PROJECTE FINAL DE GRAU

Àrea de treball : Java EE

CONTROL DE PRODUCCIÓ

MAQUINES FLEXOGRÀFIQUES

MEMÒRIA PROJECTE

Grau en Enginyeria informàtica

The worldwide university

Autor: Marc Segura Valls

Consultor: Albert Grau Perisé

Índex

Índex figures

1. Introducció

He escollit fer el projecte de final de Grau amb la plataforma Java EE per varies raons.Una perquè en el treball de final de carrera de l'enginyeria de sistemes vaig escollir la tecnologia .NET, que em va permetre conèixer a fons una de les tecnologies de sistemes distribuïts comercials més utilitzades actualment. Una altre ha estat que durant tota la carrera hem estat fent la majoria de les pràctiques amb la plataforma Java, per tant he considerat que després de tants any implementant petits projectes amb aquesta plataforma, ara era el moment de crear des de zero un projecte amb aquesta plataforma tant ben acceptada per la comunitat de del sector educatiu actual.També crec que és molt interessant conèixer a fons Java al tractar-se d'una plataforma oberta que té la capacitat de córrer sobre qualsevol sistema operatiu.Finalment, perquè em sembla una plataforma molt versàtil i robusta del qual n'he gaudit molt al llarg de la carrera.

1.1. Justificació del projecte

Ja fa uns quants anys que el món empresarial s'està beneficiant de les eines que ofereix el camp de la enginyeria informàtica, amb la finalitat de tenir un major i ràpid control en la seva gestió, però no tots els sectors ho han implementat tant aviat.Per exemple el sector industrial, i em refereixo a la part on tenen la màquina que produeix, s'ha tardat a implementar-hi programari que interactuï amb els operaris de producció i la maquinaria automatitzada. Aquests tipus d'aplicacions poden ajudar molt a treure un millor rendiment i eficiència de les cadenes de producció, gràcies a tenir una traçabilitat del producte que s'està fabricant.

Jo he escollit el sector de les arts gràfiques, concretament el de les impressores flexogràfiques (consultar www.comexigroup.com i http://es.wikipedia.org/wiki/Flexograf%C3%ADa) . És un sector molt antic que ja ha sofert molts grans canvis evolutius al llarg de la seva vida, però aquestes màquines sempre han estat aïllades dels departaments de gestió i finances.Ara, aprofitant que ells components electrònics que les controlen ja utilitzen protocols estàndards de comunicació com ethernet, ha arribat el moment de treure'n tot el suc possible i així facilitar i agilitzar la gestió del producte que estan creant.

1.2. Objectiu del projecte

L'objectiu que m'he marcat per aquest projecte és el de desenvolupar una aplicació Java EE que treballi dins un entorn de components i sistemes distribuïts, capaç de controlar la producció i la traçabilitat d'una (o més) màquina impressora industrial flexogràfica. Serà una aplicació web que residirà en un servidor d'aplicacions connectat a la xarxa interna de l'empresa. Aquesta aplicació, també haurà d'estar pensada perquè en un futur molt immediat interactuï amb una aplicació mòbil mitjançant serveis web.A més també estarà oberta a treballar amb una aplicació client de tipus escriptori.

A més, com a objectiu, m'he marcat treure el màxim de rendiment de les tecnologies principals que giren al voltant de la plataforma Java EE (JSF 2, EJB , JPA, etc...), i implementar-les en el projecte d'una manera estructurada , entenedora, elegant i que faciliti el treball en equip en el seu manteniment i creixement.

També conèixer i aprendre un nou gestor de base de dades amb bona reputació dins el sector , que no sigui ni informix, MYSQL i Microsoft SQL server que ja hi he treballat en anteriors projectes. Per això per la part de permanència de dades he optat per el gestor de base de dades obert PostgreSQL.

Un altre objectiu que m'he marcat, ha estat el de implementar un sistema multi idioma que gestioni tant la part dinàmica com la de persistència de dades.

1.3. Enfocament i mètode seguit

Aquest projecte segueix el model de cicle de vida clàssic, també anomenat en cascada. La base d'aquest model és que cada etapa del projecte es realitza quan acaba l'anterior.

