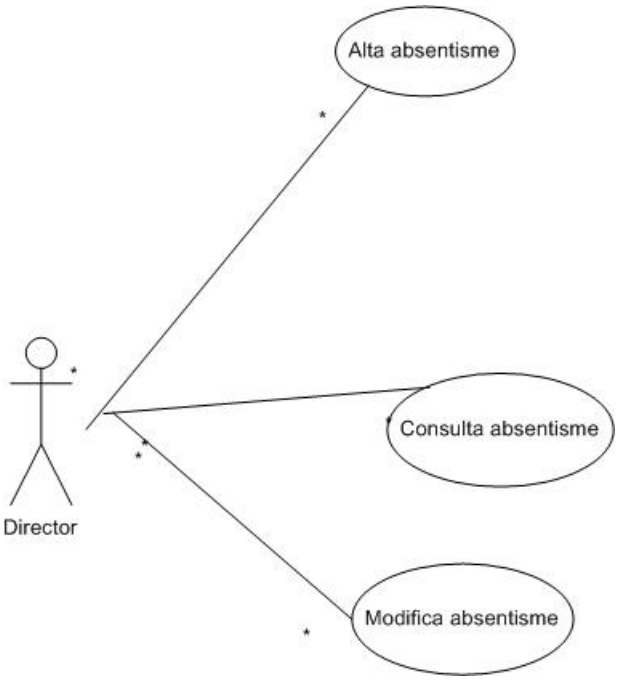
Resum funcionalitat general: introdueix, modifica, esborra i/o consulta l'absentisme corrent a la base de dades. Sempre ha de anar associat a un empleat o varis.

Actors: director.

Precondició: l'absentisme no és a la base de dades en cas d'alta. L'absentisme hi ha de ser per la resta de funcions a la base de dades.

Postcondició: L'absentisme està incorporat a la BBDD en cas d'alta. En cas de modificació, eliminació i consulta, s'efectua l'operació o s'emet un missatge explicatiu de les raons per les quals no s'ha pogut portar a terme.

Descripció:En alta l'absentisme consta d'una data d'inici i una altra de final. Si la final no se sap aquesta pot quedar en blanc fins que se sàpiga. S'ha de donar el dni de l'empleat que no hi és i la causa. En la resta d'operacions s'ha de introduir el dni o la data. Es pot consultar per rang de dates.



CU10: Obrir caixa central.

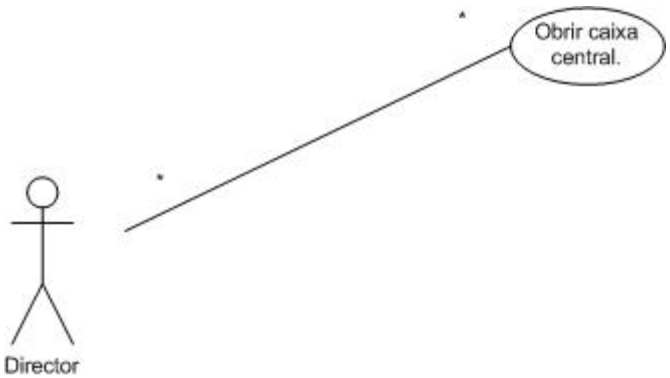
Resum funcionalitat general: obre la caixa central que tindrà la quantitat que indiqui el balanç del dia anterior.

Actors: director.

Precondició: el balanç del dia anterior està generat i tancat.

Postcondició: les caixa central del dia està oberta.

Descripció: Cada matí el director ha d'obrir la caixa central. Només es pot fer un cop al dia.



CU11: Obrir caixes

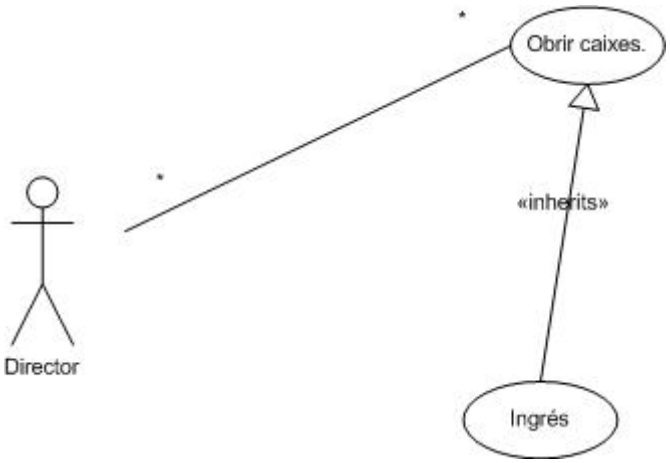
Resum funcionalitat general: obre la caixa assignada a cada empleat amb una quantitat inicial.

Actors: director.

Precondició: la caixa central es oberta. La caixa a obrir és tancada.

Postcondició: les caixes que el director consideri oportunes estan obertes amb una quantitat inicial provinent de la caixa central.

Descripció: Cada matí el director ha d'obrir les caixes dels empleats amb una quantitat inicial.



CU12: Enviar diners caixa.

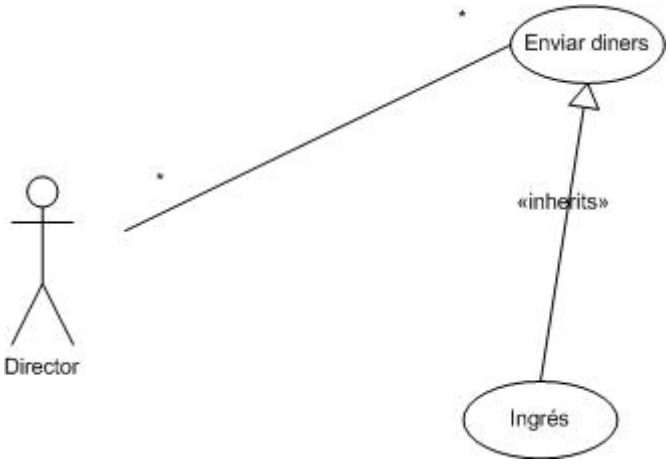
Resum funcionalitat general: si una caixa es queda sense efectiu el director fa una transferència desde la caixa central.

Actors: director.

Precondició: la caixa central disposa com a mínim de la quantitat a transferir.

Postcondició: la caixa de l'empleat augmenta el saldo la mateixa quantitat que disminueix el de la central.

Descripció: -



CU13: Retirar diners caixa.

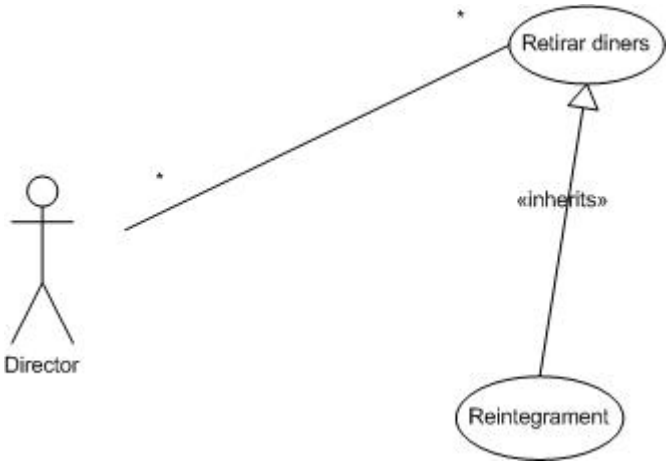
Resum funcionalitat general: si una caixa té massa efectiu el director fa una transferència cap a la caixa central per seguretat.

Actors: director.

Precondició: la caixa de l'empleat disposa com a mínim de la quantitat a transferir.

Postcondició: la caixa de l'empleat disminueix el saldo la mateixa quantitat que augmenta el de la central.

Descripció: -



CU14: Tancar caixes

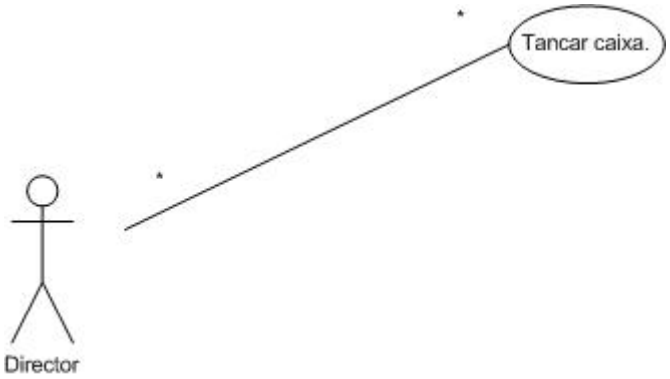
Resum funcionalitat general: en acabar la jornada tanca la caixa assignada a cada empleat.

Actors: director.

Precondició: la caixa central es oberta. La caixa a tancar és oberta.

Postcondició: la caixa és tancada.

Descripció: -



CU15: Tancar caixa central.

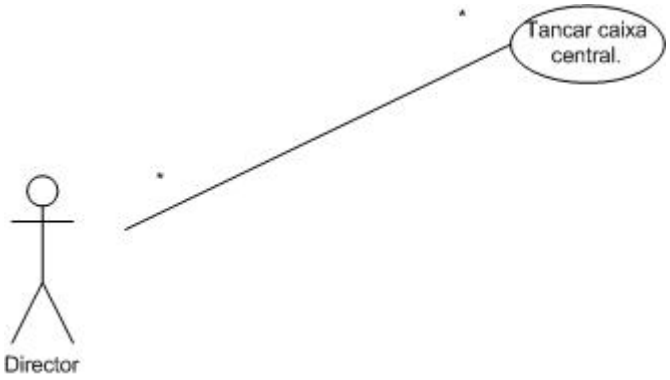
Resum funcionalitat general: tanca la caixa central.

Actors: director.

Precondició: totes les caixes dels empleat estan tancades. La caixa central és oberta.

Postcondició: les caixa central del dia està tancada.

Descripció: Al final de cada jornada el director ha de tancar la caixa central. Només es pot fer un cop al dia.



CU16: Generar balanç.

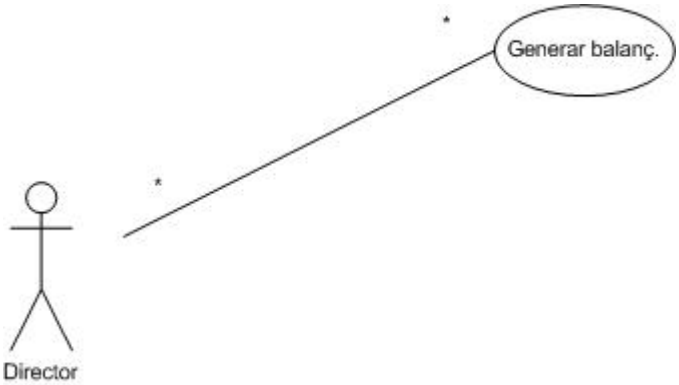
Resum funcionalitat general: en acabar la jornada s'integren les dades de totes les caixes en el balanç.

Actors: director.

Precondició: la caixa central es tancada.

Postcondició: S'ha generat el balanç provisional a l'espera d'entrar dades manualment.

Descripció: El balanç basicament consta d'entrades i sortides d'efectiu de les caixes dels empleats i d'entrades i sortides d'efectiu de la seu central a la sucursal. Les primeres es generen automaticament i les segones s'introdueixen manualment en tancar el balanç.



CU17: Tancar balanç.

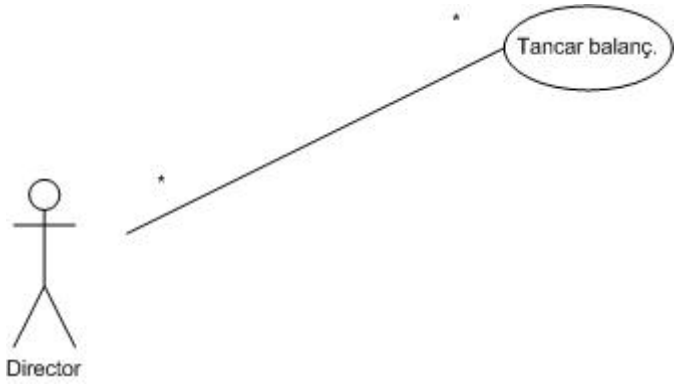
Resum funcionalitat general: Un cop generat el balanç s'han d'introduir els fons arribats i que surten de la sucursal cap a la seu central i tancar-lo amb la impossibilitat de modificar-lo.

Actors: director.

Precondició: el balanç s'ha generat.

Postcondició: El balanç del dia queda enregistrat a la base de dades.

Descripció: -



CU18: Consultar balanç.

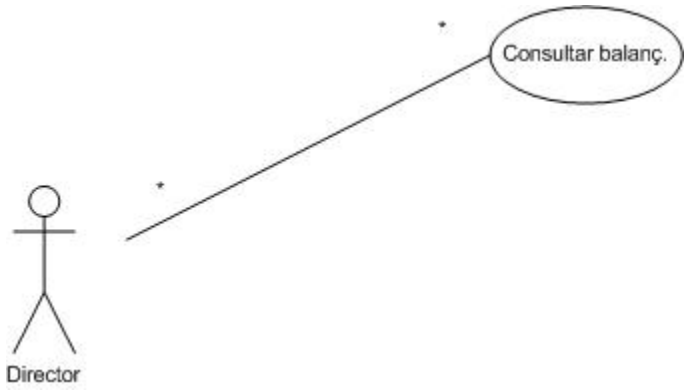
Resum funcionalitat general: Consultar el balanç d'un dia concret.

Actors: director.

Precondició: el balanç està tancat.

Postcondició: El balanç del dia dessitjat apareix per pantalla amb tots els conceptes.

Descripció: Els conceptes de les caixes són vincles que si es seleccionen detallen els moviments de cada caixa.



Veiem ara els diagrames de navegació general i particular per l'aplicació (les pantalles les posarem a l'apartat d'anexos).

Interfaces d'usuaris.

Diagrama de navegació general.

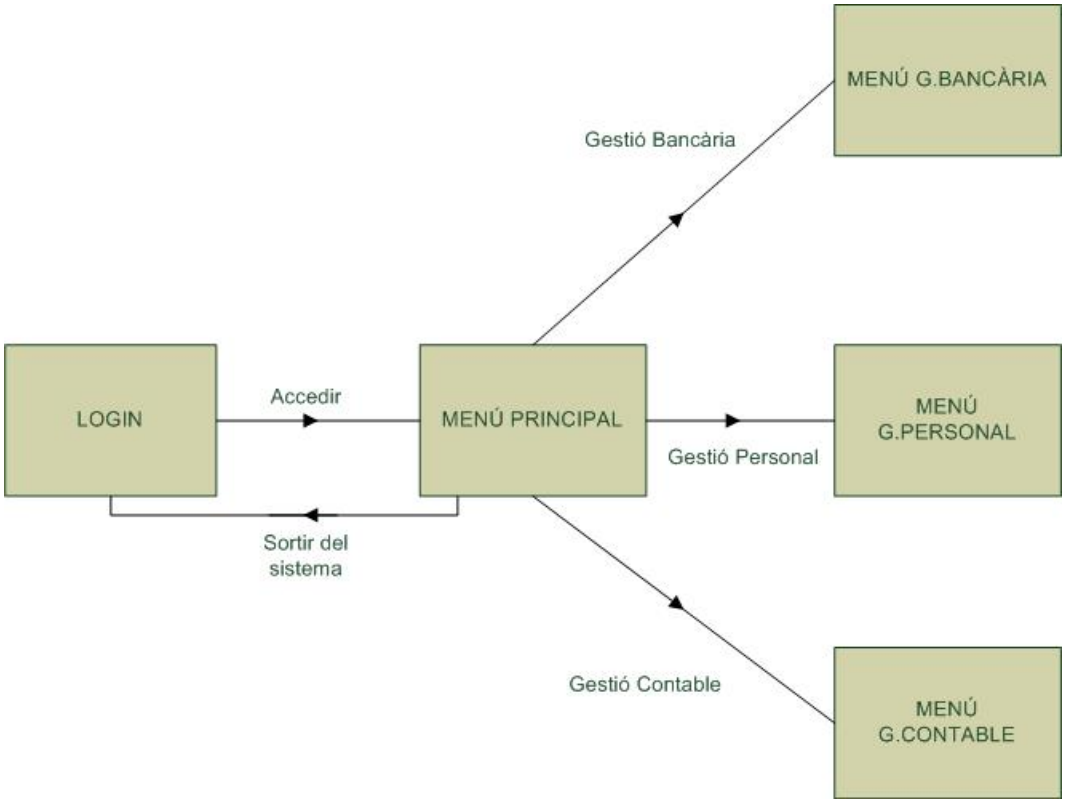
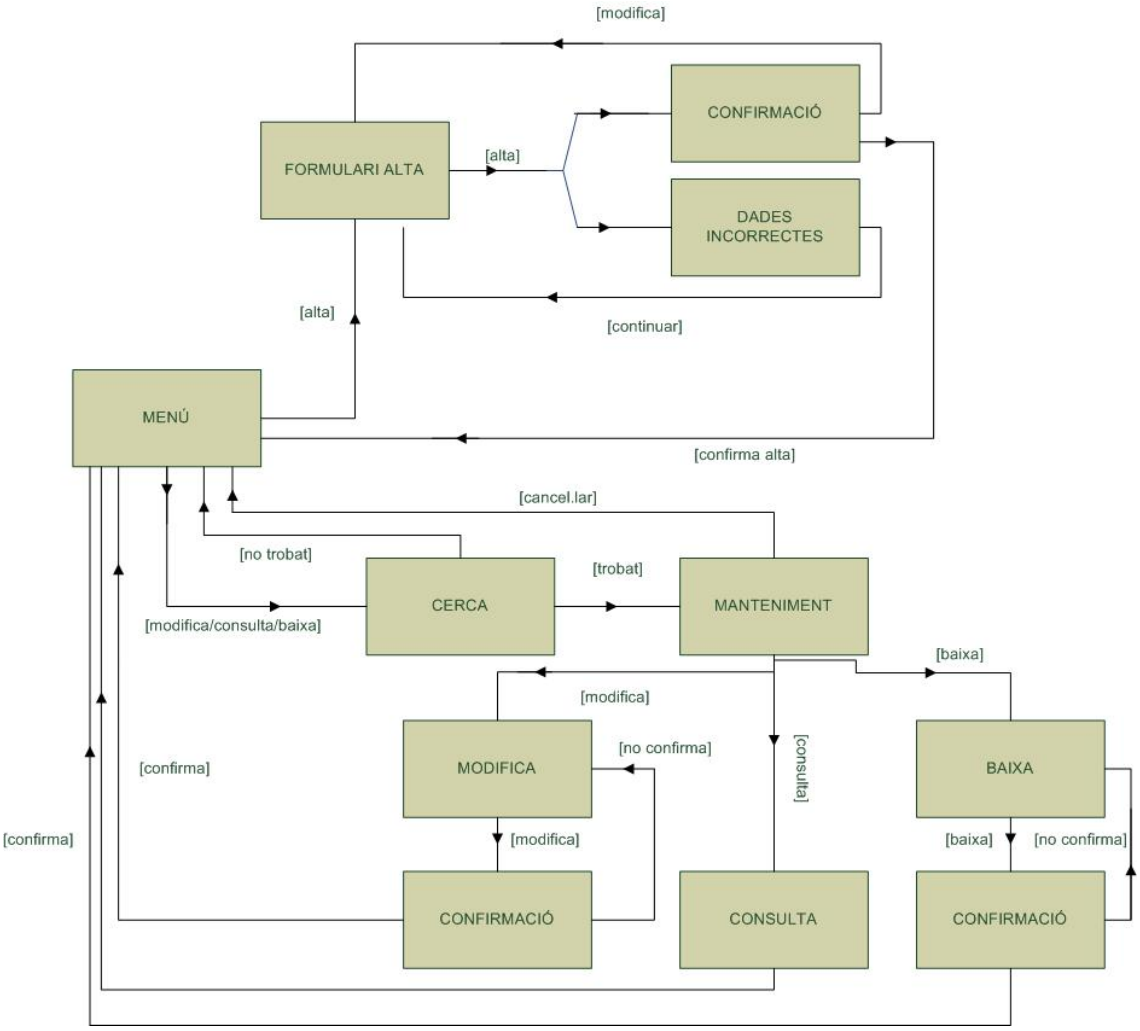


Diagrama de navegació particular.

Tots els menus segueixen aquest model:

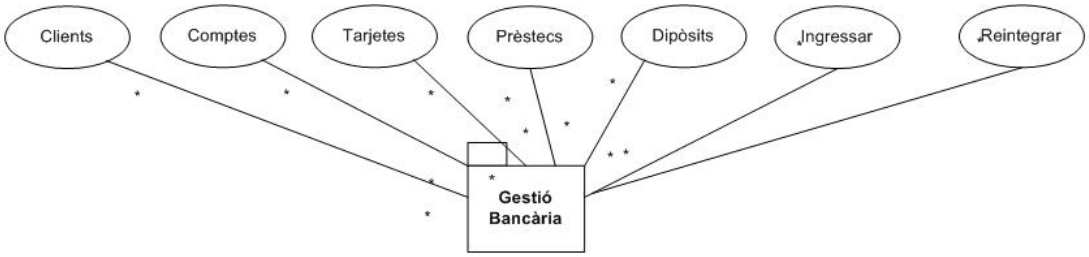


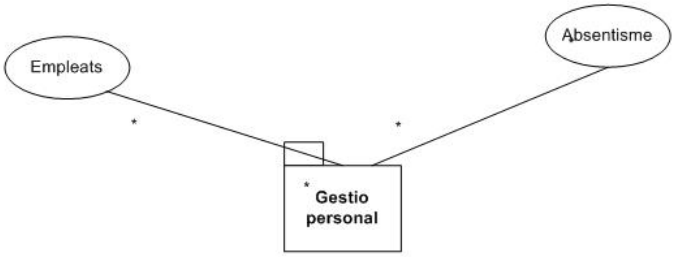
3.3 Anàlisi.

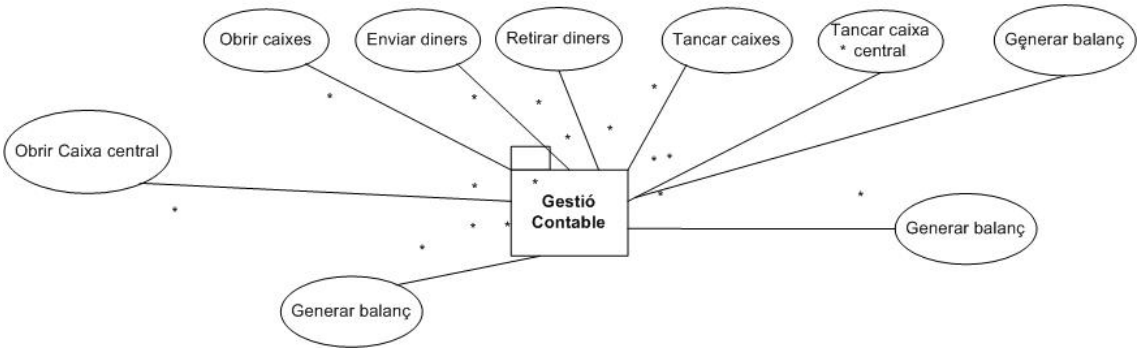
Ens ocupem ara en aquest apartat de l'etapa d'anàlisi. Dir que s'han identificat les classes presents en cada cas d'us i s'han construit els digrames de col.laboració corresponents als casos d'us de Login i Clients.

Anàlisi de l'arquitectura.

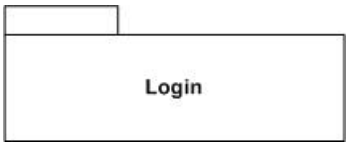
Identidicació de paquets d'anàlisi.