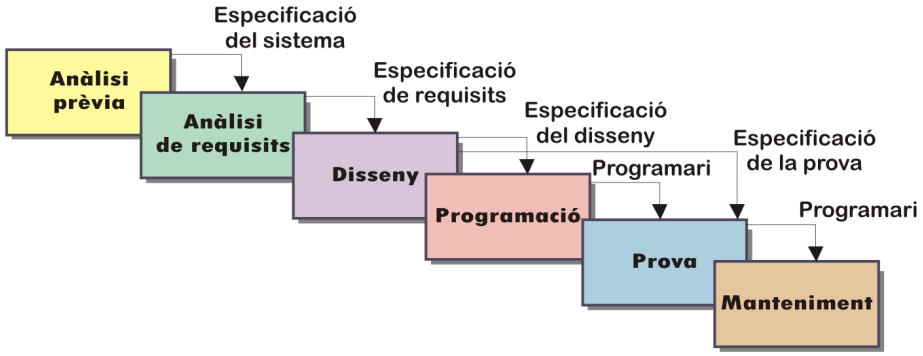

Figura 1. Cicle de vida en cascada

1.4. Planificació inicial i real

El transcurs del projecte s'ha dividit en quatre fases importants: Pla de treball i preparació de recursos, anàlisi i disseny, implementació i la fase final que és la creació d'aquesta memòria i la presentació virtual.Totes aquestes fases es varen temporitzar, i han constat de subtasques molt ben definides per tal de no trobar-nos amb imprevistos i així poder complir la data d'entrega .Cal remarcar que les quatre diferents fases coincideixen amb les tres Pacs i la elaboració de la memòria i el vídeo presentació.

FASE 1 : ***Pla de treball i preparació de recursos***: *(del 16/09/2013 al 2/10/2013)*

- Detallar un document on hi consti la descripció del projecte , els objectius, les funcionalitats, estudi de la idoneïtat del projecte i la planificació, i que és aquest mateix document.
- Descarregar tot el programari necessari per realitzar el projecte i instal·lar-l'ho.
- Escollir els principals frameworks que utilitzarem per fer el projecte (JSF, , EJB's, JPA Entity, ...)
- Comprar o descarregar bibliografia recomanada i tutorials sobre els diferents programes i frameworks a utilitzar. També començar a explorar webs especialitzades en Java EE i els seus principals frameworks, també del SGBD PostgresSQL, i registrar-me a fòrums per quan s'han tingut dubtes, poder-los exposar ràpidament.
- Començar a llegir materials i conèixer l'entorn Java EE.

FASE 2 : ***Anàlisi i disseny:*** *(del 3/10/2013 al 7/11/2013)*

- Detallar els requeriments funcionals de l'aplicació i del disseny tècnic amb les tècniques que s'han après a les assignatures sobre d'Enginyeria del programari: recollida de documentació de requisits, especificació dels casos d'ús , dels diagrames d'activitat i de seqüència més rellevants,també especificació de les classes d'anàlisi ,especificació de la base de dades a crear,etc...

- Dissenyar les especificacions com també fer el disseny arquitectònic que seria la configuració de la xarxa i els subsistemes.També pensar amb la reutilització de codi, i així poder aprofitar classes ja provades que siguin útils i ajudin a guanyar temps.

- Disseny de la persistència de dades , que seria tot el disseny relacionat amb la base de dades: persistència amb base de dades orientada a objectes i model de la base de dades relacional.

- Disseny de la interfície gràfica d'usuari a partir de prototips de les pantalles principals.

- Paral·lelament, estudiar la plataforma Java EE , els frameworks escollits i les altres tecnologies.

FASE 3 : ***Implementació:*** *(del 8/11/2013 al 16/12/2013)*

- Implementar l'aplicació a partir de la documentació que es va generar a la fase anterior dins la Java EE.

- Desplegar l'aplicació en un servidor (en el meu cas he escollit el JBOSS) , fer totes les proves convenients per assegurar el funcionament correcte de totes les funcionalitats, i resoldre tots els problemes trobats.

- Adaptar i fer proves per implementar serveis web per tal de poder crear una aplicació client mòbil de tipus android o iphone.

- Crear un manual d'instal·lació de tot el sistema amb els passos detallats a seguir.

FASE 4 : ***Memòria i Presentació virtual:*** *(del 17/12/2013 al 13/01/2013)*

- Realitzar una memòria del projecte complerta i detallada (Aquest document)

- Crear una presentació virtual sintetitzada que serveixi de suport a la defensa del projecte davant del tribunal.També serà important que en aquesta presentació es vegi una simulació per demostrar el funcionament correcte de l'aplicació.

	Task Name	Duración	Comienzo	Fin	Predecesoras
1	***Treball Final de Grau***	86 días	lun 16/09/13	lun 13/01/14	
2	**- Pla de treball**	13 días	lun 16/09/13	mié 02/10/13	
3	Pensar i fer esborrany sobre la proposta	3 días	lun 16/09/13	mié 18/09/13	
4	Buscar manuals, webs i tutorials sobre Java EE	3 días	jue 19/09/13	lun 23/09/13	3
5	Descàrrega I instal.lació programari	7 días	mar 24/09/13	mié 02/10/13	4
6	Eleboració document pla de treball	7 días	lun 23/09/13	mar 01/10/13	
7	Entrega Pla de treball (PAC1)	0 días	mié 02/10/13	mié 02/10/13	6;5
8	**-Anàlisi I disseny**	26 días	jue 03/10/13	jue 07/11/13	7
9	Estudi Java EE (EJB, JPA, JBOSS, SPRING,..)	23 días	mié 02/10/13	vie 01/11/13	
10	Recollida documentació dels requisits	3 días	jue 03/10/13	lun 07/10/13	2
11	Especificació BBDD	2 días	mar 08/10/13	mié 09/10/13	10
12	Especificació classes d'anàlisi	3 días	jue 10/10/13	lun 14/10/13	11
13	Especificació casos d'ús I altres	3 días	mar 15/10/13	jue 17/10/13	12
14	Disseny BBDD	4 días	vie 18/10/13	mié 23/10/13	13
15	Disseny estructura estàtica (Diagrama classes)	4 días	jue 24/10/13	mar 29/10/13	14
16	Disseny estructura dinàmica (Casos d'ús, etc..)	4 días	mié 30/10/13	lun 04/11/13	15
17	Disseny interfície usuari	2 días	mar 05/11/13	mié 06/11/13	16
18	Entrega documentació (PAC2)	1 día	jue 07/11/13	jue 07/11/13	17;9
19	**-Implementació**	27 días	vie 08/11/13	lun 16/12/13	18
20	Implementació de l'aplicació a la plataforma Java EE	23 días	vie 08/11/13	mar 10/12/13	8
21	Desplegament, provar i correcció d'errors	3 días	mié 11/12/13	vie 13/12/13	20
22	Crear Manual d'instal-lació	1 día	lun 16/12/13	lun 16/12/13	21
23	Entrega documentació (PAC3)	0 días	lun 16/12/13	lun 16/12/13	22
24	**-Memòria I presentació virtual**	20 días	mar 17/12/13	lun 13/01/14	23
25	Realitzar memòria	16 días	mar 17/12/13	mar 07/01/14	19
26	Realitzar presentació virtual	12 días	jue 26/12/13	vie 10/01/14	
27	Entrega documentació	1 día	lun 13/01/14	lun 13/01/14	26
28	**-TRIBUNAL**	9 días	mar 14/01/14	vie 24/01/14	1