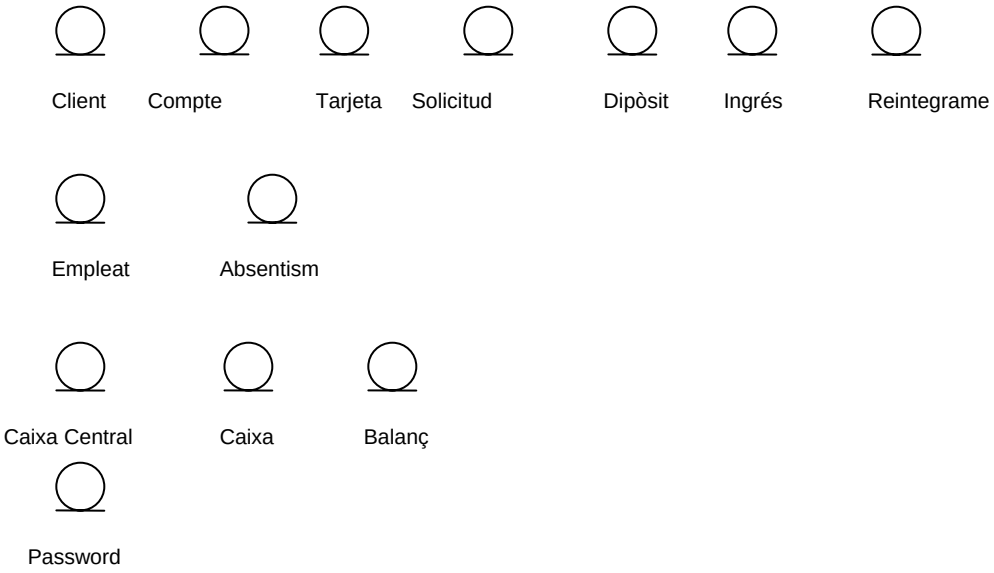


Identificació pàquets comuns.



Dependències entre paquets d'anàlisi.

Identificació de classes entitat



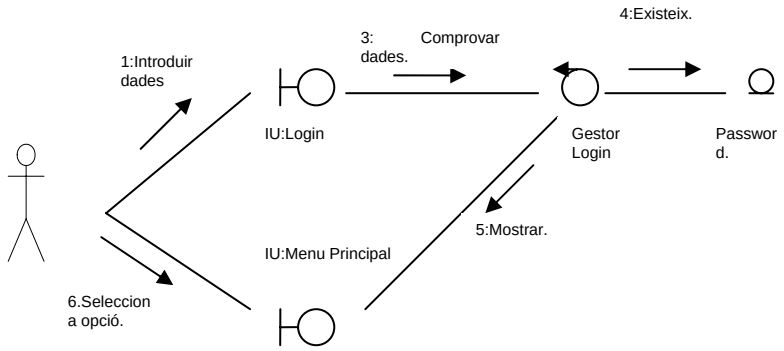
Anàlisi casos d'us.

Identificarem les classes que intervenen en cada cas d'us (Intentarem posar uns noms que siguin autoexplicatius) i dibuixarem els diagrames de col.laboració que mostri el fluxe d'envents entre elles.
Es divideixen en classes Frontera, que son l'interface amb l'usuari, de Control que realitzen les operacions, d'Entitat que representen objectes reals que han de ser persistents.
En tractar-se d'una aplicació J2EE totes les classes frontera són pàgines JSP.

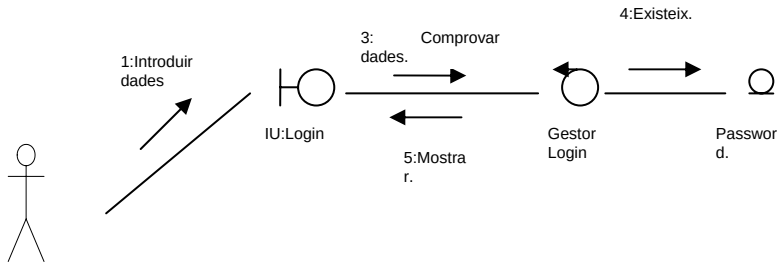
CU0. LOGIN

- Frontera:
  1) IULogin.
  2) IUMenuPrincipal.
- Control:
  1) GestorLogin.
- Entitat:
  1) Password.

Diagrama de col.laboració si l'usuari i la password introduides són correctes:



Si l'usuari i/o la password no ho són:



CU1. ALTA/BAIXA/CONSULTA/MODIFICA CLIENT.

- Frontera:
  1) IUMenuBancaria.
  2) IUMenuClients.
  3) IUAltaClientForm.
  4) IUCercaClient.
  5) IUModificaClientForm
  6) IUConfirmació.
  7) IUNotificació.
  8) IUError.

- Control:
  1) GestorClients.
- Entitat:
  1) Client.

Donar d'alta un client.

Diagrama de col.laboració si tot és correcte:

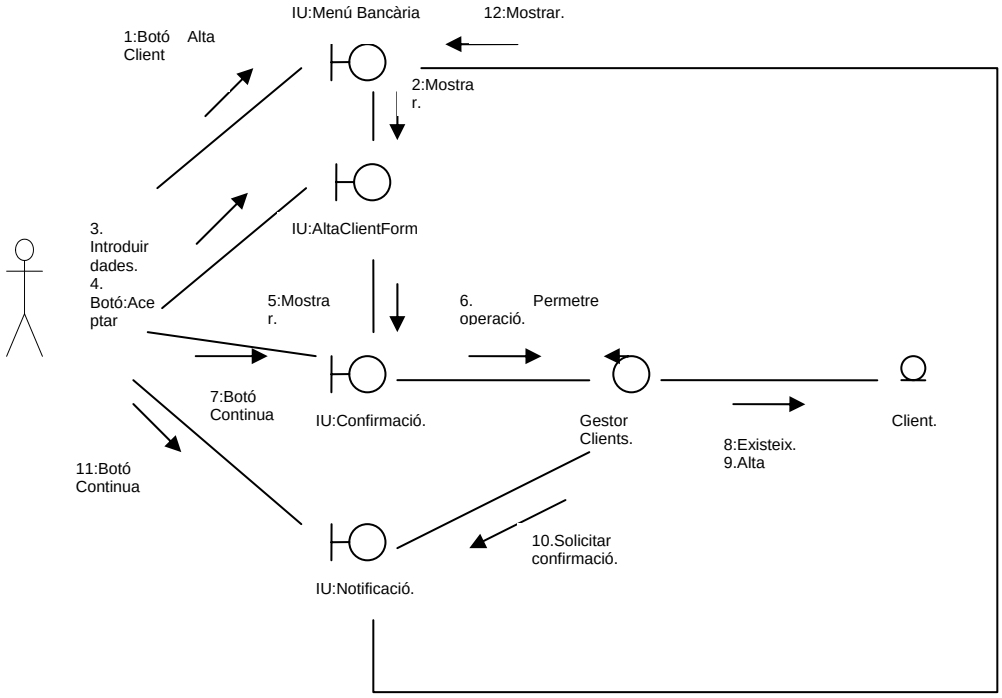


Diagrama de col.laboració corresponent al flux d'events alternatiu si l'usuari se'n adona de que ha introduit no correctes i les vol modificar:

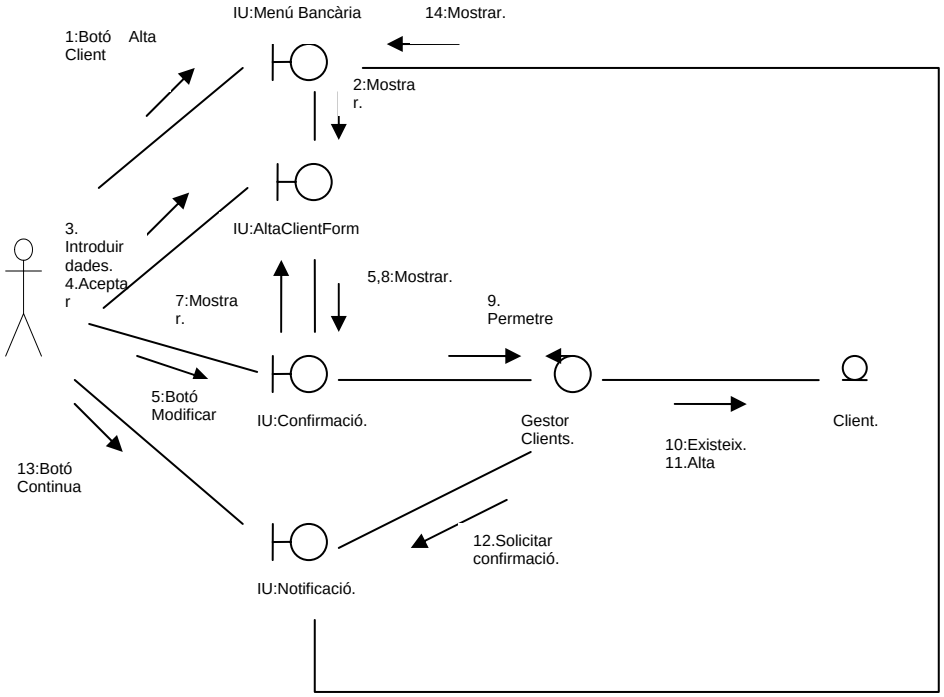
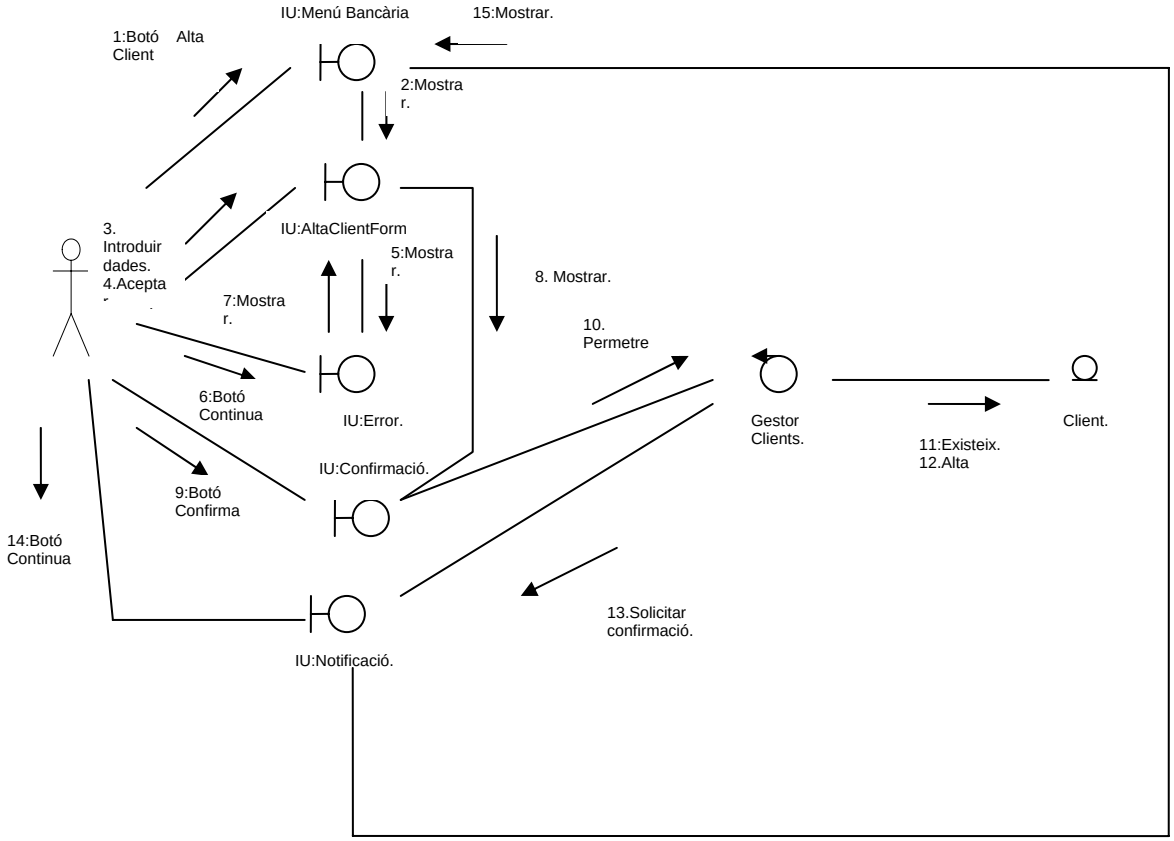
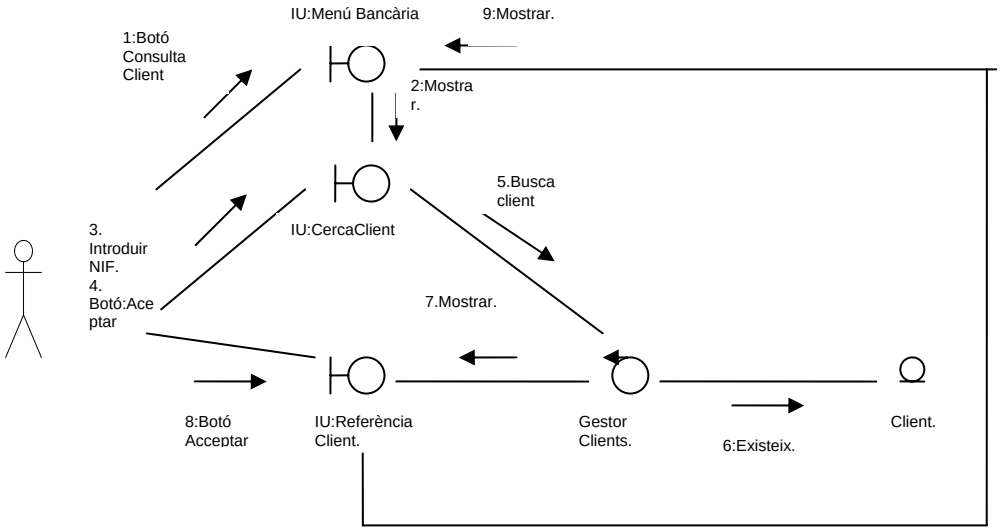


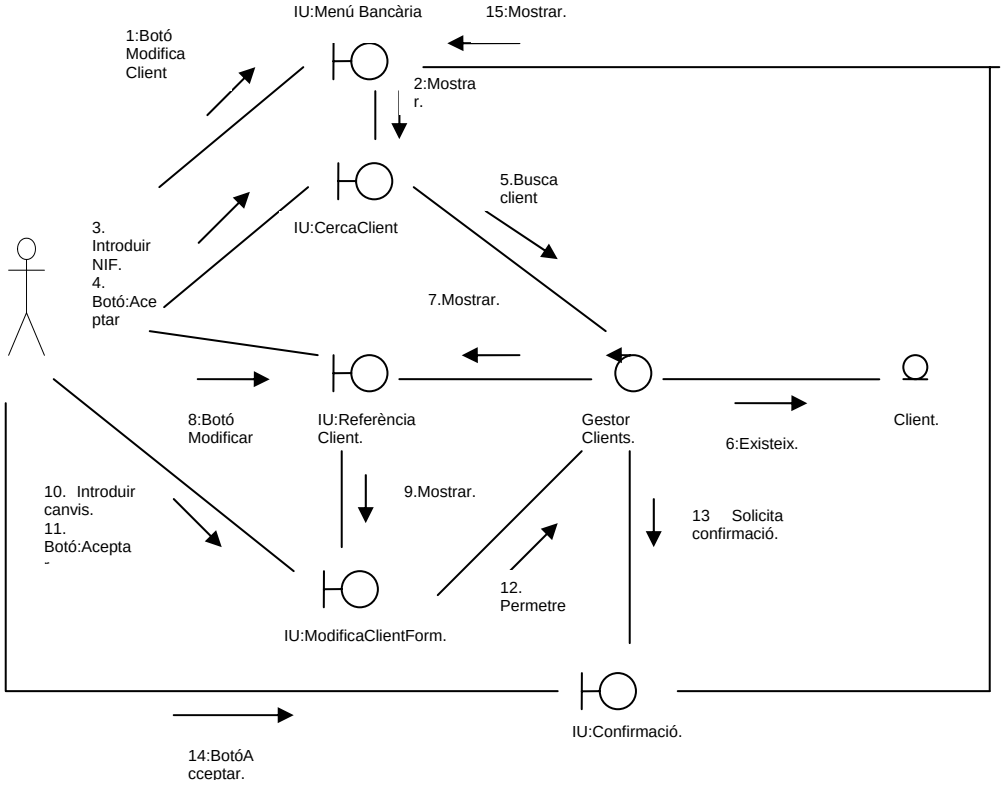
Diagrama de col.laboració corresponent al flux d'events alternatiu que es produeix quan l'usuari comet errors al introduir les dades:



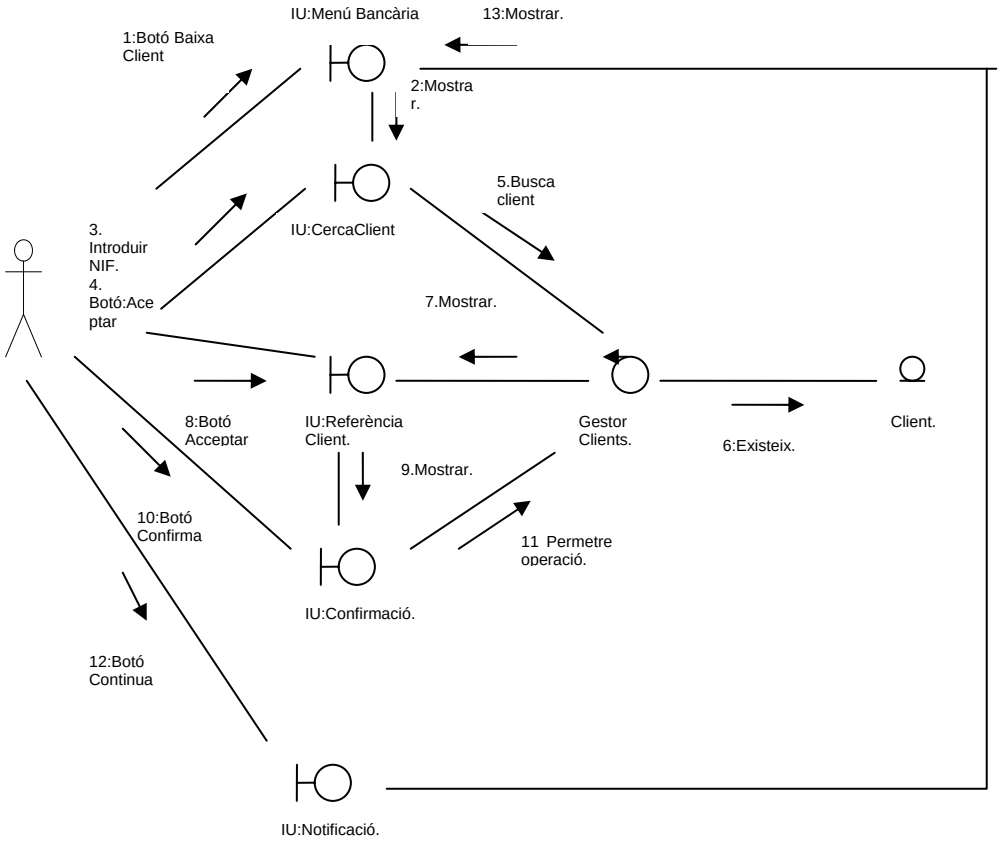
Consultar un client.



Modificar client



Baixa client.



CU2. ALTA/BAIXA/CONSULTA/MODIFICA COMPTE BANCARI.

- Frontera:
  1) IUMenuBancaria.
  2) IUAltaCompteForm.
  3) IUCercaCompte.
  4) IUConsultaCompte.
  5) IUBaixaCompte.
  6) IUMenuCompte.

- Control:
  1) GestorComptes.
- Entitat:
  1) Compte.

CU3. ALTA/BAIXA/CONSULTA/MODIFICA TARJETA.

- Frontera:
  1) IUMenuBancaria.
  2) IUAltaTarjetaForm.
  3) IUCercaTarjeta.
  4) IUConsultaTarjeta.
  5) IUBaixaTarjeta.
  6) IUModificaTarjeta.
  7) IUMenuTarjeta.

- Control:
  1) GestorTarjetes.
- Entitat:
  1) Tarjeta.

CU4. ALTA/BAIXA/CONSULTA/SOLICITUD PRÈSTEC.

- Frontera:
  1) IUMenuBancaria.
  2) IUAltaSolicitudForm.
  3) IUCercaSolicitud.
  4) IUModificaSolicitud.
  5) IUBaixaSolicitud.
  6) IUMenuSolicitud.

- Control:
  1) GestorSolicitudPrestecs.
- Entitat:
  1) SolicitudPrestec.

CU5. ALTA/BAIXA/CONSULTA/CONTRACTACIÓ DIPÒSIT.

- Frontera:
  1) IUMenuBancaria.
  2) IUAltaDipositForm.
  3) IUCercaDiposit.
  4) IUModificaDiposit.
  5) IUBaixaDiposit.
  6) IUMenuDiposit.

- Control:
  1) GestorDiposits.
- Entitat:
  1) Diposit.

CU6. INGRESSAR DINERS.

- Frontera:
  1) IUMenuBancaria.
  2) IUIngres.

- Control:
  1) GestorEfectiu.
- Entitat:
  1) Ingrés.

CU7. REINTEGRAR DINERS.

- Frontera:
  1) IUMenuBancaria.
  2) IUReintegrament.

- Control:
  1) GestorEfectiu.
- Entitat:
  1) Reintegrament.

CU8. ALTA/BAIXA/CONSULTA/MODIFICA EMPLEAT.

- Frontera:
  1) IUMenuPersonal.
  2) IUAltaEmpleatForm.
  3) IUCercaEmpleat.
  4) IUModificaEmpleat.
  5) IUBaixaEmpleat.
  6) IUMenuEmpleats.

- Control:
  1) GestorEmpleats.
- Entitat:
  1) Empleat.

CU9. ALTA/CONSULTA/MODIFICA ABSENTISME.

- Frontera:
  1) IUMenuPersonal.
  2) IUAltaAbsentismeForm.
  3) IUCercaAbsentisme.
  4) IUModificaAbsentisme.
  6) IUMenuAbsentisme.

- Control:
  1) GestorAbsentisme.
- Entitat:
  1) Absentisme.

CU10. OBRIR CAIXA CENTRAL.

- Frontera:
  1) IUMenuContable.
  2) IUObrirCaixaCentral.
- Control:
  1) GestorCaixaCentral.
- Entitat:
  1) CaixaCentral.

CU11. OBRIR CAIXA EMPLEAT.

- Frontera:
  1) IUMenuContable.
  2) IUObrirCaixaEmpleat.
- Control:
  1) GestorCaixes.
- Entitat:
  1) Caixa.

CU12. ENVIAR DINERS CAIXA.

- Frontera:
  1) IUMenuContable.
  2) IUEnviarDiners.

- Control:
  1) GestorMovimentDiners.
- Entitat:
  1) Caixa.
  2) CaixaCentral.

CU13. ENVIAR DINERS CAIXA.

- Frontera:
  1) IUMenuContable.
  2) IURetirarDiners.

- Control:
  1) GestorMovimentDiners.
- Entitat:
  1) Caixa.
  2) CaixaCentral.

CU14. TANCAR CAIXA EMPLEAT.

- Frontera:
  1) IUMenuContable.
  2) IUTancarCaixaEmpleat.

- Control:
  1) GestorCaixes.
- Entitat:
  1) Caixa.

CU15. TANCAR CAIXA CENTRAL.

- Frontera:
  1) IUMenuContable.
  2) IUTancarCaixaCentral.

- Control:
  1) GestorCaixaCentral.
- Entitat:
  1) CaixaCentral.

CU16. GENERAR BALANÇ.

- Frontera:
  1) IUMenuContable.
  2) IUGenerarBalanç.

- Control:
  1) GestorBalanç.
- Entitat:
  1) Balanç.

CU17. TANCAR BALANÇ.

- Frontera:
  1) IUMenuContable.
  2) IUTancarBalanç.

- Control:
  1) GestorBalanç.
- Entitat:
  1) Balanç.

CU18. CONSULTAR BALANÇ.

- Frontera:
  1) IUMenuContable.
  2) IUConsultarBalanç.

- Control:
  1) GestorBalanç.
- Entitat:
  1) Balanç.

Detall classes entitat.

Client

Classe.	Atribut.	Tipus.	Descripció.
Client.	Nif.	Text	Document d'identitat del client que l'identifica d'una manera única.
	Nom.	Text.	Nom del client.
	Cognom1	Text.	Primer cognom del client.
	Cognom2	Text	Segon cognom del client.
	Adreça	Text	Adreça de residència del client.
	Telèfon	Text	Nombre de telèfon del client.
	Ocupació	Text	Professió del client.
	Data alta	Data	Data en que es ona d'alta.
	Data baixa	Data	Data en que es dona de baixa.

Compte bancari.

Classe.	Atribut.	Tipus.	Descripció.
Compte bancari.	Número de compte.	Text.	Número de compte que l'identifica de manera única .
	Nif.	Text.	Nif del client al que està associat el compte.
	Data alta.	Data.	Data en que es dona d'alta el compte.
	Data baixa.	Data.	Data en que es dona de baixa el compte.

Tarjeta.

Classe.	Atribut.	Tipus.	Descripció.
Tarjeta.	Número de tarjeta.	Text.	Número de tarjeta que l'identifica de manera única .
	Número de compte.	Text.	Número de compte al que està associat la tarjeta.
	Data alta.	Data.	Data en que es dona d'alta la tarjeta.
	Data baixa.	Data.	Data en que es dona de baixa la tarjeta.
	Límit.	Número.	Quantitat màxima diaria que es pot treure.
	Típus.	Text.	Tipus de tarjeta. Només podrà pendre els valors 'Dèbit' o 'Crèdit'.

Solicitud prèstec.

Classe.	Atribut.	Tipus.	Descripció.
Solicitud prèstec.	Número de solicitud.	Text.	Número de solicitud que l'identifica de manera única .
	Número de compte.	Text.	Númeor de compte al que està associada la solicitud.
	Data alta.	Data.	Data en que es dona d'alta la solicitud.
	Data baixa.	Data.	Data en que es dona de baixa la solicitud.
	Quantitat.	Número.	Quantitat demanada a la solicitud.
	Plaç.	Número.	Número de mesos en els que s'ha de pagar el crèdit.

Dipòsit.

Classe.	Atribut.	Tipus.	Descripció.
Dipòsit.	Número de dipòsit.	Text.	Número de contractació de dipòsit que l'identifica de manera única .
	Número de compte.	Text.	Número de compte al que està associat el dipòsit.
	Data alta.	Data.	Data en que es dona d'alta el dipòsit.
	Data baixa.	Data.	Data en que es dona de baixa el dipòsit.
	Quantitat.	Número.	Quantitat dipositada.
	Plaç.	Número.	Número de mesos en que s'ha de tenir el dipòsit.
	Típus	Text.	Nom del tipus de dipòsit contractat.

Ingrés.

Classe.	Atribut.	Tipus.	Descripció.
Ingrés.	Número d'ingrés.	Text.	Número d'ingrés que l'identifica de manera única.
	Número de compte.	Text.	Número de compte al que està associat l'ingrés.
	Data d'ingrés.	Data.	Data en que es fa l'ingrés.

Reintegrament.

Classe.	Atribut.	Tipus.	Descripció.
Reintegrament.	Número de reintegrament.	Text.	Número d'ingrés que l'identifica de manera única.
	Número de compte.	Text.	Número de compte al que està associat el reintegrament.
	Data d'ingrés.	Data.	Data en que es fa el reintegrament.

Empleat.

Classe.	Atribut.	Tipus.	Descripció.
Empleat.	Nif.	Text	Document d'identitat de l'empleat que l'identifica d'una manera única.
	Nom.	Text.	Nom de l'empleat.
	Cognom1	Text.	Primer cognom del client.
	Cognom2	Text	Segon cognom del client.
	Adreça	Text	Adreça de residència del client.
	Telèfon	Text	Nombre de telèfon del client.
	Titul.lació.	Text	Titulació de l'empleat.
	Data alta	Data	Data en que es dona d'alta.
	Data baixa	Data	Data en que es dona de baixa.
	Càrrec	Text.	Càrrec de l'empleat. Pot ser 'Usuari' o 'Director'.

Absentisme.

Classe.	Atribut.	Tipus.	Descripció.
Absentisme.	Número d'absentisme.	Text.	Número d'absentisme que l'identifica de manera única .
	Nif.	Text.	Nif de l'empleat al que està associat l'absentisme.
	Data alta empleat.	Data.	Data en que es registra l'absentisme.
	Data inici baixa.	Data.	Data en que l'empleat comença a faltar.
	Data fi baixa	Data.	Darrer dia en que falta l'empleat.

Caixa central.

Classe.	Atribut.	Tipus.	Descripció.
Caixa Central.	Id Caixa Central	Text	Codi de Caixa Central que l'identifica d'una manera única.
	Quantitat inicial.	Número.	Quantitat inicial de la que disposa.
	Quantitat final.	Número.	Quantitat final de la que disposa.
	Data.	Data.	Dia de la caixa central.
	Estat.	Text.	Estat en que es troba la caixa central. Pot pendre els valors 'Oberta' o 'Tancada'.

Caixa.

Classe.	Atribut.	Tipus.	Descripció.
Caixa.	Id Caixa	Text.	Codi de Caixa que l'identifica d'una manera única.
	Quantitat inicial.	Número.	Quantitat de la que disposa a l'iniciar la jornada.
	Quantitat final.	Número.	Quantitat de la que disposa a l'acabar la jornada.
	Data.	Data.	Dia de la caixa.
	Estat.	Text.	Estat en que es troba la caixa. Pot pendre els valors 'Oberta' o 'Tancada'.
	Nif	Text	Nif de l'empleat responsable de la caixa.

Balanç.

Classe.	Atribut.	Tipus.	Descripció.
Balanç.	Id Balanç	Text	Codi de Balanç que l'identifica d'una manera única.
	Saldo Inicial.	Número.	Saldo del balanç del dia anterior.
	Saldo Final.	Número.	Saldo del balanç en acabar la jornada.
	Data.	Data.	Dia del balanç.
	Estat.	Text.	Estat en que es troba el balanç. Pot pendre els valors 'Obert' o 'Tancat'.
	Entrades	Número	Entrades al balanç.
	Sortides	Número	Sortides al balanç.

Password

Classe.	Atribut.	Tipus.	Descripció.
Password.	Id Password	Text	Codi de Password que l'identifica d'una manera única.
	Nom usuari.	Text.	Nom d'usuari.
	Password.	Text.	Password assignat a un usuari concret.

3.4 Disseny.

Aquest apartat la fase de disseny de la nostra aplicació. Comencem a tractar temes de J2EE, molts dels quals seran ampliats en el següent apartat.

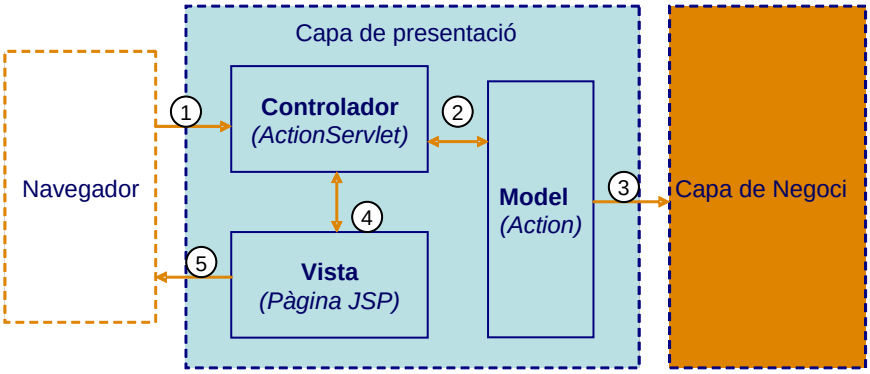
Disseny de l'arquitectura.

La nostra aplicació J2EE segueix una estructura de tres capes: capa de presentació i control en servidor, capa de negoci, capa d'integració.

Capa de presentació i control en servidor.

Encarregada de rebre les peticions dels usuaris, validar les dades que són enviades en cada petició, decidir a quin servei de la capa de negoci redirigir la petició per que sigui processada i mostrar a l'usuari el resultat de l'operació. Aquesta capa s'encarrega també de controlar el flux de navegació dels usuaris per l'aplicació.

El disseny d'aquesta capa seguirà el patró MVC (Model Vista Controlador). Per això, s'utilitzarà com a base el framework Struts. Per tant el control estarà implementat en un servlet (ActionServlet), el model en una classe Action i les vistes en JSPs.



El controlador és el sistema de processament de les dades enviades i control de navegació. Rep les dades en l'objecte request, decidint quina lògica de negoci s'ha d'aplicar sobre aquests. El principal component del controlador és l'ActionServlet. Es configura per a utilitzar ActionMappings, que defineixen la navegació a través d'un arxiu de configuració (struts-config.xml). Cada enviament és emmagatzemat en un ActionForm i rebut finalment per un object Action que el processa.
La vista són pàgines JSP i components de presentació. Inclou codi HTML, amb els seus respectius elements que formen la presentació, a més de tags estàndars i

crides a JavaBeans. Struts inclou un conjunt de tag libs que serveixen d'ajuda en l'el.laboració de la presentació.
El model està format pels action que processen les peticions i criden els Session Beans de la capa de negoci per a poder donar resposta a les peticions.

Veiem un esquema:

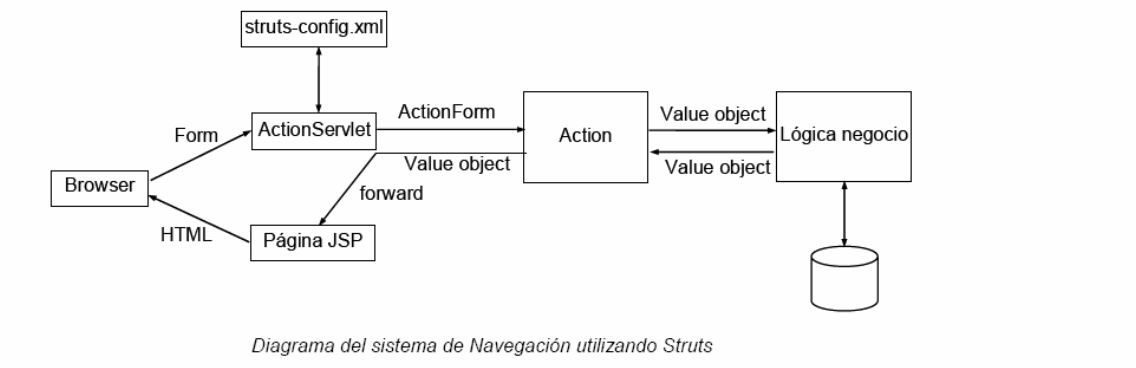
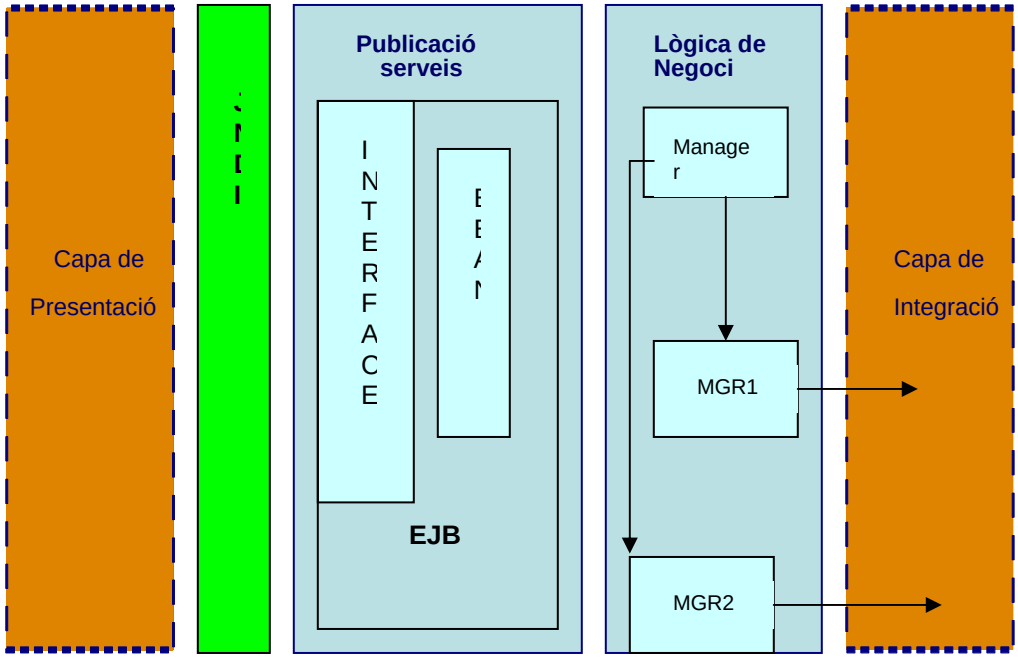


Diagrama del sistema de Navegación utilizando Struts

Per fer les validacions en servidor i client farem servir la llibreria Validator d'Struts.

Capa de negoci.

La capa de negoci del nostre sistema contindrà la lògica de negoci requerida per l'aplicació. Interactuarà amb la capa de presentació per atendre les peticions dels usuaris del sistema i amb la capa d'integració per a recuperar o manipular informació emmagatzemada a la base de dades, o en un futur interactuar amb sistemes externs.



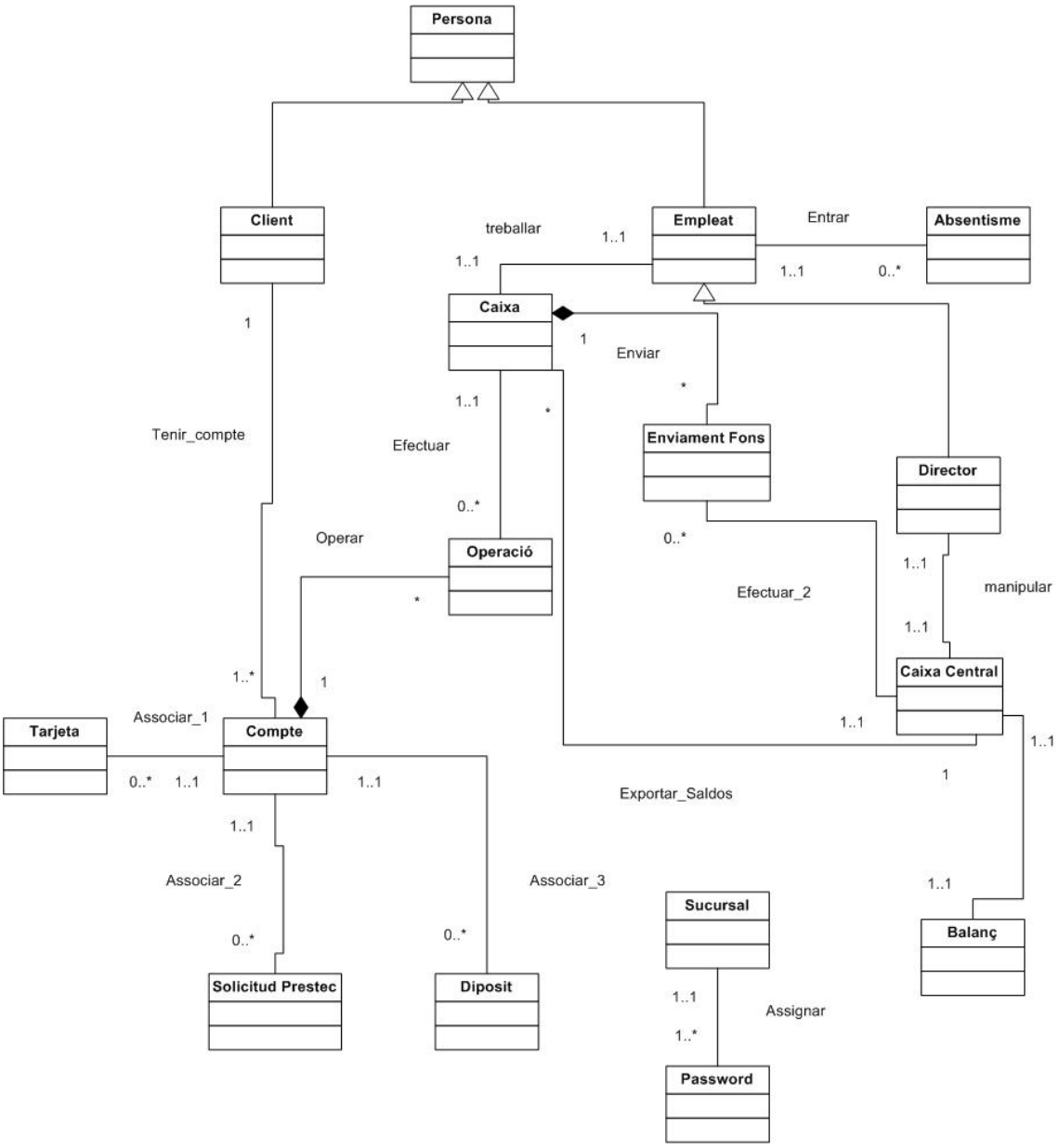
La capa de negoci serà accedida desde els clients web, els quals utilitzaran els Actions de Struts, utilitzant el patró de disseny Service Locator. A més, seran utilitzats per altres components de negoci que dessitjin el servei exposat a través dels EJB. Per a realitzar el control transaccional utilitzarem les propietats dels EJB que s'executen en el servidor de aplicacions (GlassFish de Sun). Els EJB compleixen la funcionalitat d'exportar els serveis i deleguen la lògica de negoci als components de negoci els quals s'anomenaran Manager (MGR).

Capa d'integració.

A aquesta capa s'encapsula l'accés a les fonts d'emmagatzemament de tipus persistent propis de la nostra aplicació. Aquesta capa la implementarem fent servir l'ORM Hibernate 3.

Disseny de la persistència.

Veiem ara el disseny de la persistència de la nostra aplicació. Aquesta s'implementarà mitjançant el SGBD Oracle10XE.


Persona
Client
Empleat
Absentisme
treballar
1..1
1..1
Entrar
1..1
0..*
Caixa
1
Enviar
*
1..1
*
Tenir_compte
Efectuar
Enviament Fons
Director
0..*
Operar
Operació
0..*
1..1
Efectuar_2
manipular
*
1..1
1..*
1
Caixa Central
Associar_1
Tarjeta
Compte
1..1
0..*
1..1
1..1
1..1
1
Exportar_Saldos
1..1
Associar_2
Associar_3
1..1
Sucursal
0..*
0..*
Balanç
Solicitud Prestec
Diposit
1..1
Assignar
1..*
Password

No s'ha posat al diagrama per claredad però totes les taules tenen un camp corresponent a la sucursal a on està associada.

EMPLEAT.
nif, nom, cognom1, cognom2, adreça, població, telèfon, titulació, càrrec, dataAlta, dataBaixa.

CLIENT
nif, nom, cognom1, cognom2, adreça, població , telèfon, ocupacio, dataAlta, dataBaixa.

COMPTE.
numCompte, dataAlta, dataBaixa.

TARJETA
numTarjeta, dataAlta, dataBaixa, límit, típus.

SOLICITUD PRÈSTEC
numSolicitud, dataAlta, dataBaixa, quantitat, plaç.

DIPÒSIT
numDipòsit, dataAlta, dataBaixa, quantitat, plaç, típus.

OPERACIÓ.
dataOperació, quantitat.

CAIXA.
quantitatInicial, quantitatFinal, data, estat.

ENVIAMENT FONS.
dataEnviament, quantitat.

CAIXA CENTRAL.
quantitatInicial, quantitatFinal, rebutExtern, rebutCaixes, data, estat.

BALANÇ.
saldoInicial, saldoFinal, data, estat.

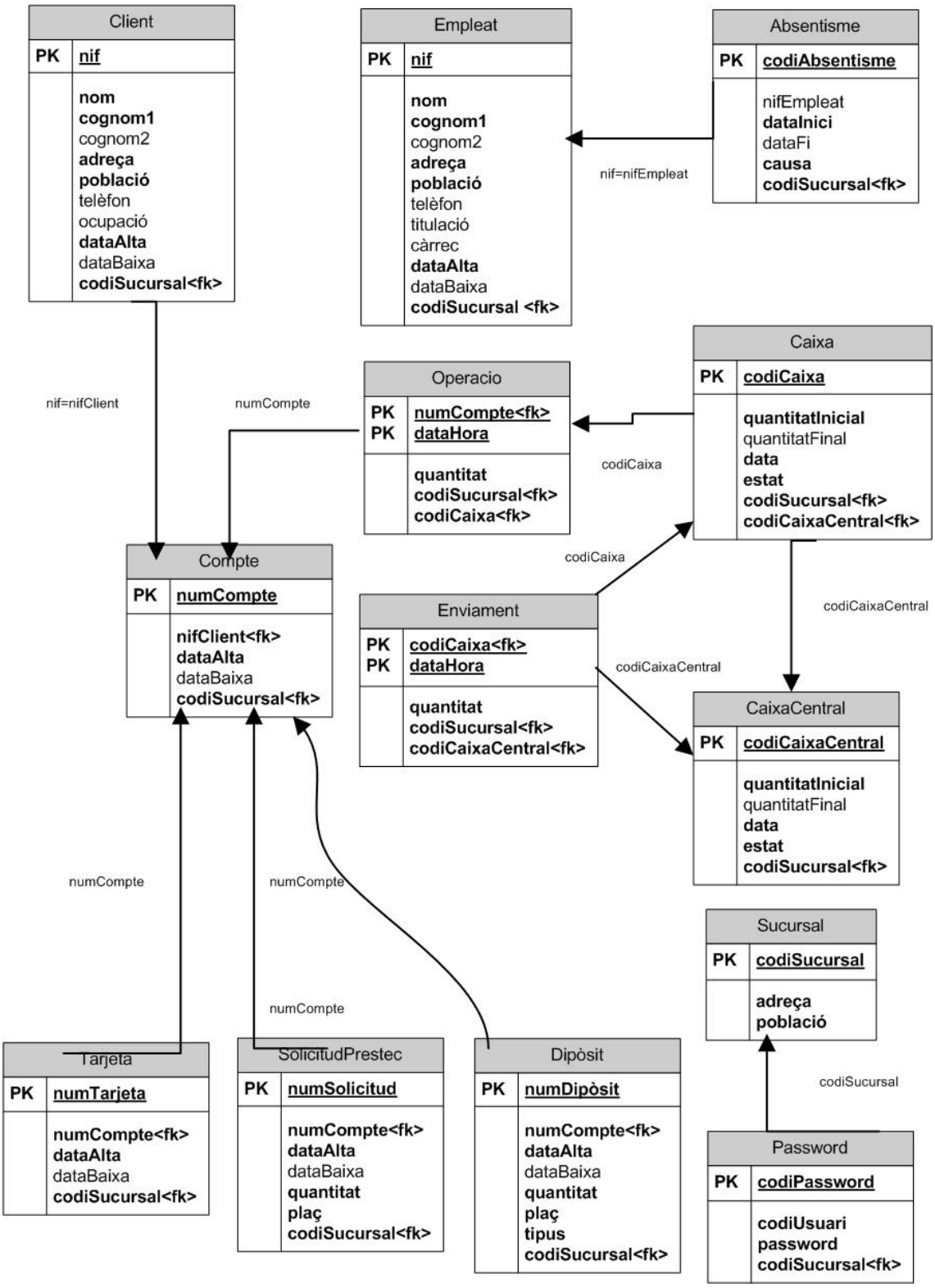
ABSENTISME.

codiAbsentisme, data, dataInici, dataFi, causa.

PASSWORD.
codiUsuari, password.

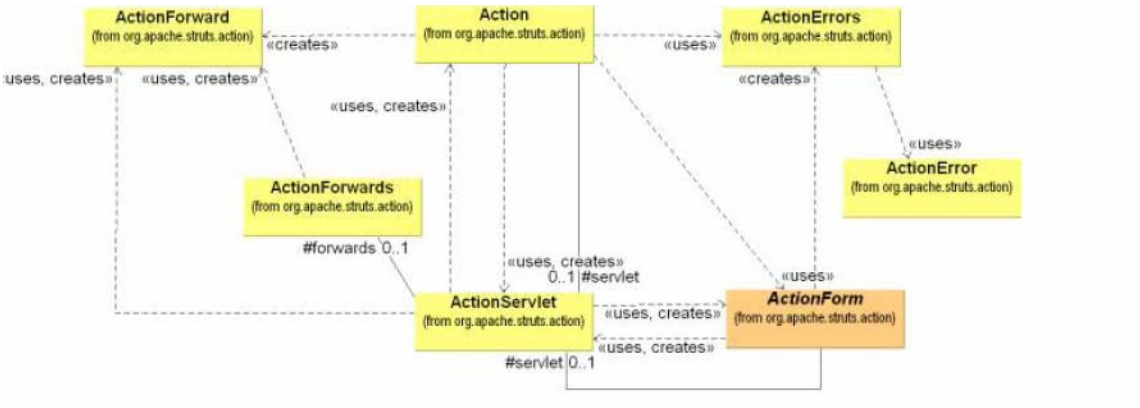
SUCURSAL.
codiSucursal, adreça, població.