Figura 2. Dates clau de les tasques del projecte

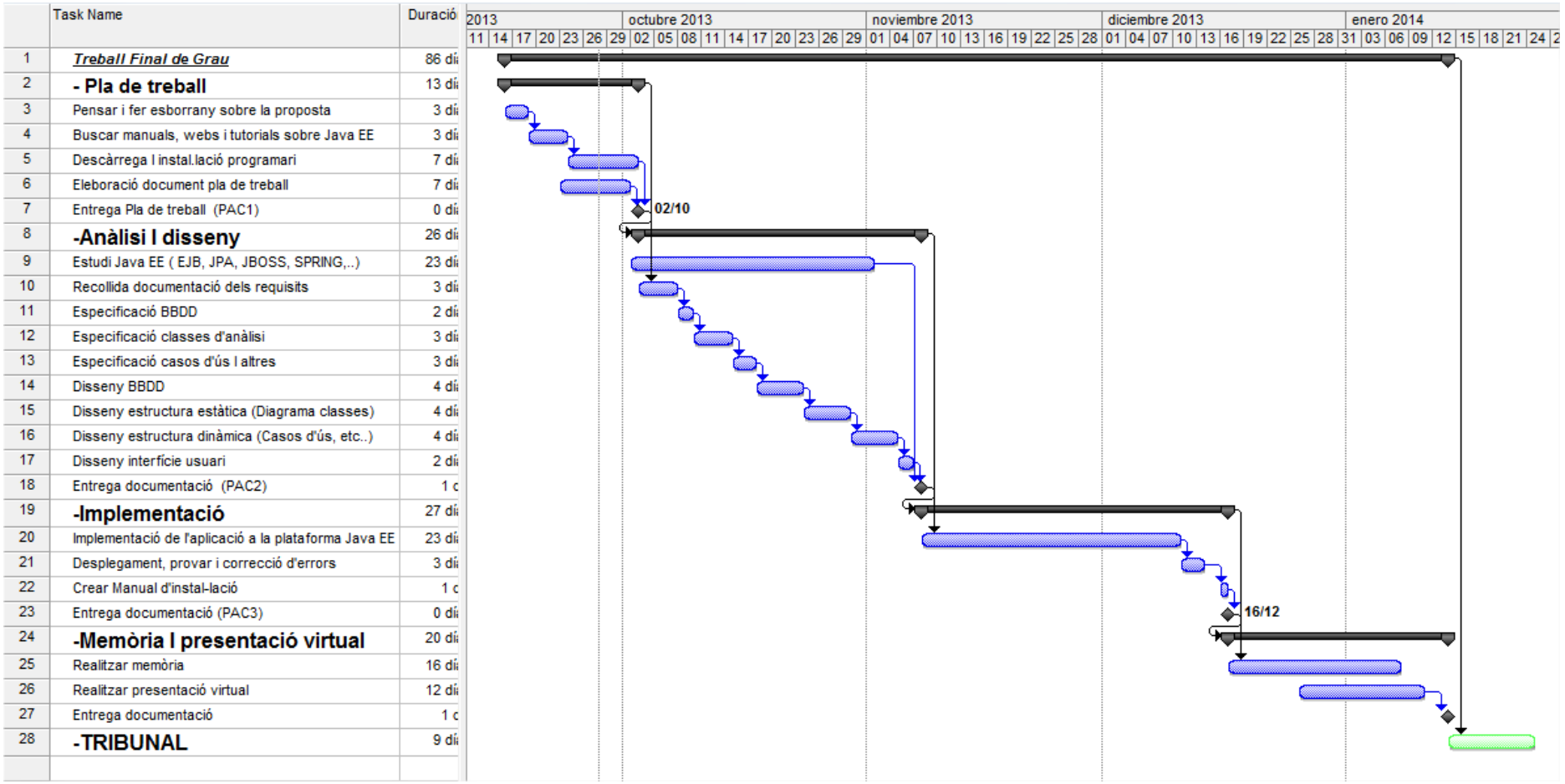

Figura 3. Diagrama de Gantt del projecte

La planificació real practicament no ha diferit de la inicial, ja que per respectar les dates claus era necessari seguir la planificació de les tasques marcades a la primera fase del projecte. Una de les tasques que apareixia a la planificació i no s'ha pogut dur completament a terme, ha estat la de l'aplicació mòbil, ja que la corba d'aprenentatge de tots els frameworks de la plataforma Java EE és bastant lenta i s'hi ha hagut de dedicar més hores de les previstes.A més deixar l'aplicació web ben depurada amb totes les funcionalitats, també ha portat més hores de les que s'esperaven.

De totes maneres, el projecte ja té un servei web programat apunt per fer-hi proves des d'una aplicació de dispositiu mòbil, per tant això seria una següent fase immediata d'aquest projecte.

1.5. Eines i tecnologies emprades

Per dur a terme un projecte d'aquest tipus, calen un gran ventall d'eines que ofereixen tant plataformes obertes, com de pagament.

Jo he escollit les següents:

- ✓ **Eclipse** com a IDE, concretament la versió Helios descarregat a la pàgina oficial www.eclipse.org.
- ✓ **Java JDK** , com a mínim la versió JDK 1.6 versió, instal.lat i configurat a la màquina on es desplegarà l'aplicació Web, o sigui on hi ha el servidor web.
- ✓ **Apache Ant** , que és una eina molt útil en la programació per realització de tasques mecàniques i repetitives, en aquest cas per la fase de compilació i construcció (fitxer build.xml). És una eina creada en llenguatge Java i no depèn de les ordres del Shell de cada sistema operatiu, ja que es basa en arxius de configuració XML i classes Java per la realitzacióde diferents tasques.S'ha descarregat el programa des de http://ant.apache.org. S'ha utilitzat la versió la versió Ant 1.8.3: http://apache.rediris.es/ant/binaries/apache-ant-1.8.3-bin.zip .
- ✓ **PostgreSQL** com a sistema de bases de dades, ja que diuen els que en saben que és un dels motors de base de dades de codi obert més potent a dia d'avui. S'ha descarregat en el link http://www.postgresql.org i s'ha he treballat amb la versió PostgreSQL v. 9.1.3-1.
- ✓ **JBOSS** ha estat el servidor d'aplicacions escollit.Està desenvolupat en Java i per tant pot córrer en qualsevol plataforma que suporti Java 1,6 o superior.
- ✓ **Connector Java JDBC de PostgreSQL** per què el JBOSS pugui treballar amb el postgreSQL. Es troba a http://jdbc.postgresql.org , des de on es baixa JDBC Driver per a Java http://jdbc.postgresql.org/download/postgresql-9.1-901.jdbc4.jar .
- ✓ **EJB's** (Enterprise java bean), un component de la part servidora, gestionat pel contenidor i pensat per la construcció modular d'aplicacions distribuides. L'especificació EJB és una de les moltes API's de Java de la Plataforma Java Enterprise Edition. EJB encapsula la lògica de negoci d'una aplicació.

- ✓ **JPA** (Java Persistence API) que és un entorn de la plataforma java que permet tractar les entitats de la base de dades com a classes i objectes, i d'aquesta manera podem treure més profit de la plataforma java EE , ja que serà el servidor d'aplicacions qui s'encarregarà d'accedir a la part de persistència de dades i així aquesta, quedarà aïllada de l'aplicació.

- ✓ **JSF 2** (Java Server Faces) , i que realment són pàgines xhtml, és framework que ve amb l'especificació JavaEE. És utilitzat per crear la interfície i juntament amb el controlador Faces Servlet i les acciones creades per Managed Bean, aquesta interactuarà amb la lògica de l'aplicació.

- ✓ **HTML + CSS** per tal de crear i ordenar les pàgines de l'aplicació Web (xhtml), s'han utilitzat etiquetes de HTML (HiperText Markup Language), i per donar un agradable i dinàmic disseny aquestes pàgines he programat una fulla d'estils en cascada CSS (Cascading Style Sheets).

- ✓ **MagicDraw 17 SE** que és la eina CASE (Computer Aided Software Enginering) per crear els diagrames UML , tant estàtics com els dinàmics.

- ✓ **Microsoft Word 2007 ,** utilitzat per la elaboració de tota la documentació.

- ✓ **Corel Draw X3,** programa de dibuix vectorial utilitzat per fer la majoria de il·lustracions de la documentació, inclòs els diagrames de casos d'ús, diagrama de classes, etc...

- ✓ **Microsoft Project 2007 ,** per la creació del diagrama de Gantt a la part de planificació.

- ✓ **Microsoft PowerPoint**, per la creació d'una part de la presentació virtual del projecte.

- ✓ **Audicity**, és un programa de captura i edició de so, que m'ha permès obtenir una bona qualitat de so a la presentació.

- ✓ **Camtasia Studio 7,** per fer les captures en viu de la demostració de l'aplicació i de les diapositives del Powerpoint més l'edició final de so i vídeo per la presentació virtual.

- ✓ **Virtual Box** . És un programari que emula un ordinador i pot executar programes com si fos un ordinador real.S'ha escollit treballar amb màquina Virtual de la plataforma Virtual Box i sobre un sistema operatiu Windows XP. Treballar amb una màquina virtual per crear un projecte per un entorn Java EE ha estat una elecció prèviament reflexionada , ja que el nostre entorn està format per vàries eines i aplicacions. A més per poder connectar la base de dades amb el servidor haurem de configurar fitxers XML i instal·lar el connector JBDC. També haurem d'instal·lar i configurar llibreries i plugins en el Eclipse per poder fer tasques concretes i preparar i configurar el desplegament de l'aplicació en el servidor. Són molts els passos de instal·lació i configuració, i una màquina virtual ens assegura poder replicar de manera ràpida i poder recuperar aviat el sistema en cas d'alguna falla o emergència. A més ens permetrà poder utilitzar el sistema que hem creat en més d'un computador indistintament de la versió de Windows.

1.6. Productes obtinguts

Durant la realització d'aquest projecte s'han generat els productes que es mostren a la taula següent.

Producte	Descripció
Pla de treball (PAC 1)	Document inicial que conté informació com: descripció del projecte, l'especificació de les funcionalitats, l'arquitectura computacional, la planificació i altre informació d'interès.
Anàlisi (PAC 2)	Aquest document presenta una descripció molt més detallada, conté un anàlisi de les funcionalitats més exhaustiu on es detallen els processos més importants a implementar mitjançant diagrames de casos d'ús i altres diagrames UML.
Disseny (PAC 2)	Documentació que fa referència a l'arquitectura del sistema, el disseny de la base de dades i també el disseny del diagrama de classes i els diagrames de components distribuïts,.
Implementació (PAC 3)	Un comprimit d'un conjunt de subcarpetes amb el codi font de la implementació de l'aplicació que s'ha creat amb Eclipse.També l'script per la creació de la base de dades amb les insercions per poder provar l'aplicació.
Manuals (PAC 3)	Manual d'instal·lació i configuració l'entorn on haurà de córrer l'aplicació, on també explica com desplegar-la en el JBOSS des del Eclipse.
Memòria (PAC 4)	Aquest document, que és la memòria de tot el projecte.
Presentació virtual (PAC 4)	Vídeo d'uns vint minuts de durada que és un complement de la memòria , que conté els punts més destacats i a més inclou una demostració en viu de l'aplicació i que ajuda a defensar el projecte davant d'un tribunal.

2. Anàlisi

En aquest apartat hem de conèixer tot el que ha de fer la nostra aplicació i a partir d'aquí identificar els requeriments funcionals i els no funcionals.Els primers es refereixen a totes les funcionalitats que ha de proporcionar el sistema i les dades que ha de conèixer, com les manipula i quines ha de desar. Els no funcionals (també anomenats de qualitat) , són el conjunt de requisits que no es refereixen a cap funcionalitat en concret, sinó que són els de producte.Són qualitats esperades del sistema com ara la usabilitat, fiabilitat rendiment o manteniment.

2.1. Descripció del projecte

Aquest control de producció es basa en la creació d'unes fitxes per cada treball que es realitzarà sobre la impressora en qüestió.Aquestes fitxes guardaran una sèrie de valors relacionats amb els paràmetres de treball , com temperatures de secatge, tensions del material a imprimir, llargada dels clixés, espessor material on s'imprimeix, números de tinters (capçals) colors de tintes, viscositats, referència anilox (transportadors de tinta) , nom de l'operador que l'ha creat, etc...

A partir d'aquestes fitxes es creen comandes, per tant treball pot tenir un número il·limitat de comandes.Per cada una d'aquestes comandes, es guardarà una sèrie de informació com els metres totals, metres no qualitat, metres qualitat, número de bobines, nom de l'operador etc...A més també guardarà els temps de parades de producció i les causes, per tal de fer una estadística acurada i un estudi per ajudar a millorar la producció reduint el temps d'aquestes parades.