Transformació al model lògic:


Client
PK nif
nom
cognom1
cognom2
adreça
població
telèfon
ocupació
dataAlta
dataBaixa
codiSucursal<fk>
Empleat
PK nif
nom
cognom1
cognom2
adreça
població
telèfon
titulació
càrrec
dataAlta
dataBaixa
codiSucursal <fk>
Absentisme
PK codiAbsentisme
nifEmpleat
dataInici
dataFi
causa
codiSucursal<fk>
nif=nifEmpleat
Caixa
PK codiCaixa
quantitatInicial
quantitatFinal
data
estat
codiSucursal<fk>
codiCaixaCentral<fk>
Operacio
PK numCompte<fk>
PK dataHora
quantitat
codiSucursal<fk>
codiCaixa<fk>
nif=nifClient
numCompte
codiCaixa
Compte
PK numCompte
nifClient<fk>
dataAlta
dataBaixa
codiSucursal<fk>
codiCaixa
Enviament
PK codiCaixa<fk>
PK dataHora
quantitat
codiSucursal<fk>
codiCaixaCentral<fk>
codiCaixaCentral
codiCaixaCentral
CaixaCentral
PK codiCaixaCentral
quantitatInicial
quantitatFinal
data
estat
codiSucursal<fk>
numCompte
numCompte
numCompte
Sucursal
PK codiSucursal
adreça
població
codiSucursal
Tarjeta
PK numTarjeta
numCompte<fk>
dataAlta
dataBaixa
codiSucursal<fk>
SolicitudPrestec
PK numSolicitud
numCompte<fk>
dataAlta
dataBaixa
quantitat
plaç
codiSucursal<fk>
Dipòsit
PK numDipòsit
numCompte<fk>
dataAlta
dataBaixa
quantitat
plaç
tipus
codiSucursal<fk>
Password
PK codiPassword
codiUsuari
password
codiSucursal<fk>

Disseny dels casos d'us i classes.

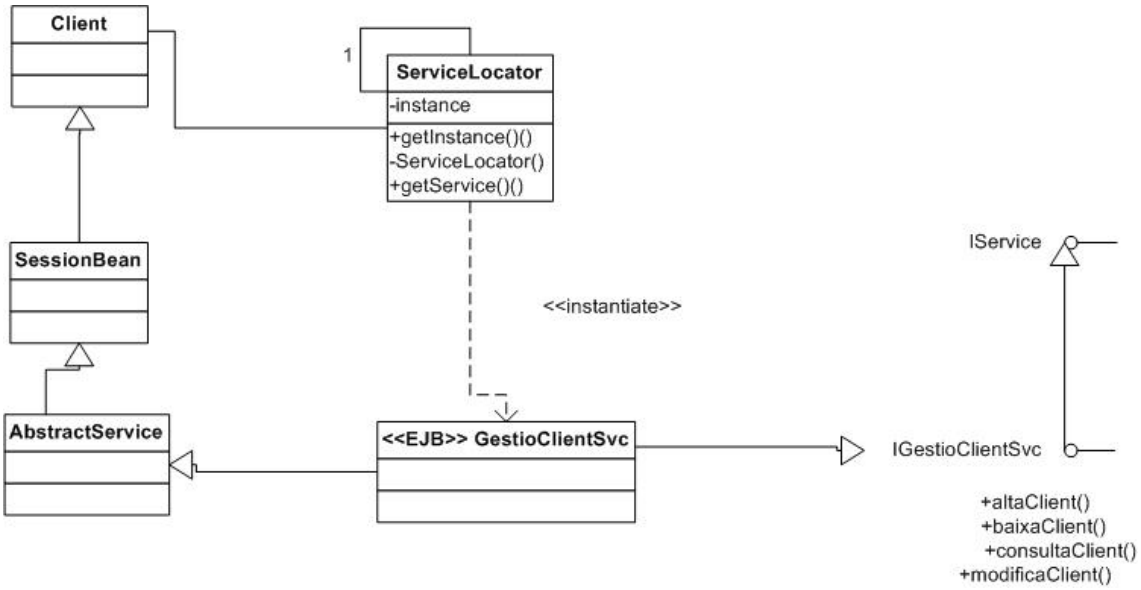
La capa de presentació seguirà fidelment les especificacions d'Struts per lo que presento un diagrama genèric:



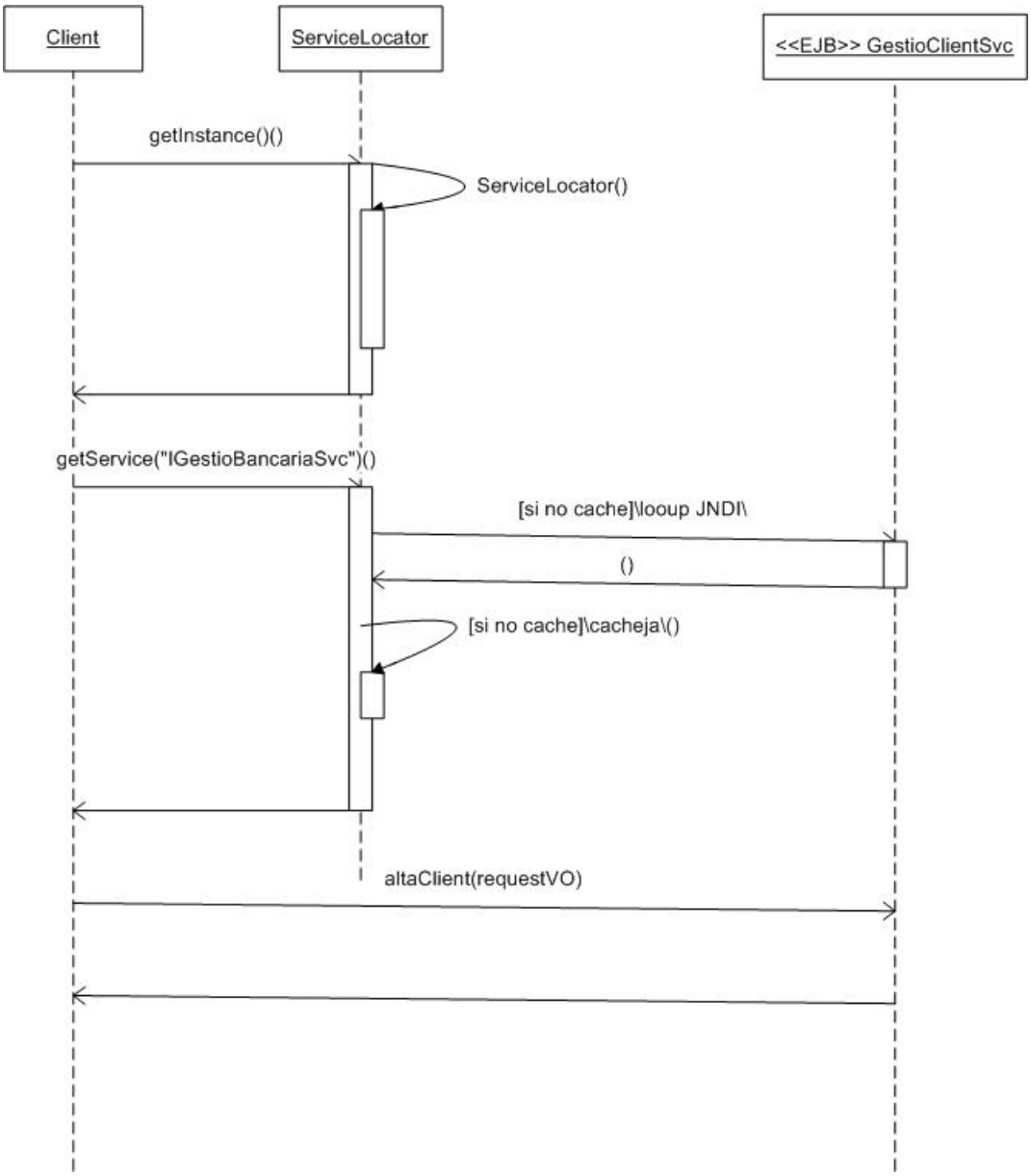
En quant a la capa de negoci lo que explicarem pel subsistema bancari /clients servirà pels altres mòduls.

Localitzacio de serveis.

Les classes externes al mòdul de negoci, accediran als serveis proporcionats per aquesta capa a través d'un localitzador, el ServiceLocator, seguint així el patró Service Locator.
Quan un client vol accedir a un determinat servei, li demanarà al localitzador una implementació d'un determinat interface, i és a aquesta implementació de l'interface. La classe ServiceLocator implementa la interface Singleton, per lo que només hi haurà una única instància per màquina virtual.
Veiem el diagrama de classes:

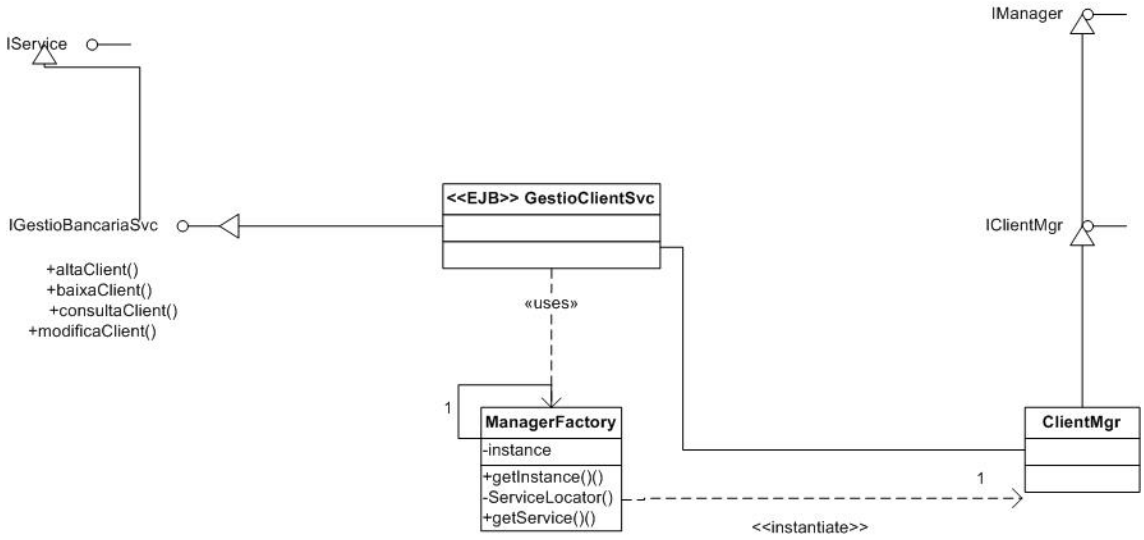


Ara el de sequència:



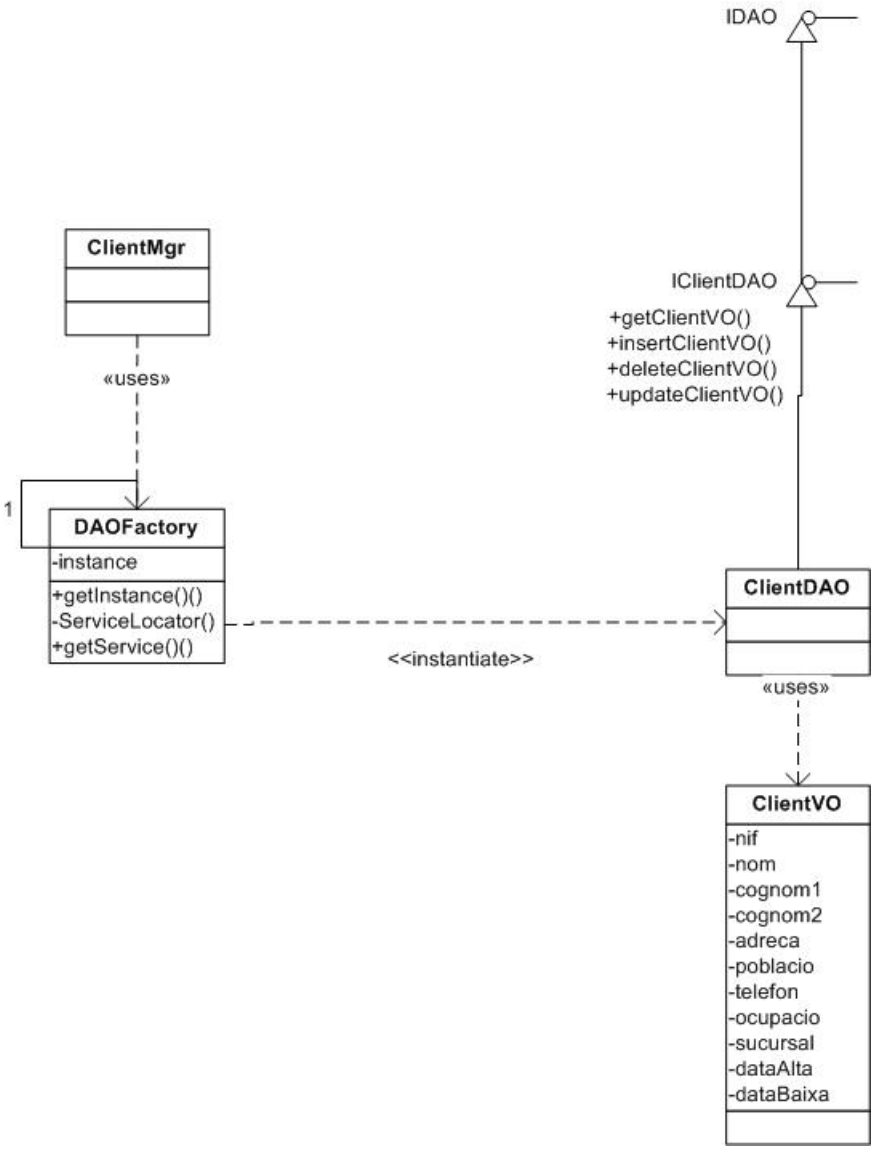
Manager.

Veiem ara la part que hi ha entre el Session Bean i la integració:



DAO.

Veiem ara com queden els DAOs.



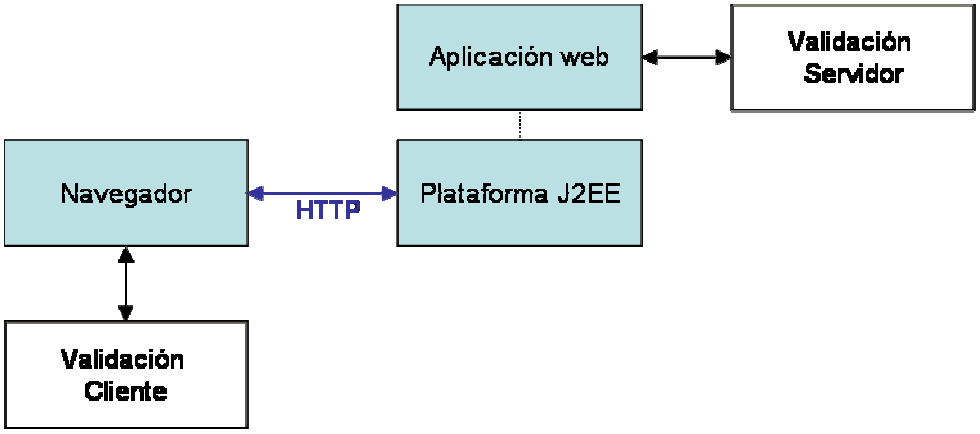
3.5 Implementació.

Anem a explicar ara una mica més en profunditat com hem implementat l'anterior disseny i els patrons i tecnologies utilitzades:

Capa presentació.

Un framework es un conjunt de classes e interfaces que treballen per a solucionar un tipus específic de problema software.
El framework seleccionat per a construir la capa de presentación de l'aplicació es Struts, creat per la fundació Apache.
Ja hem explicat a l'apartat de disseny el funcionament general d'Struts, per lo que aquí ens centrarem en el tema de la validació de camps de formularis, en concret del framework Validator, integrat a Struts (anomenat també Struts Validator).

El procés de validació de les dades en una aplicació web segueix , típicament, una estructura similar a la d'aquí:



Struts incorpora la noció de formularis d'acció, ActionForm, que proporcionen als actions accés a les dades recuperades del formulari des d'on es va accedir. Un formulari d'acció incorpora el mètode validate(), que l'aplicació sobrescriu per a afegir lògica de validació de les dades del formulari. Sense Struts Validator es necessari dissenyar i programar dins del mètode validate() tota la lògica de validació, així com gestionar tots els missatges d'error a usar per a aquelles validacions fallides.
L'us d'Struts Validator permet fer ús de funcionalitats de validació ja desenvolupades, de manera que no faci falta programar la validació dels camps ni l'emmagatzament dels missatges d'error. Això s'aconsegueix fent ús de classes pròpies d'Struts Validator, que extenen les classes d'Struts com ActionForm.
Aquest framework de validació conté un conjunt de rutines, que es corresponen amb mètodes desenvolupats en Java. Cada mètode efectua un tipus específic de validació. Per defecte, ja es proporcionen implementacions per a certes rutines que satisfan els casos de validació més comuns.Tot i així Struts Validator ens proporciona les eines necessàries per a desenvolupar validacions a mida.

Capa de negoci.

En aquesta capa es fan servir tota una sèrie de patrons de disseny. S'ha de dir que en aquest punt m'he detingut especialment a comentar el codi per explicar "sobre el terreny" el que fa cada mètode en cada classe. Els patrons utilitzats han estat els següents (d'alguns ja hem parlat a l'etapa de disseny):

- Locator: realment pertany a la capa de presentació, i és el nexe entre aquesta capa i la de negoci.
  Els serveis de la nostra aplicació (i normalment totes les que utilitzen J2EE) estan representats per uns components diferenciats anomenats EJBs. Aquest fet afegeix flexibilitat a les aplicacions així com una distribució de la

càrrega dels treball. Per contra, el fet d'invocar a un component de forma remota afegeix una sobrecàrrega de treball, ja que s'ha de buscar el recurs a través de JNDI de forma remota. Per a minimitzar aquest problema es dissenya un component anomenat Service Locator.
Aquest component "cacheja" els objectes demanats al JNDI, de manera que la primera vegada que són demanats, se'ls crida a través del JNDI i s'emmagatzemen en una caché (usualment, una HashMap). D'aquesta manera s'obtenen els beneficis següents:

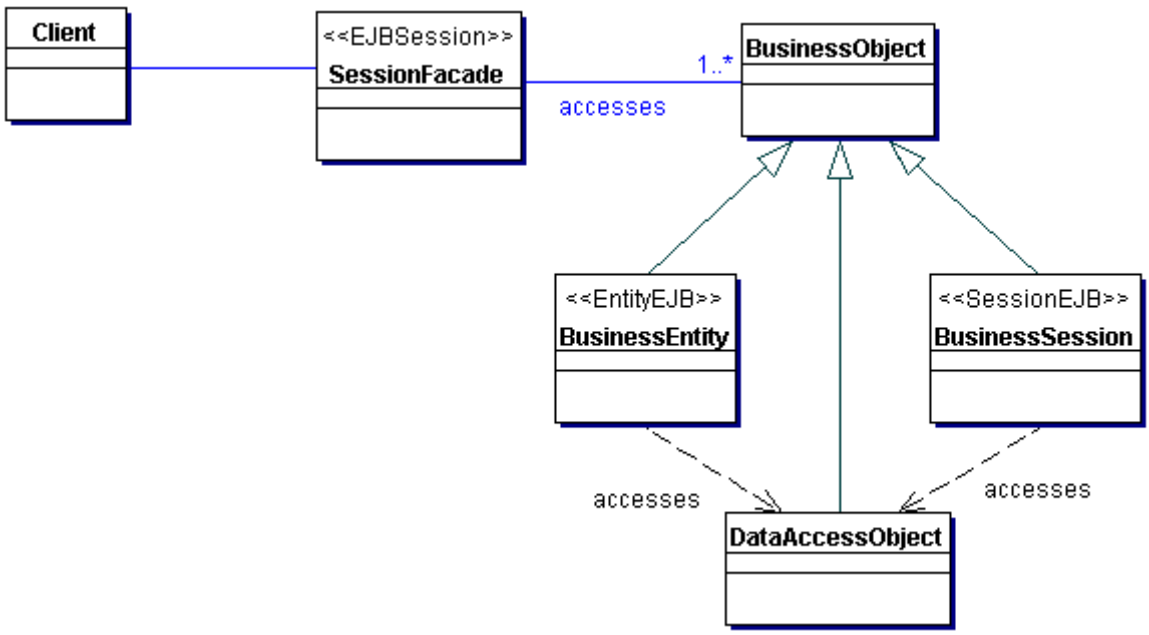
a) S'evita crear el context inicial JNDI en múltiples ocasions, ja que la tasca és realitzada pel Service Locator en l'arrancament de la classe corresponent.

b) La cerca de l'EJB no es realitza cada vegada que s'accedeix a ell, sino que es realitza a l'inici, mantenint així la referència per a futurs accessos.

c) La forma en que s'ha de buscar l'EJB s'independitza del client que l'invoca, de manera que es poden realitzar crides mitjançant diferents mecanismes a diferents lògiques de negoci, sense que per això es vegi afectat el client.

d) El Service Locator es tracta d'una classe singleton, de tal forma que pooseix una única instància en memòria, lo que permet ocupar menys espai a memòria, a més de que es troba sempre construida.

- Façade: Aquest patró simplifica l'accés a un conjunt de classes proporcionant una única classe que tots utilitzen per a comunicar-se amb aquest conjunt de classes. L'aventatge principal és que els clients no necessiten coneixer les classes que hi han darrera la classe Facade, podent canviar les classes "ocultades" sense necessitat de canviar els clients.



- Interface: Defineix un comportament independent d'on vagi a ser utilitzat. Això permet un desacoplament entre comportament i classe. A la nostra aplicació es pot observar com sempre utilitzem per a fer crides d'objectes, passar com a paràmetre, etc... interfaces, de manera que en un moment donat podem canviar la classe que implemnti aquestes interfaces sense haver de tocar res més.

- Configuration: Per a un bon manteniment de l'aplicació és important que totes les dades de configuració es trobin en fitxers, fora del codi propiament dit de l'aplicació. Aquest patró consisteix bàsicament en això, en llegir el màxim de dades de configuracions de fitxers properties. Per a això s'utilitza la classe ConfigurationManager.

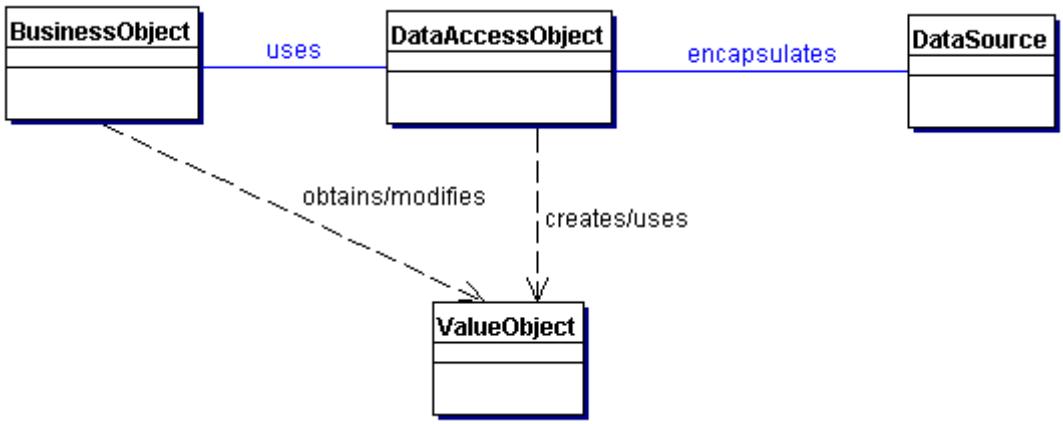
- ManagerFactory:L'obtenció d'objectes de tipus Manager, es fa a través del Manager Factory. El seu funcionament és com s'explica a continuació: una classe de l'aplicació requereix un Manager per a invocar-lo. Lo normal en aquest cas és que la pròpia classe instanciï l'objecte Manager, l'usi i després el destrueixi. La instanciació d'objectes Manager pot arribar a ser molt costosa en temps, per lo que aquest enfocament no és l'adequat en entorns on el rendiment és important. Per això sorgeix el Manager Factory, els objectes Manager no són instanciats directament desde la classe que els ha d'utilitzar, sino que aquesta classe se'ls demana a unaltra classe, el ManagerFactory, la qual els instancia i els retorna a la classe demandant. En el ManagerFactory s'implementa una estratègia caché, per evitar haver d'instanciar el mateix Factory cada cop que s'ha d'utilitzar.