Aquestes fitxes es podran recuperar per quan s'hagi de tornar a fer un nou tiratge del treball en qüestió (comanda) .

Aquest sistema d'entrada estarà compost per quatre subsistemes:

- Subsistema de connexió:

  Es tracta d'un sistema d'autenticació i/o autorització que mitjançant nom usuari i contrasenya que entraran els usuaris, obtindran l'accés només on els hi sigui permès en funció del seu rol en el sistema, així doncs un operador només podrà accedir a les funcions del sistema d'operació, un cap o coordinador a totes les funcions dels subsistemes d'estadística, i un administrador en el subsistema de manteniment.En aquest subsistema l'usuari també podrà modificar la seva contrasenya.

- Subsistema d'Operació:

  És el que utilitzaran els operadors de les impressores per crear les fitxes, mantenir-les actualitzades (modificació només d'alguns paràmetres) i crear comandes a partir d'aquestes fitxes.

- Subsistema de Consulta i Estadística :

  És el que principalment utilitzaran els caps i coordinadors de planta per consultar la producció de les comandes de cada treball. Podran consultar els metres bons, metres de no qualitat, temps de parada màquina, causes, etc...

  A més podran fer servir funcions d'estadística que ajudin a conèixer els costos de producció, coneixent-ne per exemple els metres de no qualitat d'una determinada comanda , de totes les comandes d'un mes (interval de temps) determinat, etc...

- Subsistema Manteniment:

  Des d'aquest subsistema el personal administratiu donarà d'alta als usuaris, en modificarà les dades o bé els donarà de baixa.També podrà llistar tots els usuaris. En un futur immediat, també es gestionaran els clients amb les mateixes funcions

2.2. Requisits funcionals

Els actors que interactuen amb la nostra aplicació són : administrador, coordinador o cap i operador màquina.

Veiem els requeriments funcionals principals segons dels quatre subsistemes:

2.2.1 Funcionalitats subsistema de connexió

- ***Connexió al sistema***: Quan qualsevol usuari vulgui accedir al sistema, aquest mostrarà dos camps buits on l'usuari haurà d'entrar el seu nom d'usuari i la seva contrasenya. També mostrarà un polsador per poder enviar les dades al sistema.Si les dades són correctes, llavors el sistema verifica quin tipus de usuari és i mostrarà per la interfície les funcions que aquest pot utilitzar.Encas contrari mostrarà un missatge d'avís.

- ***Canvi de contrasenya:*** En un moment donant per qüestions de seguretat un usuari voldrà modificar la seva contrasenya.Així quan un usuari accedeix al sistema, a la part inferior de la pantalla principal d'autenticació, es mostra una opció on l'usuari podrà escollir "canvi de contrasenya".Si l'escull, el sistema l'hi mostra una nova pantalla on hi ha d'entrar el nom usuari, la contrasenya actual i la nova contrasenya dues vegades.Si les dades han estat acceptades i no es mostra cap missatge d'error, el sistema executa el canvi i enviarà a l'usuari de nou a la pantalla principal.

2.2.2 Funcionalitats subsistema d'operació

- ***Crear nova fitxa (nou treball)***: Un operador, dins del sistema escull la opció crear fitxa nova i aquest l'hi mostra una pantalla on l'hi demana introduir el nom i totes les dades. En el nostre projecte es poden triar fins a quatre tinters, però s'ha de tenir en compte que una impressora flexogràfica en pot tenir més, només s'haurien d'afegir en una petita actualització.

- ***Crear nova comanda***: Un cop creada la fitxa d'un treball, el sistema demanarà a l'operador el codi de la primera comanda d'aquest. Un cop entrat el codi , el sistema hi afegeix -1 al final i mostra la pantalla on l'operador haurà d'entrar totes les dades de la comanda i enviar-les. A més perquè es puguin fer més comandes, l'usuari podrà recuperar una fitxa ja creada (d'una llista), llavors el sistema utilitzarà el mateix codi de la primera incrementant una unitat després del guió.

- ***Consultar/Modificar fitxa***: A més l'operador sempre que ho necessiti podrà recuperar una fitxa per consultar-ne els seus paràmetres, i amb l'opció de modificar-los (la informació referent als tinters no es podrà modificar ja que sinó s'estaria canviant la naturalesa del treball).

- ***Recuperar fitxa***: Per recuperar una fitxa, l'usuari tindrà una opció en el menú del seu perfil on demanarà la llista de les fitxes ja creades.

- ***Llistar fitxes***: Aquesta funcionalitat és el complement de les dues anteriors. El sistema llista fitxes ja creades, totes o bé filtrades per el seu nom o part del seu nom.

2.2.3 Funcionalitats consulta i estadística.

- ***Llistar comandes***: Un cap/coordinador podrà obtenir una llista de totes les comandes, o bé filtrades per el seu codi o part d'aquest.

- ***Mostrar dades comanda:*** Un cop l'usuari hagi escollit una comanda de la llista, el sistema l'hi mostrarà totes les dades d'aquesta.