- Singleton: Aquest patró s'utilitza per a assegurar que una classe te una única instància i proporciona un punt d'accés global a ella. És necessari quan hi ha classes que han de gestionar de manera utilitzada un recurs. Aquest patró és utilitzat a la vegada per la majoria dels altres patrons.

- Caché: Aquest patró com el de Singleton és utilitzat com ja hem vist per la majoria dels altres patrons. El seu funcionament és senzill: a l'hora d'instanciar un objecte primer mira si el tenim a la nostra cache (normalment un HashMap), si és així el recuperem, sino el creem.

- Value object (VO): Aquest patró té l'objectiu d'encapsular els objectes de negoci, de manera que en comptes de intercanviar informació fent servir els seus atributs ho fem intercanviant objectes, d'aquesta manera es redueix la crides remotes, habituals en l'arquitectura J2EE. Aquest patró a més és molt convenient tenint en compte que utilitzarem un ORM com Hibernate.

- Data Acces Object (DAO): Realment aquest patró hauria d'estar a l'apartat de la capa d'integració. El posem aquí perque estiguin explicats tots al mateix apartat.
  El DAO implementa el mecanisme d'accés requerit per treballar amb la font de dades. Aquesta font de dades pot ser un emmagatzament persistent com una RDMBS, un servei extern com un intercanvi B2B, etc...Lo que fa aquest patró és ocultar la implementació de la font de dades als seus clients. Esencialment, el DAO actua com un adaptador entre el component i la font de dades.



Capa d'integració.

La programació orientada a objectes i les bases de dades relacionals es basen en dos paradigmes diferents. El model relacional tracta amb relacions i conjunts. En canvi el paradigma orientat a objectes tracta amb objectes i associacions entre ells. Hi ha una desavenència entre aquests dos paradigmes, la també anomenada diferència objecte – relacional. Aquesta diferència s'amplia molt rapidament si tens grans models d'objectes.
És degut a aquesta diferència per lo que apareixen els mapejadors objecte-relacional (ORM) com és el cas d'Hibernate.
Hibernate es basa en dos fitxers de configuració, hibernate.cfg.xml per a la configuració general de l'aplicació (conexió jdbc, datasources, ...) i un per cada classe entitat NomClasse.hbm.xml a on es fa el "mapping" entre classes i taules de la base de dades relacional.
Tot i que no s'ha tocat en l'aplicació en profunditat Hibernate n'hi ha hagut prou per comprovar els seus avantatges, especialment en lo que fa en l'ocultació al programador de bona part del model relacional (un cop has contruit els fitxers xml, això sí).

4. Conclusions.

Les conclusions que trec després d'haver fet aquest projecte són totes positives; m'ha permés enfrontar-me a molts aspectes de la construcció d'una aplicació i m'ha permés conèixer la arquitectura J2EE.
També he entrat al món dels framework (amb Struts i Hibernate) que evolucionen molt rapidament, fins al punt que fins i tot (els esmentats Struts i Hibernate) s'estan quedant desfasats respecte per exemple lels Java Server Faces (JSF) i, Spring, EJB3 Entity,..
Considero que la arquitectura J2EE té un gran futur en el camp de les aplicacions empresarials ja que dona resposta a totes les demandes empresarials sent capaç d'integrar-se amb diferents tecnologies sense gaire esforç.
El límit temporal d'aquest projecte i l'amplitud de la tecnologia J2EE no m'ha permés aprofondir molts aspectes J2EE que m'han semblat interessants en una visió superficial com per exemple JMS, JAXP, WebServices i en general tot el tema d'integració i comunicació amb altres sistemes.
Trobo important dir, a més, que després de fer una ullada al pla d'estudis del segon cicle d'informàtica potser un TFC de J2EE seria més adequat per aquest segon cicle.
Tot i així com he dit abans trobo molt positiu tot l'esforç extra que he hagut de fer per adquirir tots els coneixements nous referent a aquesta tecnologia.

5. Eines i Software utilitzats.

Per portar a terme aquest projecte s'han fet servir diferents eines i programari:

- Part d'Enginyeria Software: Per a fer els diagrames UML i similars hem fet servir bàsicament una versió de prova del Microsoft Visio.
- Entorn integrat programació (IDE): Al final em vaig decantar per Netbeans 6.1, ja que l'altre alternativa que vaig estudiar, Eclipse, requeria de la instal.lació de diferents plug-in que en el cas de Netbeans no eren necessaris.
- Servidor d'aplicacions: En part derivada de l'elecció anterior em vaig decantar pel servidor d'aplicacions GlassFish v2 de Sun que venia integrat amb Netbeans 6.1. És un servidor potent que disposa d'un contenidor per EJB seguint la darrera especificacions d'EJB 3.0.
- Base de dades: Em vaig decantar per Oracle10XE per ser una base de dades gratuita que té el mateix nucli que la seva germana major Oracle10g, utilitzada en moltes empreses. Unaltre raó ha sigut estar matriculat simultaniament al PFC a l'assignatura SGBD, a on es fa servir la mateixa base de dades.
- Presentació: Microsoft PowerPoint.
- Altres: Paquet Office2003 de Microsoft.

6. Bibliografia.

Beginning Hibernate de Dave Minter and Jeff Linwood. Apress .

Jakarta Struts de O'Reilly Chuck Cavaness.

Core J2EE Patterns. Second Edition. Deepak Alur.

http://java.sun.com/

http://struts.apache.org/

http://siul02.si.ehu.es/~alfredo/iso/06Patrones.pdf

http://www.programacion.com/java/tutorial/patrones2/8/

7. Anexos.
7.1 Guia de proves.

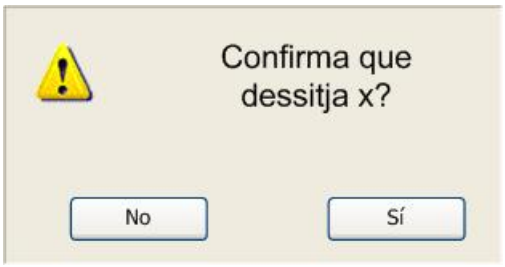
Número prova.	Descripció acció.	Resultat esperat.	Motiu.
1	Entrem un usuari i password correctes amb perfil "director".	Pantalla d'inici amb tot el menú habilitat.	Veure l'aspecte de l'aplicació amb perfil "director". Veure a la part superior els codis d'usuari i sucursal emmagatzemats en variables de sessió.
2	Clickem l'enllaç "Sortir".	L'usuari surt de l'aplicació i es torna a carregar la pantalla de login.	Comprovar la sortida de l'aplicació. Utilització d'una funció javascript per a el.liminar frames.
3	Entrem un usuari i password correctes amb perfil "empleat".	Pantalla d'inici amb el menú habilitat només per a funcions bancàries.	Veure l'aspecte de l'aplicació amb perfil "empleat".
4	Després de sortir de l'aplicació entren dades incorrectes en quant a format a la pantalla de login.(Començant per no ficar res als camps).	Missatges d'error javascript i recàrrega de la pantalla de login.	Comprovar el funcionament d'Struts Validator.
5	Introduir un usuari i password amb format correcte pero que no existeix.	Missatge d'error explicatiu.	Comprovar validació usuari contra base de dades.
6	Entrar a l'aplicació amb qualsevol usuari existent, i mitjançant el menu entrar a la pantalla alta client i donar d'alta un client que no existeixi.	Pantalla clients.	Comprovar la funcionalitat alta client.
7	Donar d'alta un client que ja existeixi.	Missatge d'error explicatiu.	Utilització de l'excepció llançada per la base de dades per impedir la repetició de clients.
8	Consultar les dades d'un client existent introduint el seu NIF.	Pantalla amb les dades del client solicitat.	Comprovar la funcionalitat consulta client.
9	Consultar les dades d'un client no existent introduint el seu NIF.	Missatge d'error explicatiu.	Comprovar la funcionalitat consulta client.
10	Donar de baixa un client introduint el seu NIF	Pantalla amb les dades del client com a confirmació. Després d'acceptar esborra el client i torna a la pantalla de clients.	Comprovar funcionalitat baixa client.
11	Donar de baixa un client no existent introduint el seu NIF	Missatge d'error explicatiu.	Comprovar funcionalitat baixa client.

12	Modificar un client introduint el seu NIF.	Pantalla amb les dades del client amb els camps que es poden modificar habilitats.. Després d'acceptar modifica el client i torna a la pantalla de clients.	Comprovar funcionalitat modifica client.
13	Modificar un client no existent introduint el seu NIF.	Missatge d'error explicatiu.	Comprovar funcionalitat modifica client.
14	Llistar client.	Pantalla amb 11 clients que pitjant "mes" es veuen més clients. Es pot veure en detall cada un d'aquests client mitjançant els botons de radio.	Comprovar funcionalitat llista client.

7.2 Especificació de pantalles.

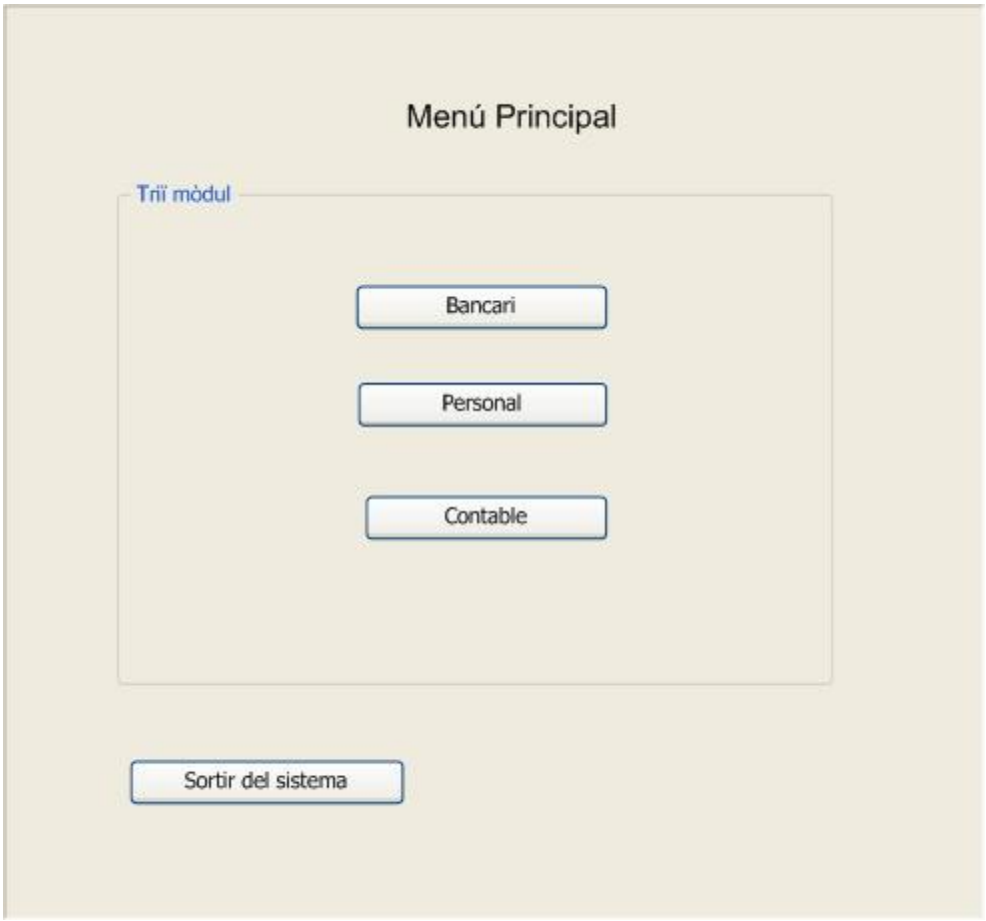
Pantalla login.

Pantalla confirmació.



Pantalla error

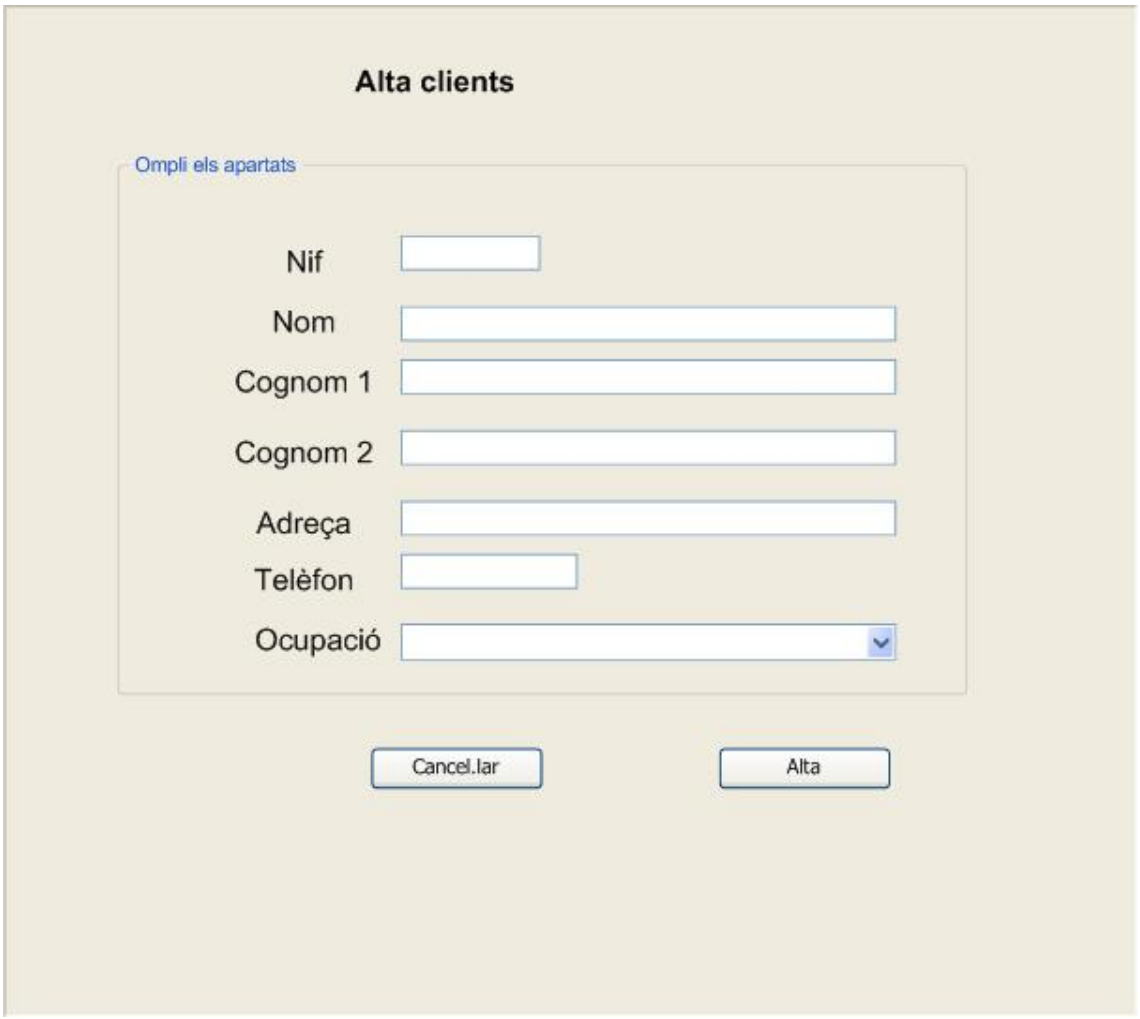
Pantalla menú principal.



Pantalla Menú Gestió Bancària.

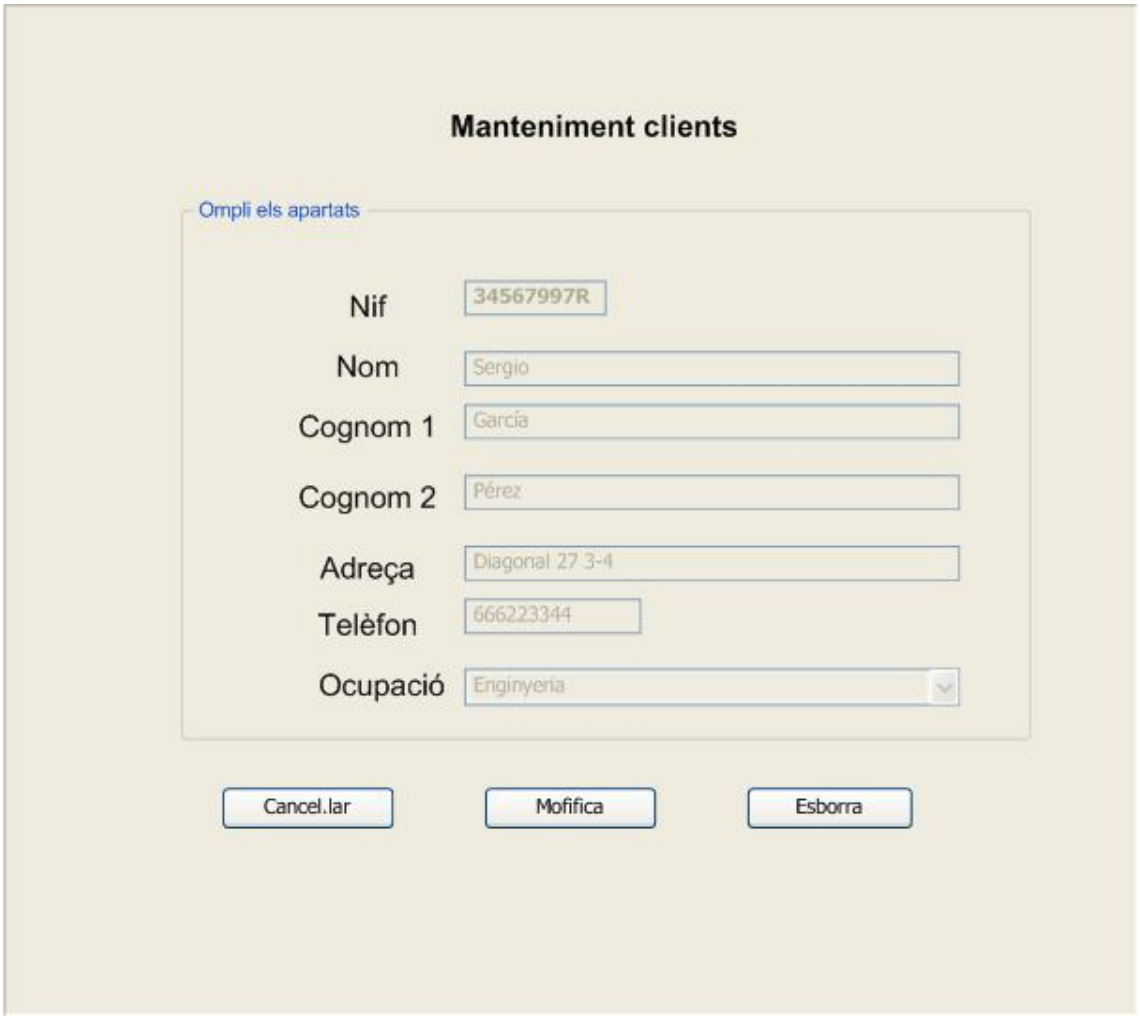
Pantalla clients.

Pantalla Alta Client.



Pantalla localitza el client:

Pantalla Modifica/Consulta/Esborra client (Manteniment clients)



Pantalla modifica clients.

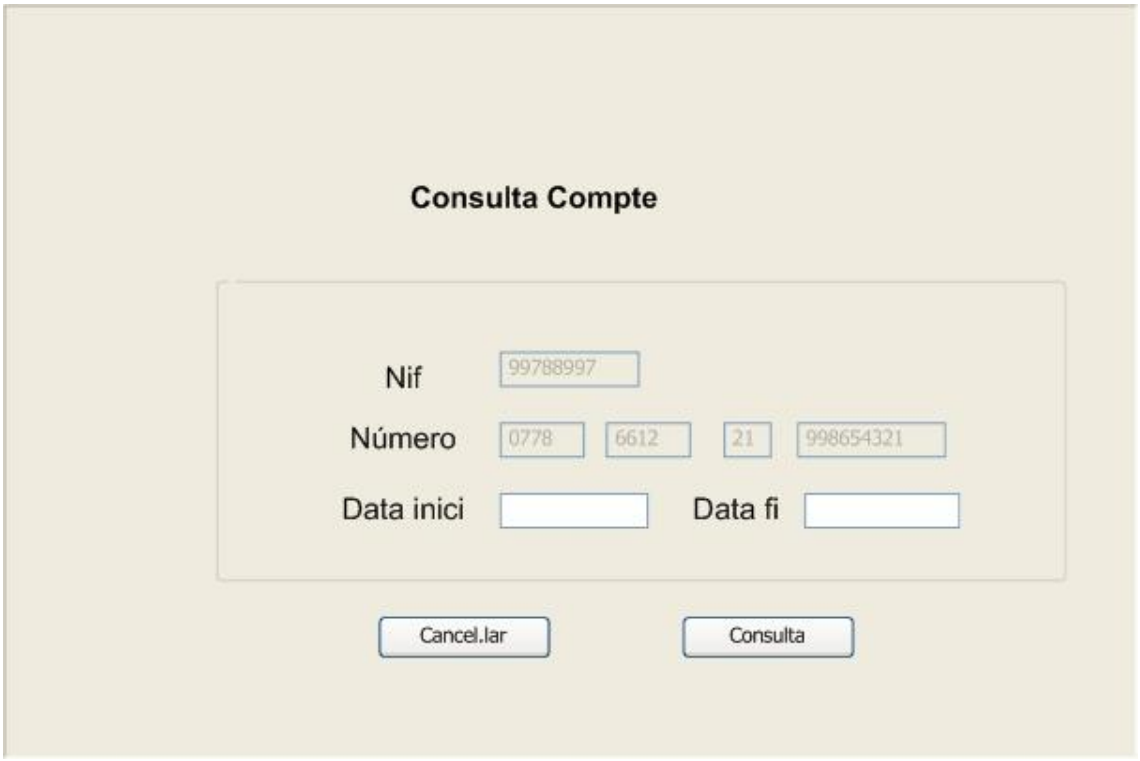
La mateixa pantalla anterior amb els camps habilitats.

Pantalla comptes bancaris.

Pantalla Alta Compte

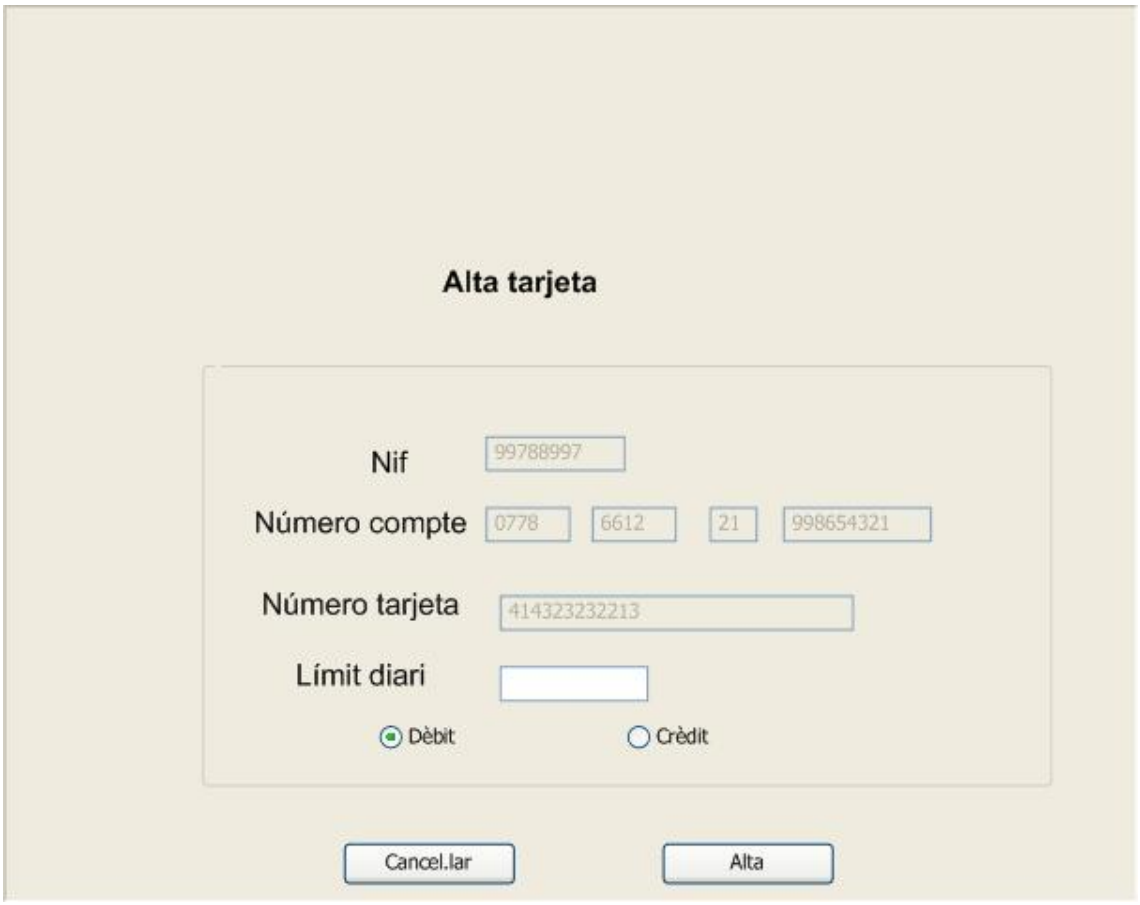
Pantalla Cerca Compte

Pantalla consulta compte



Pantalla esborra compte

Pantalla alta tarjeta

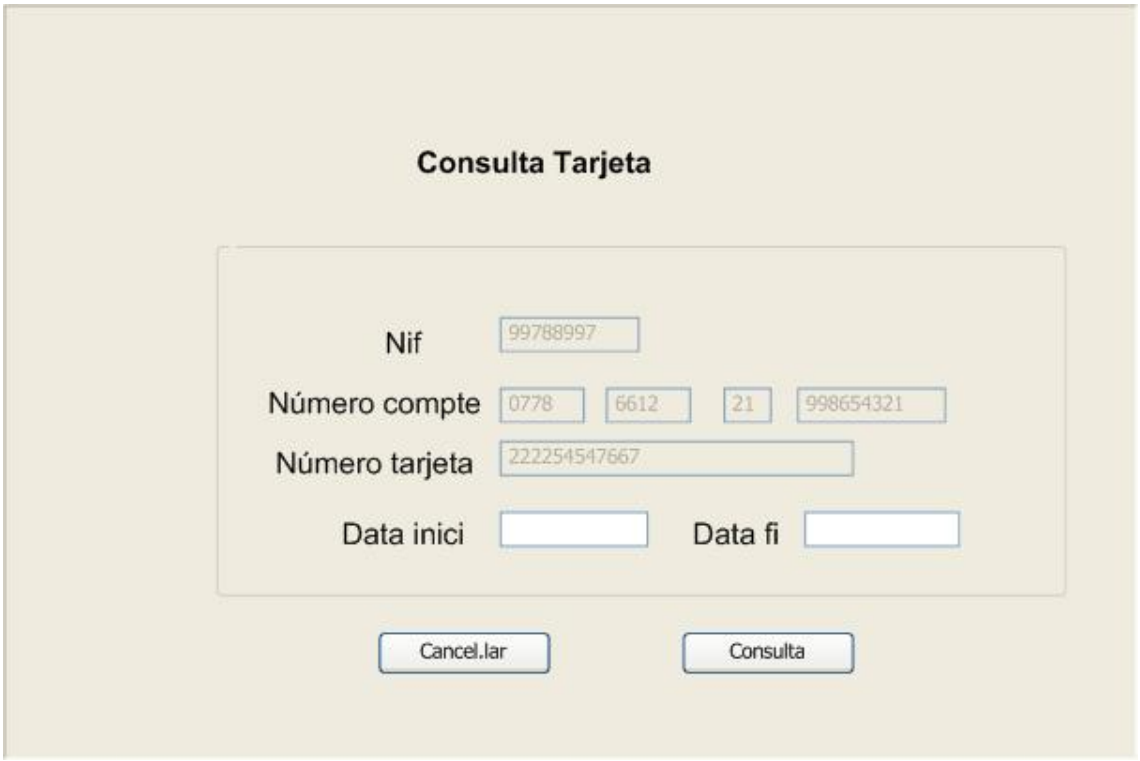


Pantalla modifica tarjeta

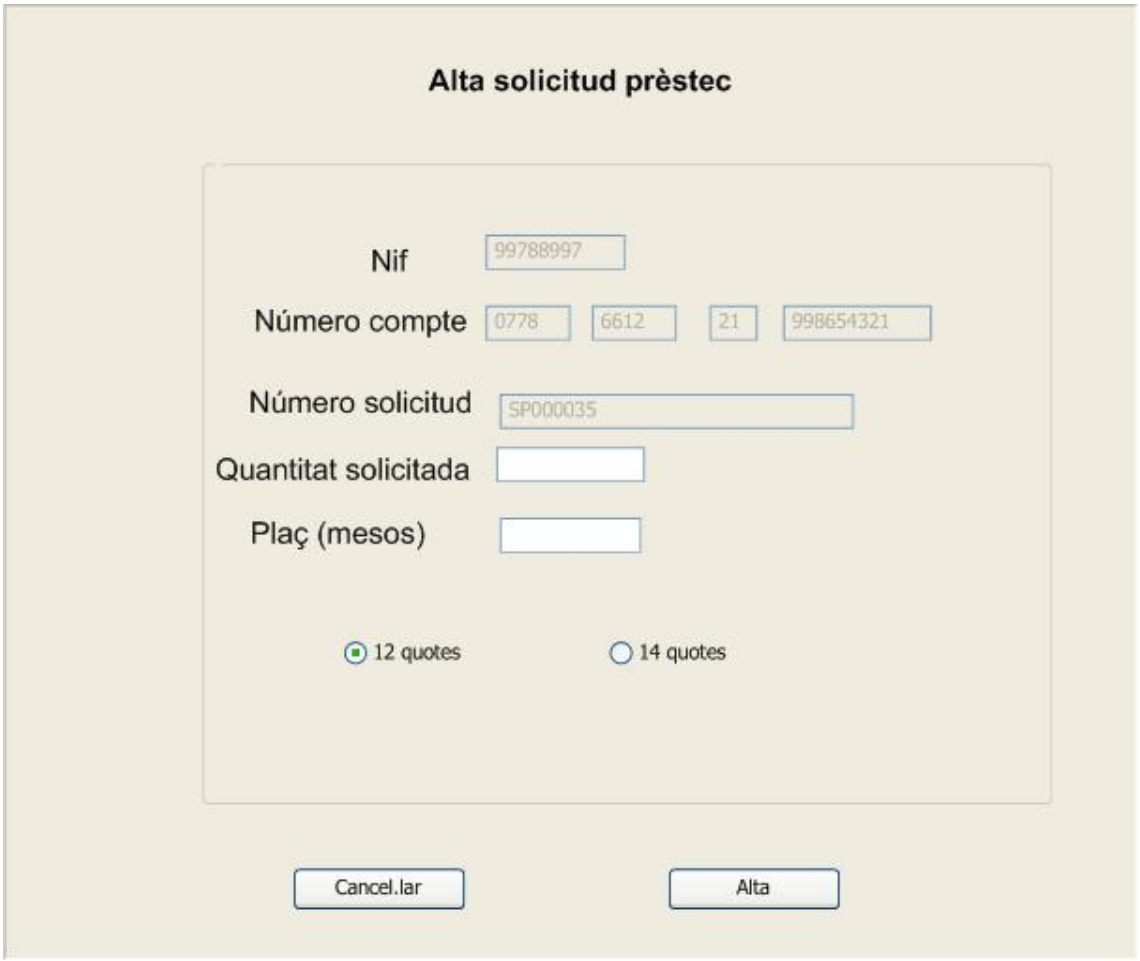
És igual que l'anterior, ja que només es pot modificar el límit.

Pantalla baixa tarjeta

Pantalla consulta tarjeta

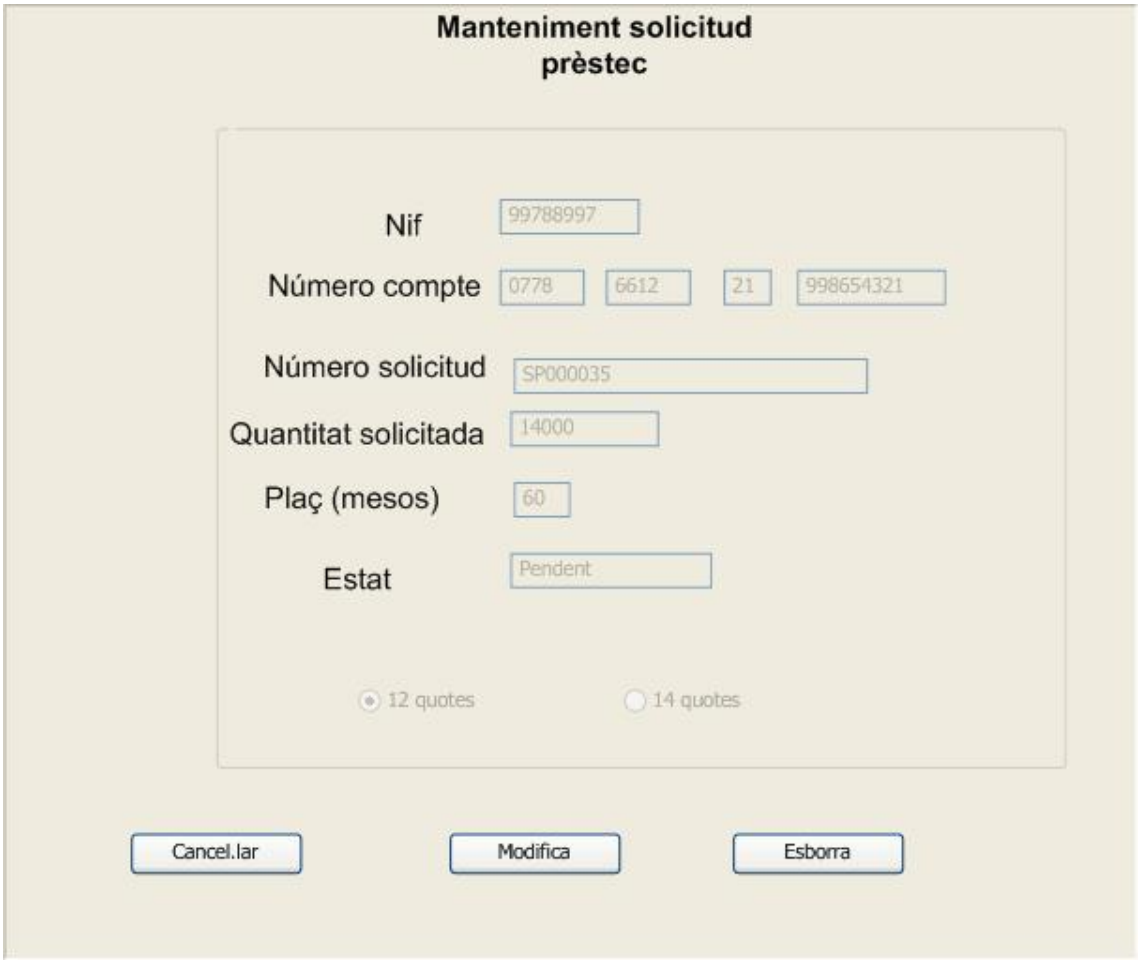


Pantalla alta solicitud prèstec.



Pantalla localització solicitud prèstec.

Pantalla manteniment solicitud.



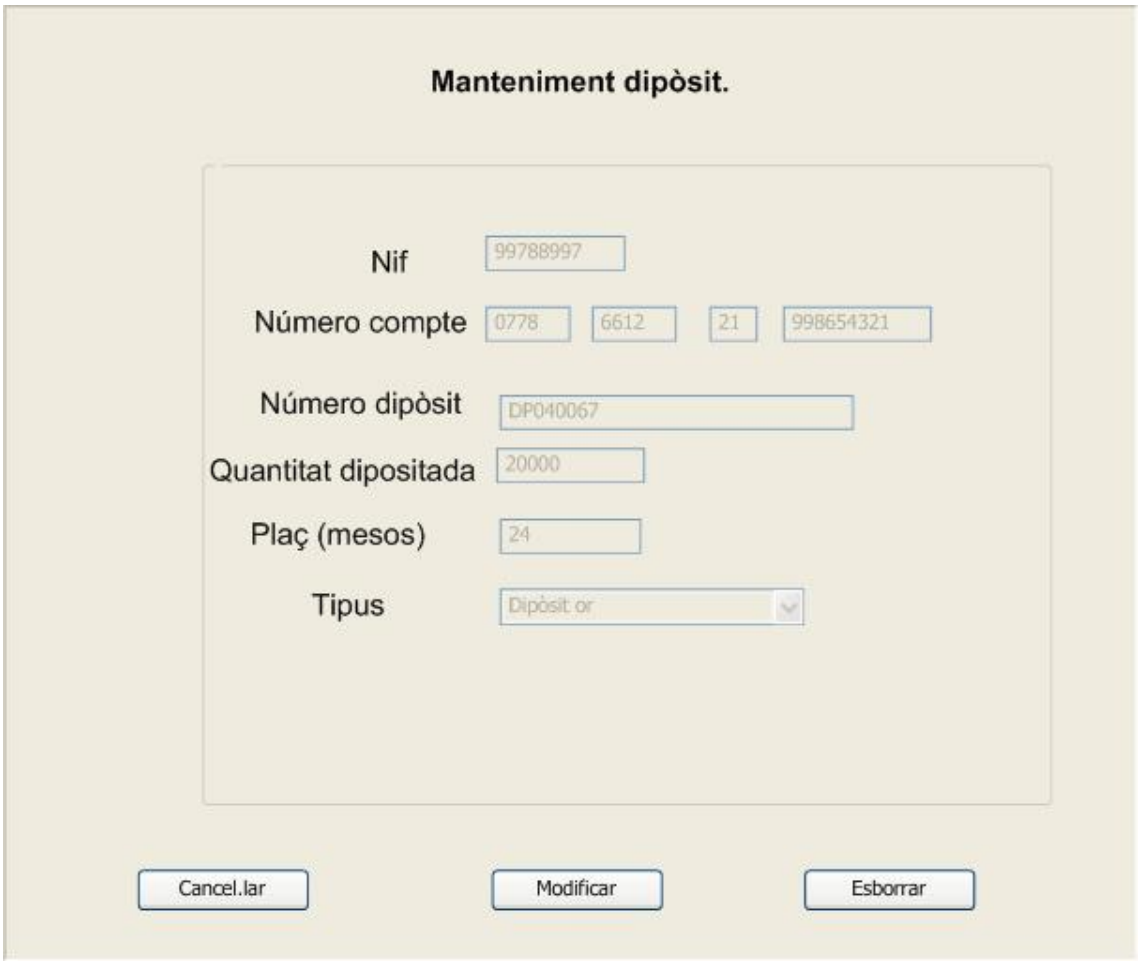
Pantalla modifica solicitud

Igual que l'anterior però amb els camps "quantitat solicitada", "plaç" i "quotes" activats per a poder canviar-los.

Pantalla alta contractació prèstec.

Pantalla cerca dipòsit.

Pantalla manteniment dipòsit.



Pantalla modificació dipòsit

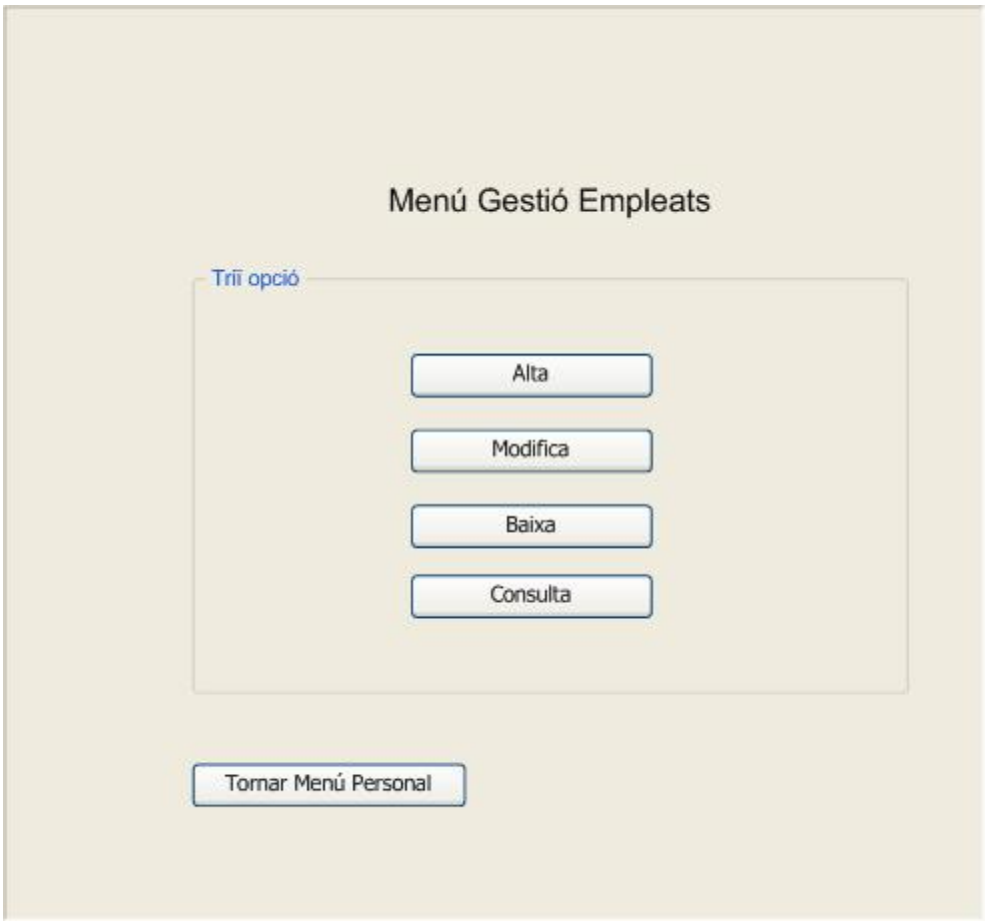
Igual que l'anterior amb el camp plaç activat.

Pantalla ingrés

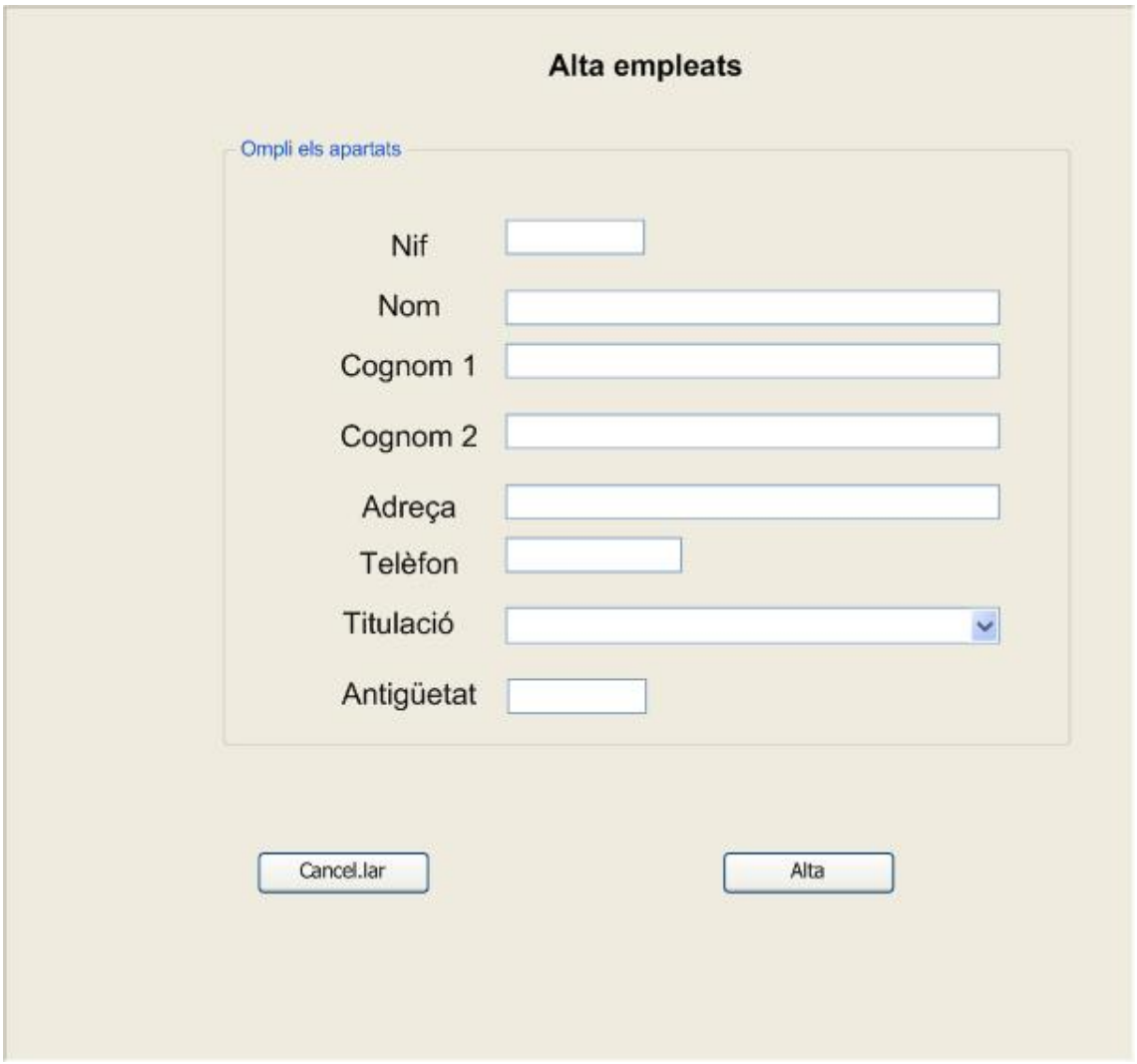
Pantalla reintegrament.

Pantalla menú gestió personal.

Pantalla Empleats.

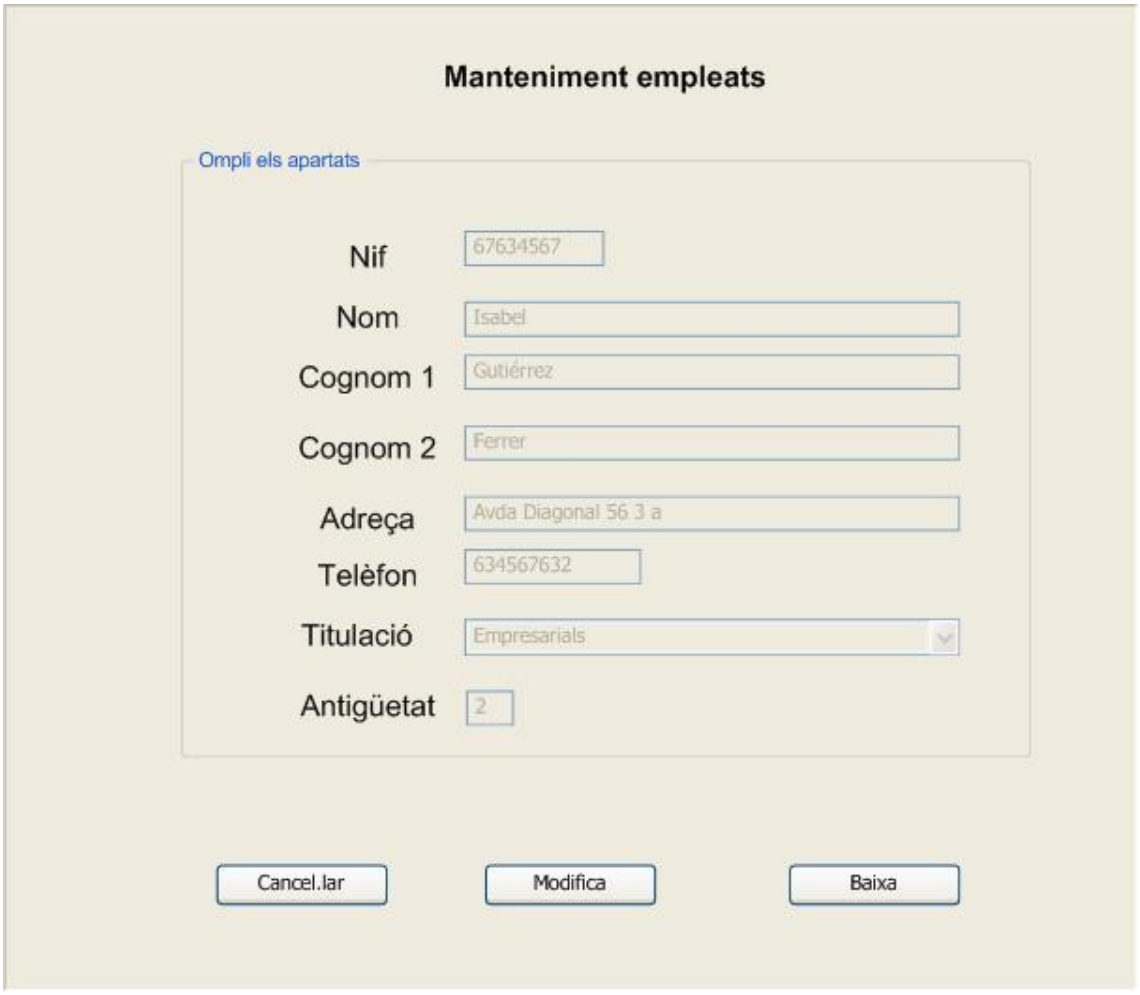


Pantalla alta empleats.