- ***Metres totals entre dues dates:*** Un cop el cap/coordinador hagi escollit aquesta opció, el sistema l'hi mostrarà un petit filtre on l'usuari haurà d'entrar una data d'inici i una data final. Un cop hagi enviat la petició , se l'hi mostraran els metres totals, de qualitat i de no qualitat entre aquest interval de temps.

2.2.3 Funcionalitats manteniment.

- ***Alta usuari***: En aquesta funcionalitat el personal administratiu (a títol d'administrador) ,quan vol donar d'alta un usuari en el sistema, escull la opció alta usuari dins el subsistema de manteniment i el sistema l'hi mostra totes els camps que aquest haurà d'omplir. (nom usuari, Nom, Cognoms, correu electrònic, nif, tipus usuari (rol), etc... i quan ha acabat envia l'alta en el sistema .

- ***Llistar usuaris:*** En el cas que necessitin conèixer les dades dels usuaris, tindran l'opció de llistar tots els usuaris del sistema Només haurà de seleccionar l'opció llistar Usuaris. A més disposarà d'un filtre on podrà llistar segons nom. Si es vol modificar usuaris es mostraran tant els actius com els no actius, i si es vol donar de baixa un usuari només es llistaran els actius.

- ***Consultar/Modificar usuari:*** Si l'administrador ha de modificar o consultar alguna dada d'un usuari, un cop hagi executat la funcionalitat anterior "Llistar Usuaris", seleccionarà l'opció "editar" de l'usuari en qüestió. L'usuari pot modificar la dada que necessiti i activar el polsador "Modificar" perquè tingui efecte, o bé només consultar les dades i seleccionar el polsador "cancel·lar" per tornar a la pantalla "llistar usuaris".

- ***Baixa usuari:*** Quan sigui necessari donar de baixa a un usuari dins el sistema, i si l'administrador ha executat la funcionalitat anterior "Llistar Usuaris", seleccionarà l'opció "baixa" de l'usuari en qüestió i el sistema l'hi mostrarà un missatge preguntant-l'hi si està segur de l'acció.Si selecciona "si" la baixa tindrà efecte i si selecciona "cancel·lar" el sistema el tornarà a la pantalla anterior "Llistar Usuaris baixa".

2.3. Requisits no funcionals

Seguidament s'esmenten els requisits que afecten el sistema en general.

- ***Intuïtiu i corba d'aprenentatge lenta:*** El sistema s'ha de poder utilitzar sense la necessitat de rebre formació prèvia.És de tipus de usabilitat i humanitat.

- ***Fàcil de mantenir:*** Ha de ser fàcil de mantenir sense necessitat de contactar personal per a configuracions i manteniment com còpies de seguretat, etc... És de tipus de manteniment i suport.

- ***Suportar milers de fitxes i comandes:*** El sistema ha d'estar preparat per suportar una gran quantitat de informació, ja que al caps dels anys les fitxes i les seves comandes seran de milers.Aquesta és de tipus de compliment d'escalabilitat.

- ***Varis idiomes:*** Sistema multi idioma, ja que ha de ser un sistema que pot arribar a qualsevol país. Primer implementat amb tres idiomes (català , castellà i anglès), i el idioma per defecte s'ha de poder configurar.Aquest és de tipus cultural i polític.

- ***Logotip i colors corporatius:*** Totes les pantalles de l'aplicació han de mostrar el logotip (FlexographicControl) amb els seus colors corporatius. Aquest requisit no funcional és de tipus de presentació.

- ***Validacions de totes les dades:*** Totes les dades que l'usuari ha d'entrar estaran validades abans de ser enviades al sistema per tal d'evitar excepcions i errors innecessaris.Per tant quan l'usuari entri dades com numèriques, límits, dates, nif, correu electrònic , etc..., la capa de presentació validarà aquesta informació avisant a l'usuari en el cas que no hagi entrat les dades correctament. És de tipus de usabilitat i humanitat.

2.4. Diagrama de casos d'ús

De tots els diagrames UML (Unified Modeling Language), els de cas d'ús són els més útils de cares a detectar les classes d'entitat del nostre projecte.Per tant aquest diagrama ha estat el punt de partida del nostre projecte.També s'han elaborat altres diagrames dinàmics com els de seqüència i activat perquè ens ajudin a implementar alguns dels cassos d'ús més rellevants.