Pantalla cerca empleats.

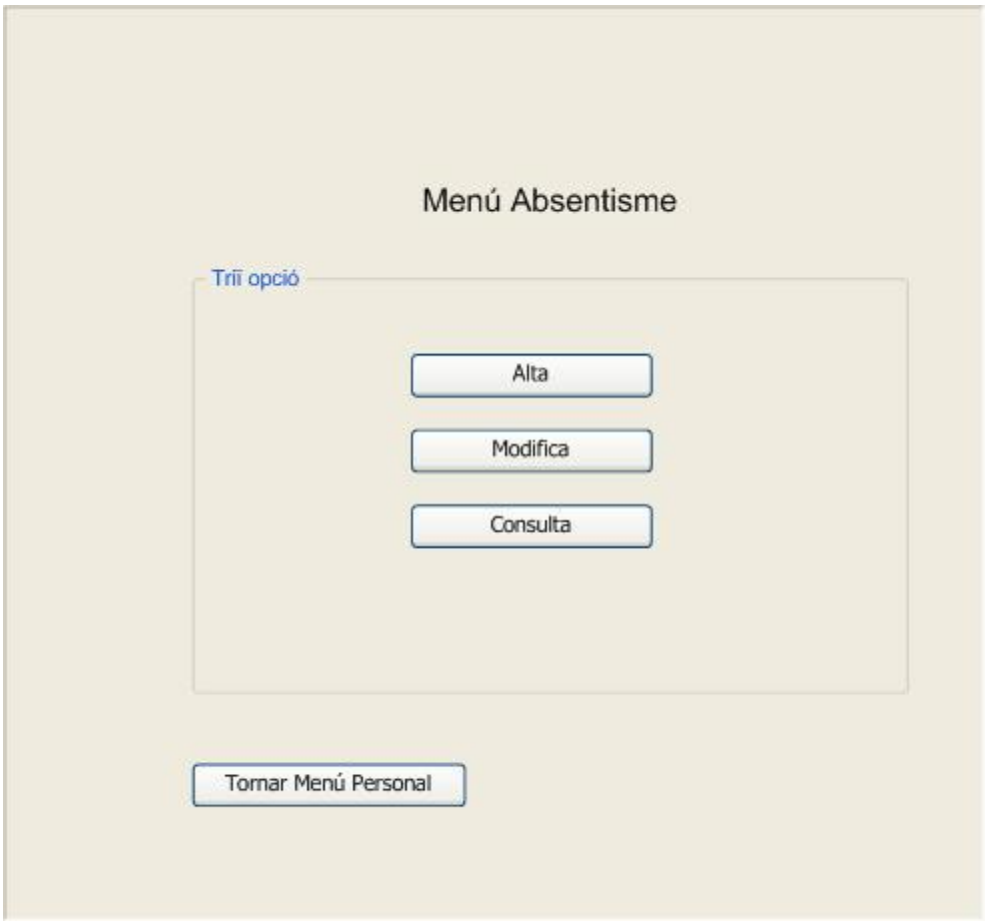
Pantalla manteniment empleat.



Pantalla modifica empleat.

És igual que l'anterior amb els camps activats.

Pantalla absentisme.

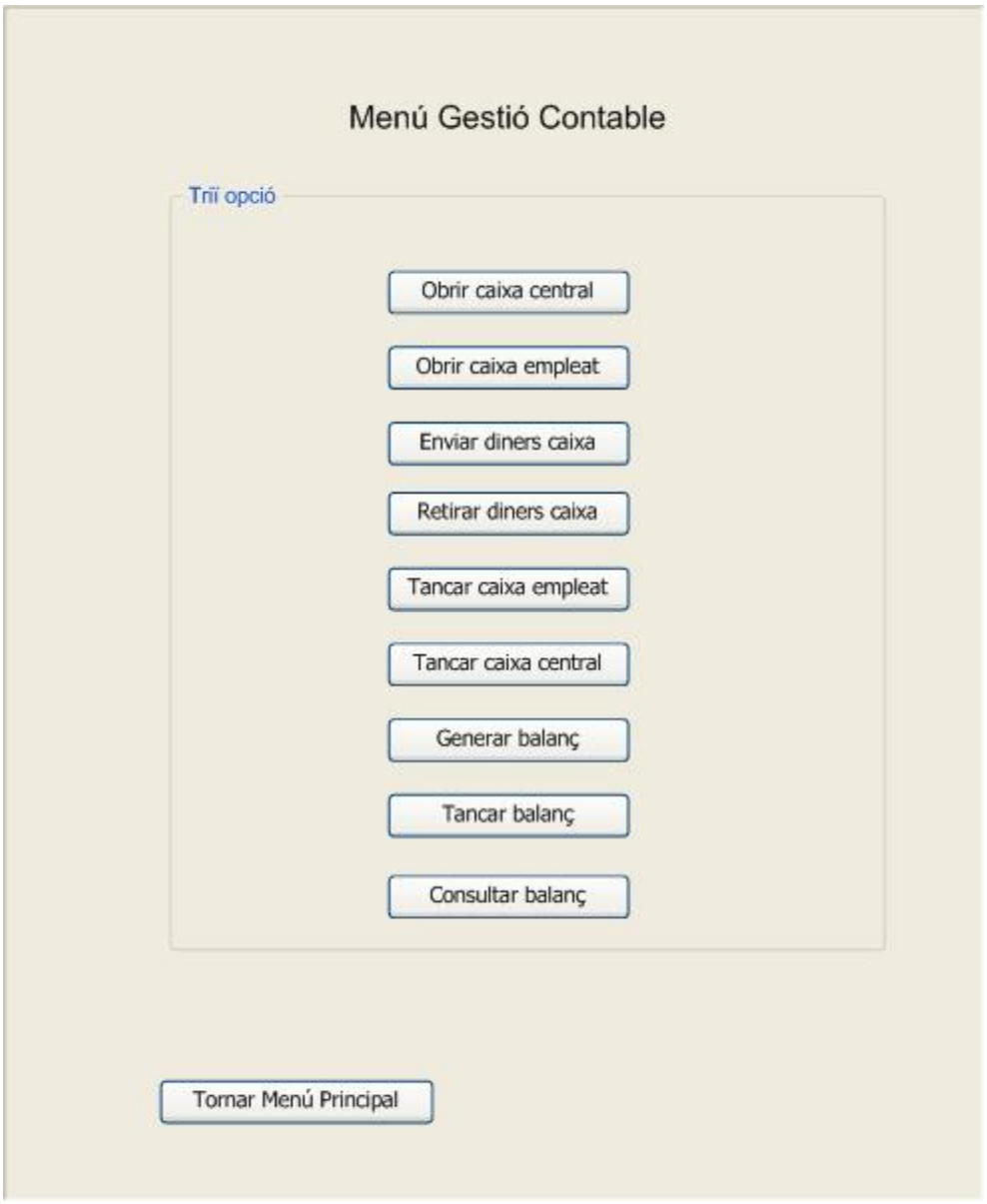


Pantalla alta absentisme.

Pantalla cerca absentisme.

Pantalla modifica absentisme.

Pantalla menu gestió contable.



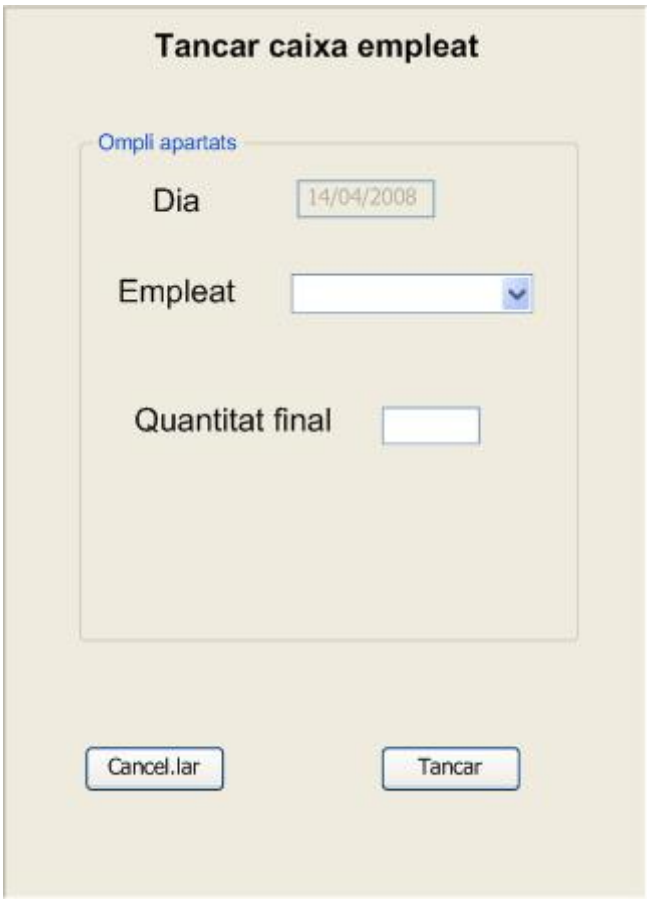
Obrir caixa central.

Pantalla obrir caixa empleat.

Pantalla enviar diners caixa.

Pantalla retirar diners caixa.

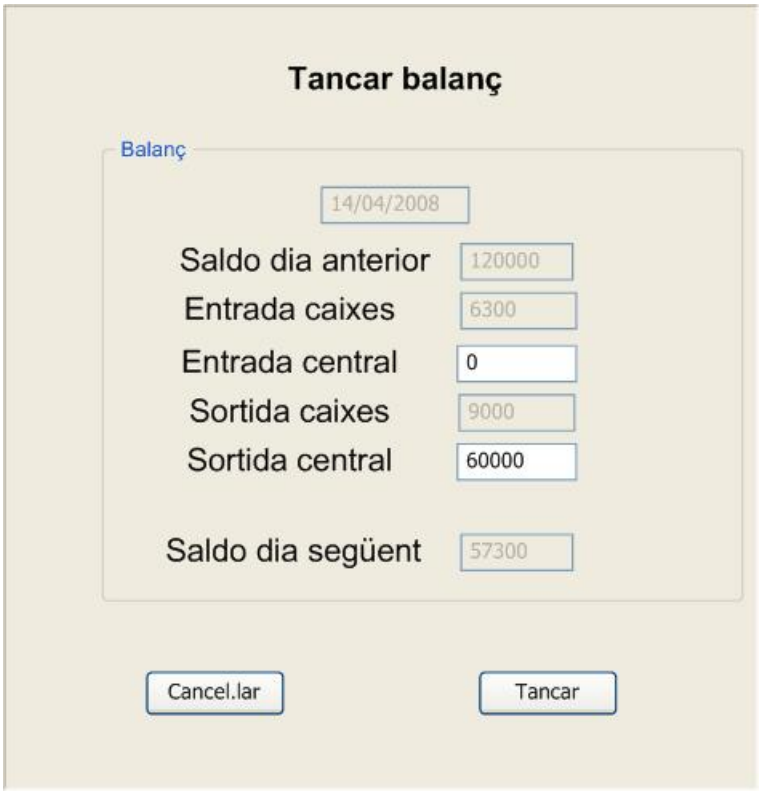
Pantalla tancar caixa empleat.



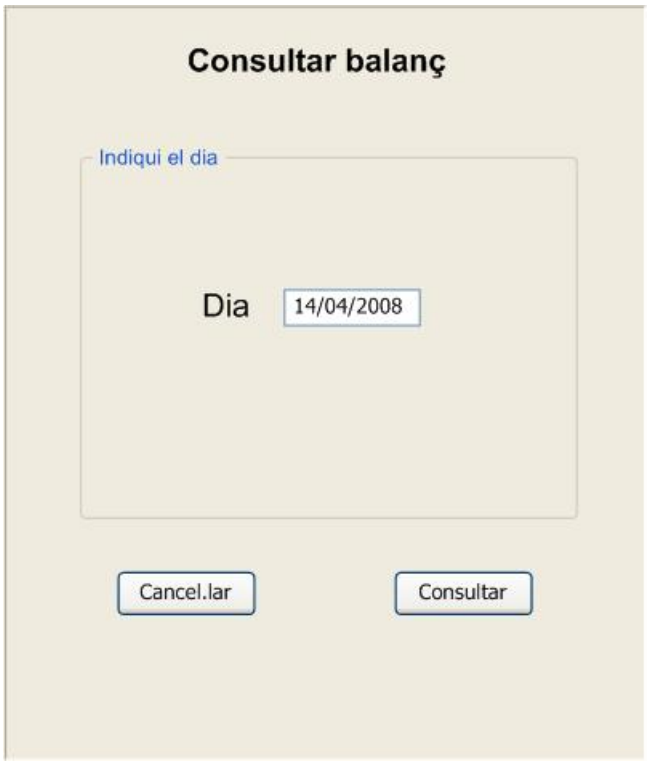
Pantalla tancar caixa central.

Pantalla generar balanç.

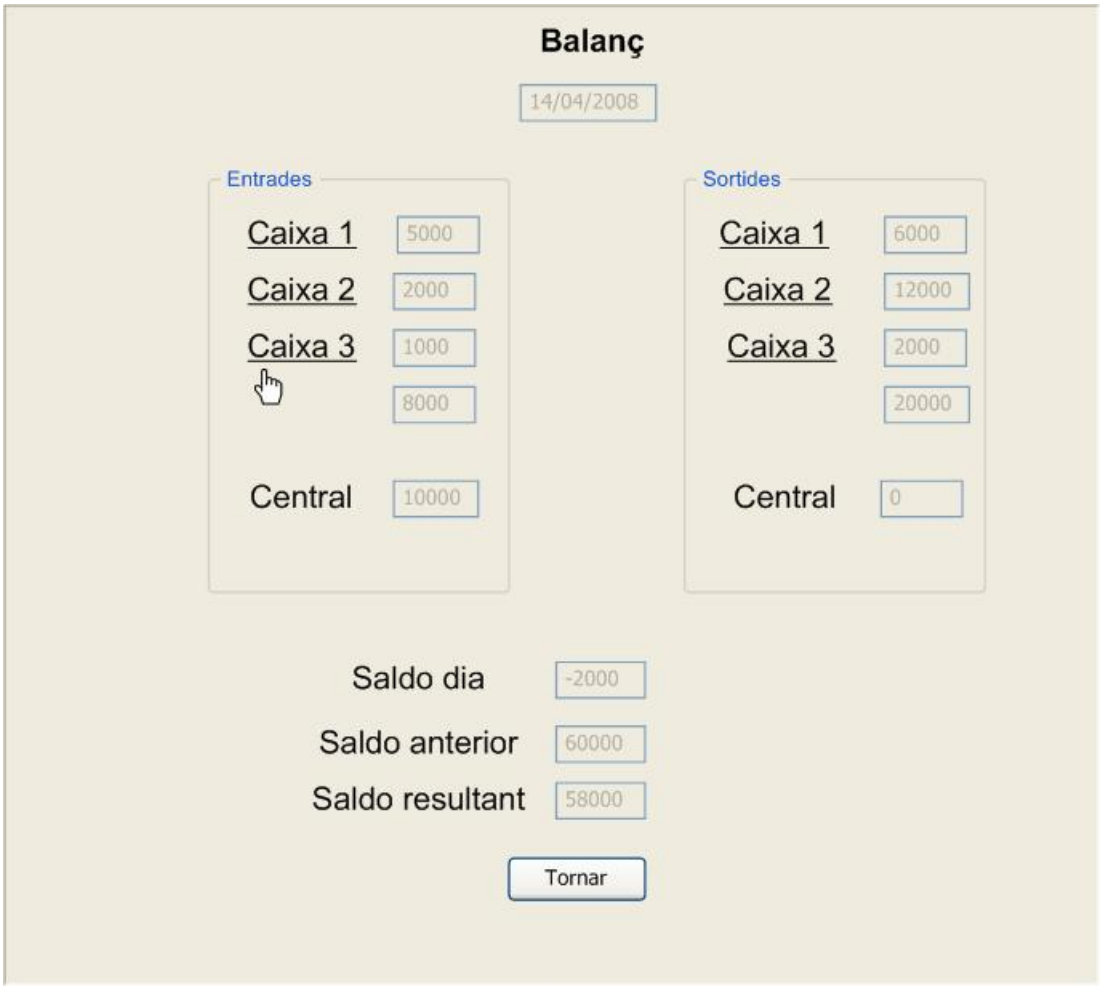
Pantalla tancar balanç.



Pantalla consultar balanç(1)



Pantalla consultar balanç(2)



TREBALL DE FI DE CARRERA (TFC)

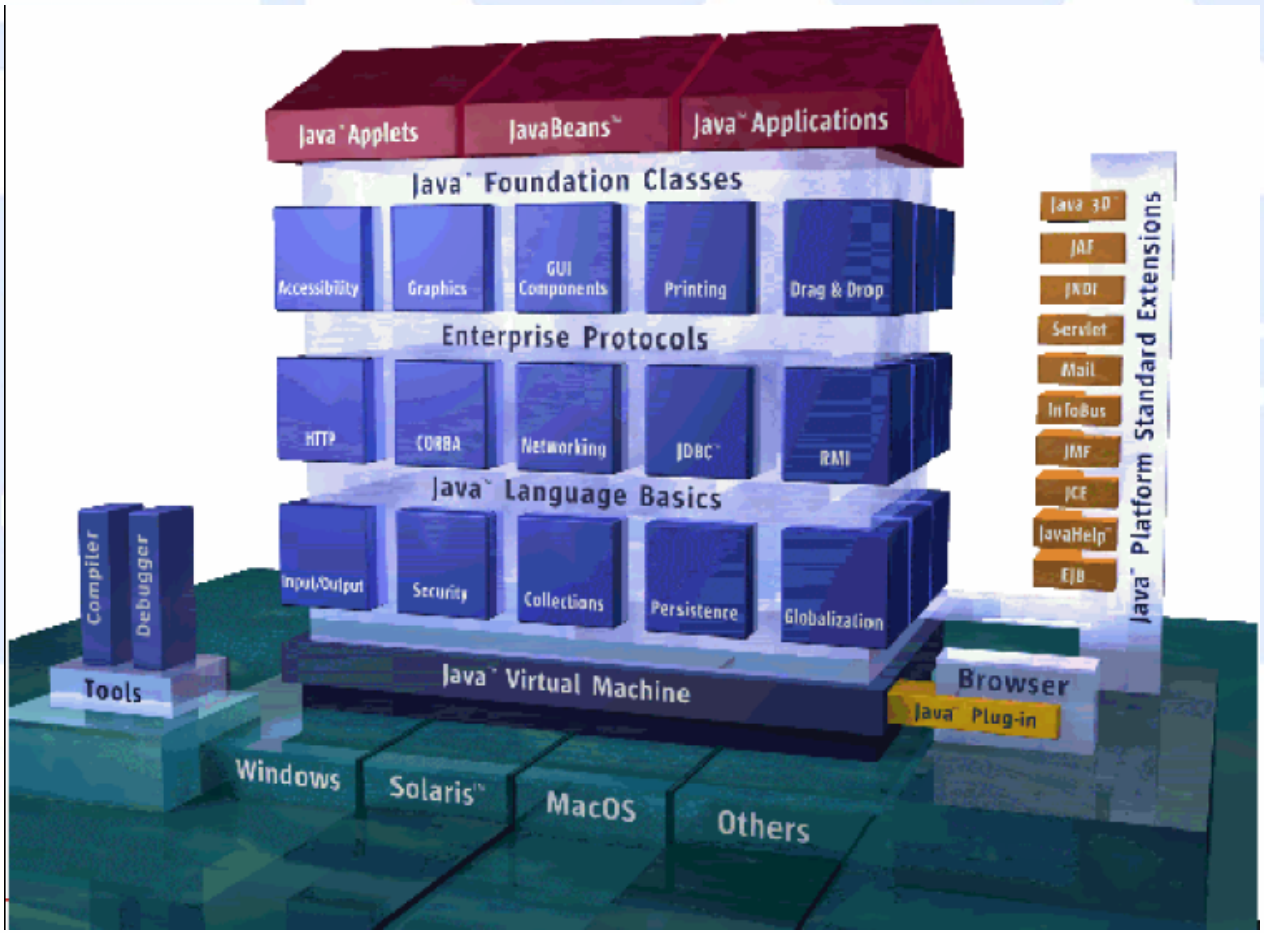
www.uoc.edu

Alumne: Àngel Bustamante Maldonado
ETIG
Consultor: Albert Grau Perisé
25/06/2008

INTRODUCCIÓ

El treball fi de carrera (TFC) pretén ser un treball de síntesi dels coneixements adquirits en diferents assignatures de la carrera d'ETIG.

Jo he triat Arquitectura J2EE.



OBJECTIUS

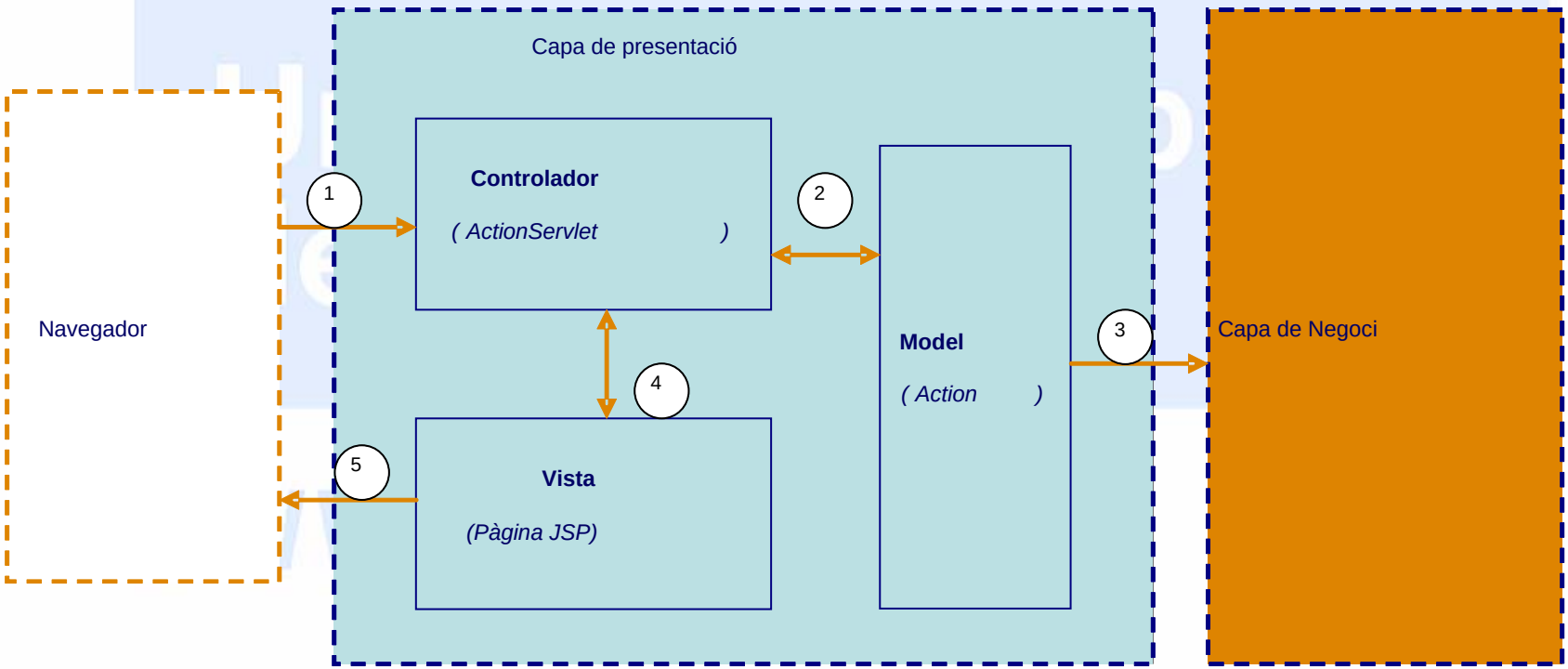
- **Construcció d'una aplicació web a partir de les especificacions d'un client imaginari.**
- **Estudi i utilització del mètode Rational Unified Process, el més habitual actualment en la construcció de software.**
- **Disseny d'una base de dades i implementació del model lògic mitjançant un SGBD punter al mercat com Oracle.**
- **Estudi i utilització de patrons de disseny.**
- **Estudi i utilització del framework Struts.**
- **Estudi i utilització d'un Object Relational Mapping com és Hibernate.**
- **Visió global de l'Arquitectura J2EE.**

Universitat Oberta de Catalunya
www.uoc.edu

DISSENY DE L'ARQUITECTURA

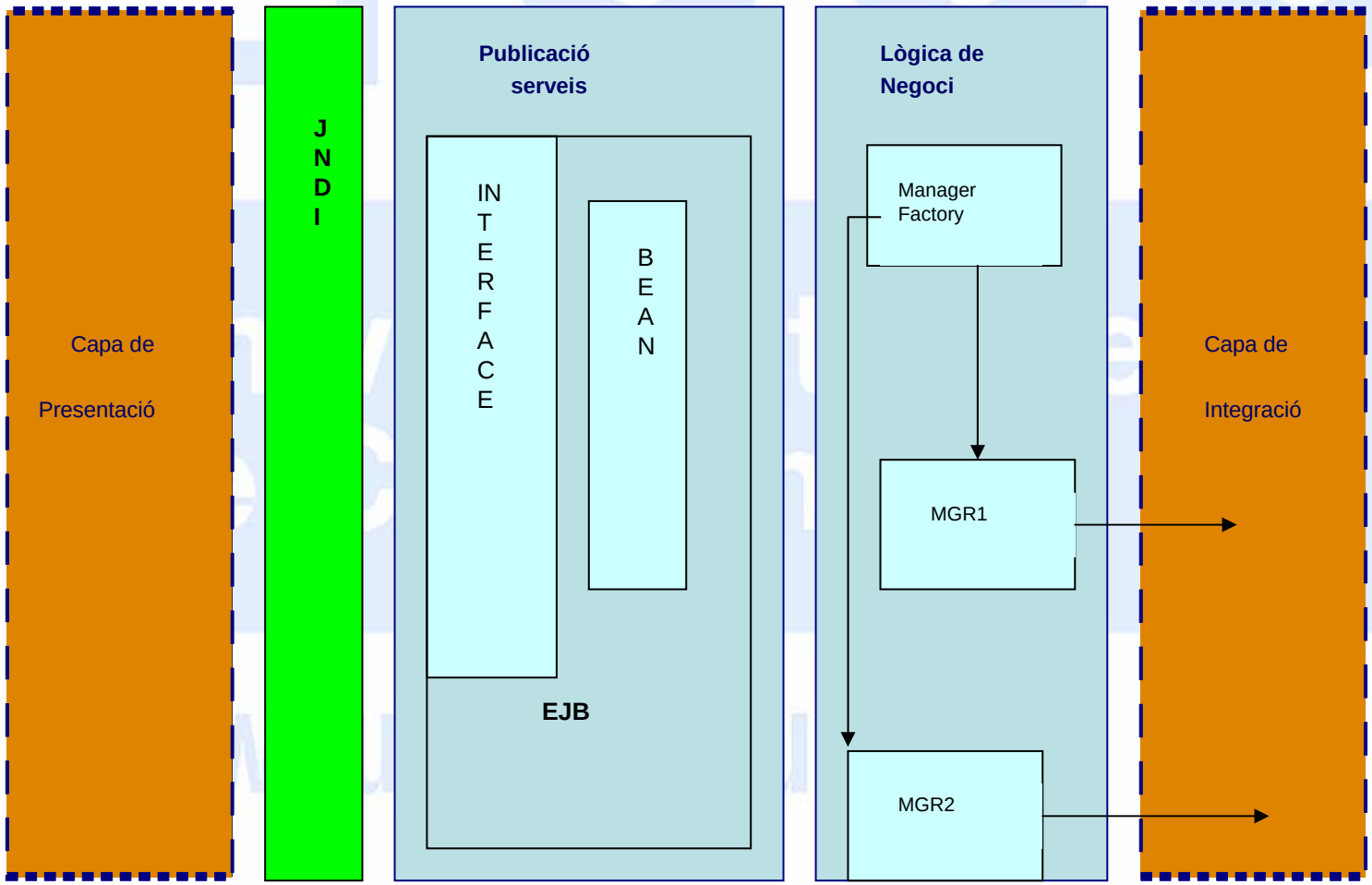
CAPA DE PRESENTACIÓ I CONTROL EN SERVIDOR

Encarregada de rebre les peticions dels usuaris, validar les dades que són enviades en cada petició, decidir a quin servei de la capa de negoci redirigir la petició per que sigui processada i mostrar a l'usuari el resultat de l'operació. Aquesta capa s'encarrega també de controlar el flux de navegació dels usuaris per l'aplicació.



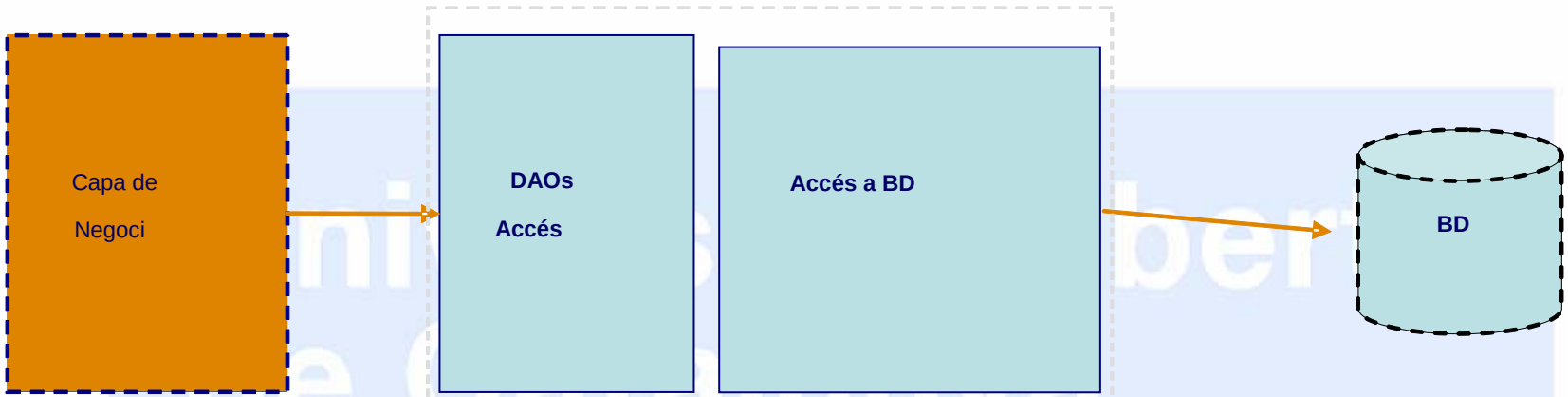
CAPA DE NEGOCI

La capa de negoci del nostre sistema contindrà la lògica de negoci requerida per l'aplicació. Interactuarà amb la capa de presentació per atendre les peticions dels usuaris del sistema i amb la capa d'integració per a recuperar o manipular informació emmagatzemada a la base de dades, o en un futur interactuar amb sistemes externs.



CAPA D'INTEGRACIÓ

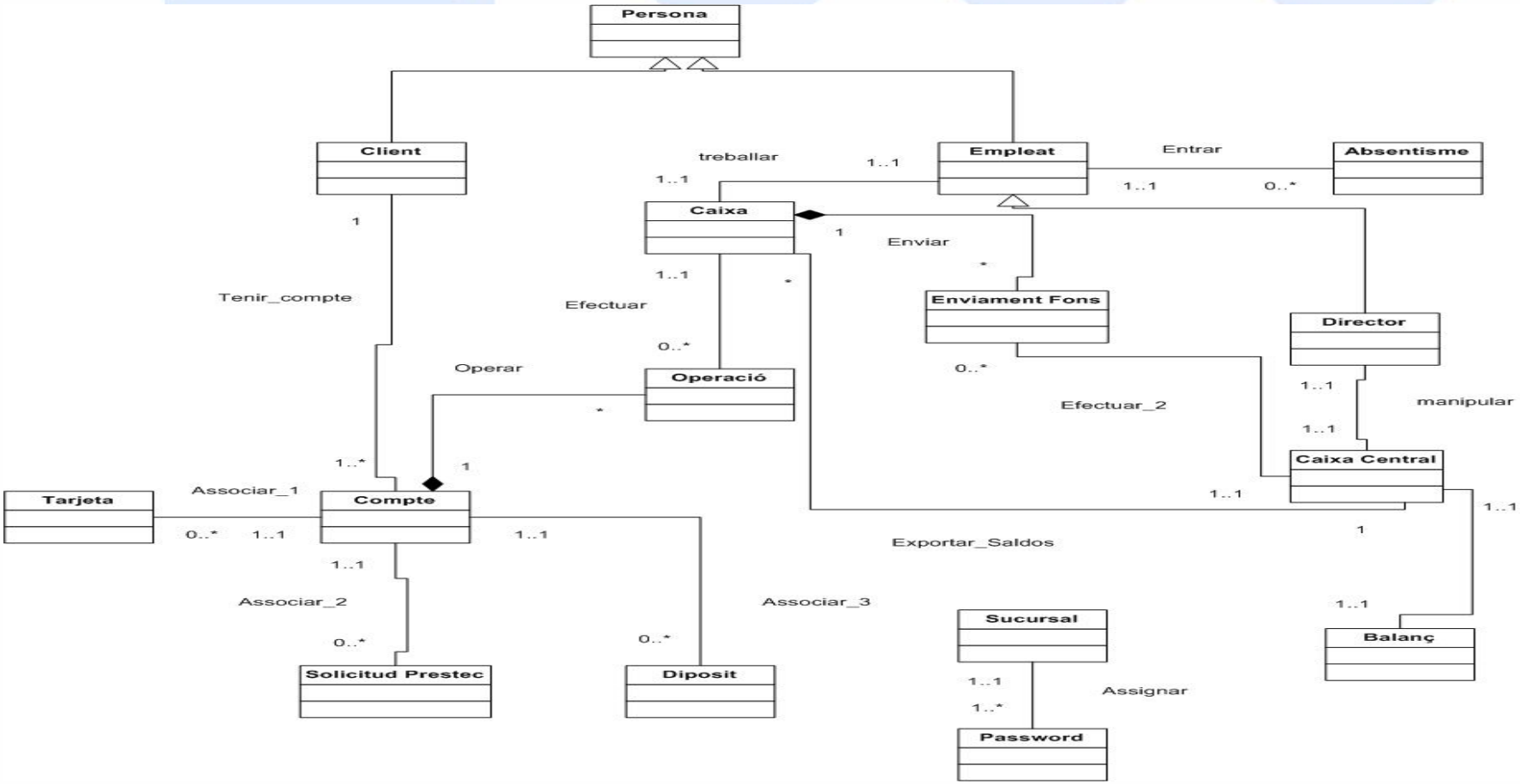
A aquesta capa s'encapsula l'accés a les fonts d'emmagatzemament de tipus persistent propis de la nostra aplicació. Aquesta capa la implementarem fent servir l'ORM Hibernate 3.



www.uoc.edu

DISSENY DE LA PERSISTÈNCIA

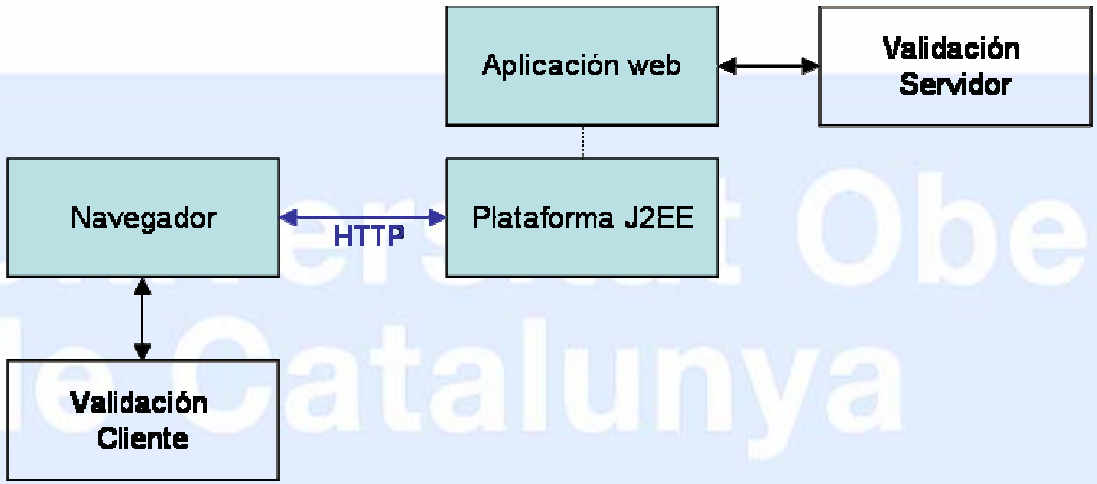
Veiem ara el disseny de la persistència de la nostra aplicació. Aquesta s'implementarà mitjançant el SGBD Oracle10XE.



IMPLEMENTACIÓ
(PATRONS)

CAPA DE PRESENTACIÓ

El procés de validació de les dades en una aplicació web segueix , típicament, una estructura similar a la d'aquí:

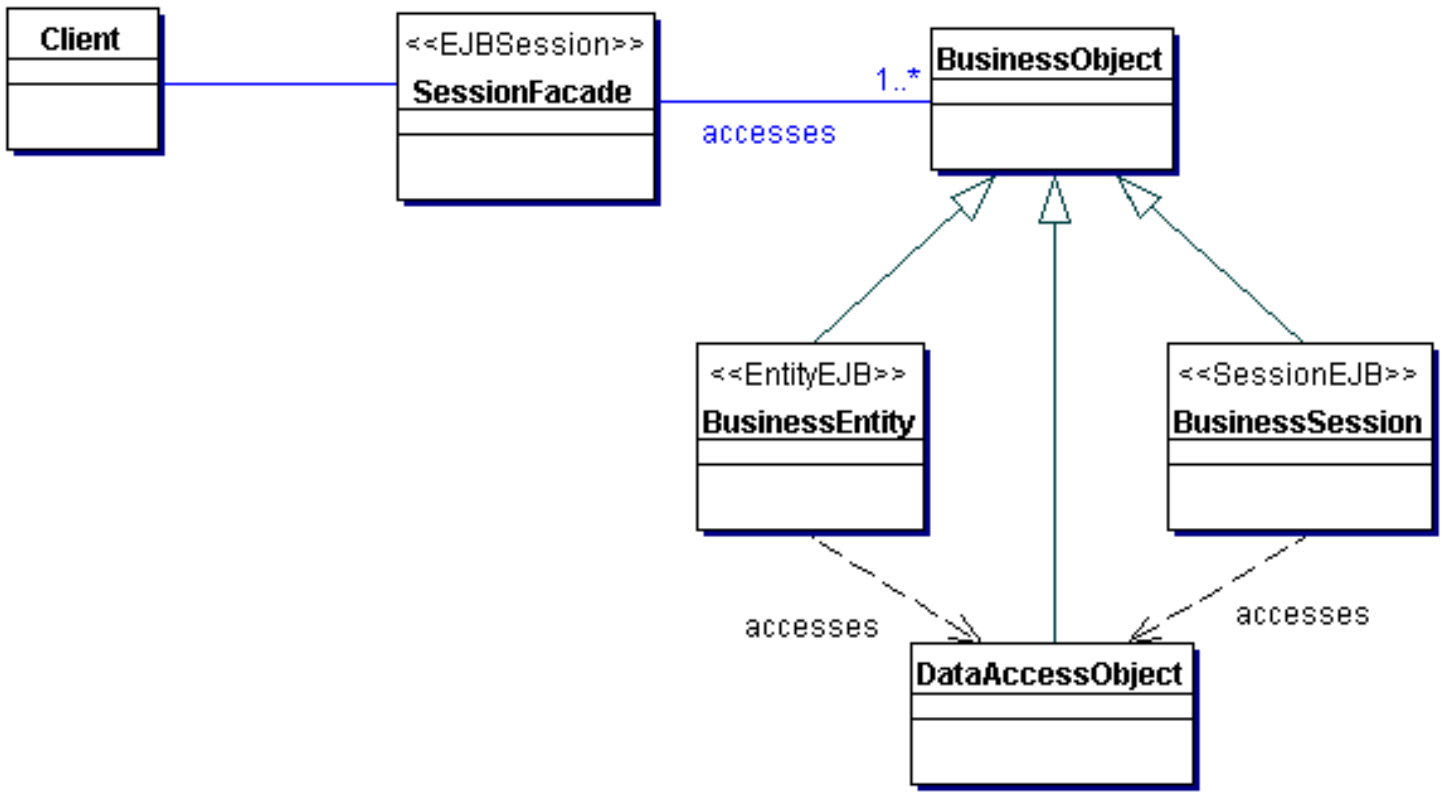


Struts Validator s'encarrega de la validació.

Universitat Oberta de Catalunya
www.uoc.edu

CAPA DE NEGOCI

- **Façade: Aquest patró simplifica l'accés a un conjunt de classes proporcionant una única classe que tots utilitzen per a comunicar-se amb aquest conjunt de classes. L'aventatge principal és que els clients no necessiten coneixer les classes que hi han darrera la classe Facade, podent canviar les classes "ocultades" sense necessitat de canviar els clients.**



- **Interface: Defineix un comportament independent d'on vagi a ser utilitzat. Això permet un desacoplament entre comportament i classe. A la nostra aplicació es pot observar com sempre utilitzem per a fer crides d'objectes, passar com a paràmetre, etc... interfaces, de manera que en un moment donat podem canviar la classe que implemnti aquestes interfaces sense haver de tocar res més.**

- **Configuration: Per a un bon manteniment de l'aplicació és important que totes les dades de configuració es trobin en fitxers, fora del codi propiament dit de l'aplicació. Aquest patró consisteix bàsicament en això, en llegir el màxim de dades de configuracions de fitxers properties. Per a això s'utilitza la classe ConfigurationManager.**

- **ManagerFactory: L'obtenció d'objectes de tipus Manager, es fa a través del Manager Factory. El seu funcionament és com s'explica a continuació: una classe de l'aplicació requereix un Manager per a invocar-lo. Lo normal en aquest cas és que la pròpia classe instanciï l'objecte Manager, l'usi i després el destrueixi. La instanciació d'objectes Manager pot arribar a ser molt costosa en temps, per lo que aquest enfocament no és l'adequat en entorns on el rendiment és important. Per això sorgeix el Manager Factory, els objectes Manager no són instanciats directament desde la classe que els ha d'utilitzar, sino que aquesta classe se'ls demana a unaltra classe, el ManagerFactory, la qual els instancia i els retorna a la classe demandant. En el ManagerFactory s'implementa una estratègia caché, per evitar haver d'instanciar el mateix Factory cada cop que s'ha d'utilitzar.**

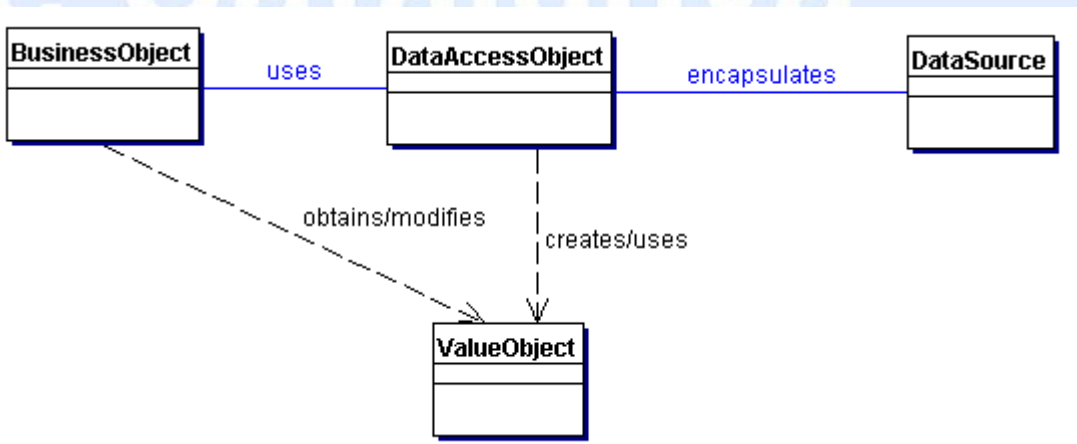
- **Singleton: Aquest patró s'utilitza per a assegurar que una classe te una única instància i proporciona un punt d'accés global a ella. És necessari quan hi ha classes que han de gestionar de manera utilitzada un recurs. Aquest patró és utilitzat a la vegada per la majoria dels altres patrons.**

- **Caché: Aquest patró com el de Singleton és utilitzat com ja hem vist per la majoria dels altres patrons. El seu funcionament és senzill: a l'hora d'instanciar un objecte primer mira si el tenim a la nostra cache (normalment un HashMap), si és així el recuperem, sino el creem.**

- **Value object (VO): Aquest patró té l'objectiu d'encapsular els objectes de negoci, de manera que en comptes de intercanviar informació fent servir els seus atributs ho fem intercanviant objectes, d'aquesta manera es redueix la crides remotes, habituals en l'arquitectura J2EE. Aquest patró a més és molt convenient tenint en compte que utilitzarem un ORM com Hibernate.**

- **Data Acces Object (DAO): Realment aquest patró hauria d'estar a l'apartat de la capa d'integració. El posem aquí perque estiguin explicats tots al mateix apartat.**

  El DAO implementa el mecanisme d'accés requerit per treballar amb la font de dades. Aquesta font de dades pot ser un emmagatzament persistent com una RDMBS, un servei extern com un intercanvi B2B, etc...Lo que fa aquest patró és ocultar la implementació de la font de dades als seus clients. Esencialment, el DAO actua com un adaptador entre el component i la font de dades.



CAPA D'INTEGRACIÓ

- La programació orientada a objectes i les bases de dades relacionals es basen en dos paradigmes diferents. El model relacional tracta amb relacions i conjunts. En canvi el paradigma orientat a objectes tracta amb objectes i associacions entre ells. Hi ha una desavenència entre aquests dos paradigmes, la també anomenada diferència objecte – relacional. Aquesta diferència s'amplia molt rapidament si tens grans models d'objectes.

- És degut a aquesta diferència per lo que apareixen els mapejadors objecte-relacional (ORM) com és el cas d'Hibernate.

- Hibernate es basa en dos fitxers de configuració, hibernate.cfg.xml per a la configuració general de l'aplicació (conexió jdbc, datasources, ...) i un per cada classe entitat NomClasse.hbm.xml a on es fa el "mapping" entre classes i taules de la base de dades relacional.

Universitat Oberta de Catalunya
www.uoc.edu

CONCLUSIONS

- **Les conclusions que trec després d'haver fet aquest projecte són totes positives; m'ha permés enfrontar-me a molts aspectes de la construcció d'una aplicació i m'ha permés conèixer la arquitectura J2EE.**

- **També he entrat al món dels framework (amb Struts i Hibernate) que evolucionen molt rapidament, fins al punt que fins i tot (els esmentats Struts i Hibernate) s'estan quedant desfasats respecte per exemple lels Java Server Faces (JSF) i, Spring, EJB3 Entity,..**

- **Considero que la arquitectura J2EE té un gran futur en el camp de les aplicacions empresarials ja que dona resposta a totes les demandes empresarials sent capaç d'integrar-se amb diferents tecnologies sense gaire esforç.**

- **Considero que la arquitectura J2EE té un gran futur en el camp de les aplicacions empresarials ja que dona resposta a totes les demandes empresarials sent capaç d'integrar-se amb diferents tecnologies sense gaire esforç.**

Universitat Oberta de Catalunya
www.uoc.edu