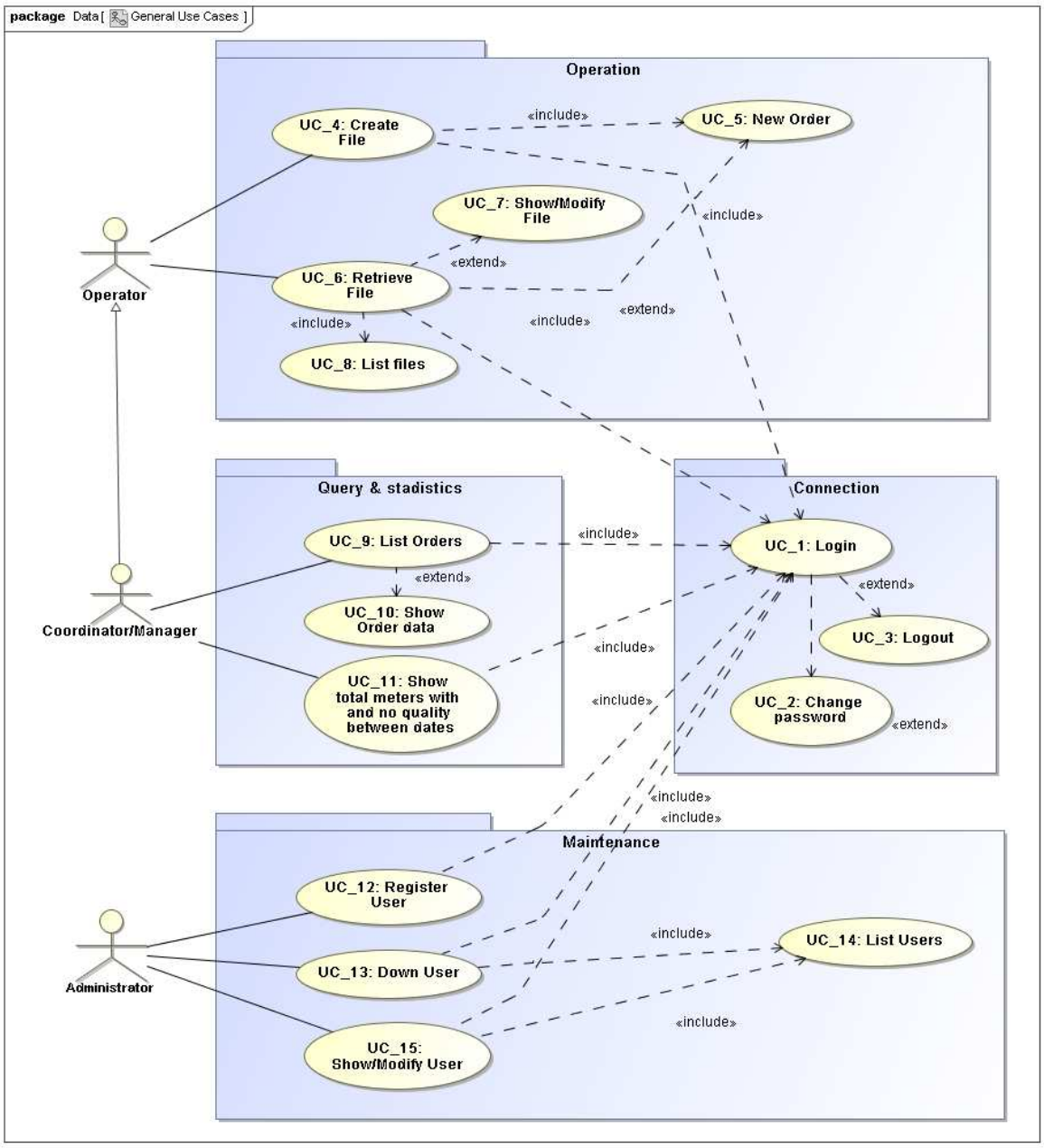

Figura 4. Diagrama casos d'us

2.5. Especificació textual dels casos d'ús

Gràcies a les especificacions textuals dels casos d'ús podrem conèixer amb més precisió les seves tasques internes, i a més ens servirà de guió a l'hora de fer la seva implementació.

2.5.1 Autenticar (Login)

1 . Autenticar (Login)	
Resum de la funcionalitat	Un usuari accedeix al sistema i aquest comprova que el nom usuari i la contrasenya són correctes.
Actors	Operador, cap, coordinador i administrador
Casos d'ús relacionats	4.Crear fitxa, 6.Recuperar fitxa, 9.Llistar comandes, 11.Mostrar metres totals entre dos dates, 12.Registrar usuari, 13.Baixa usuari, 15.Mostrar/Modificar usuari
Precondició	Un usuari vol entrar al sistema
Postcondició	L'usuari ha accedit al sistema i el sistema l'hi mostra el seu perfil.
Procés normal principal	1. L'Usuari arranca l'aplicació mòbil/escriptori o bé escriu la direcció a l'explorador web (actualment només web). 2. El sistema demana el nom d'usuari i contrasenya. 3. L'usuari introdueixen les dades. 4. El sistema accedeix a la base de dades, i fa la consulta. 5. El sistema comprova les dades introduïdes amb les de la base de dades i identifica el rol de l'usuari. 6. El sistema dona mostra a l'usuari la interfície amb les funcionalitats que està autoritzat a utilitzar en funció del seu rol . 7.El sistema finalitza el cas d'ús.
Alternatives de procés i excepcions	4a. No s'ha pogut connectar amb la base de dades. 4a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús. 4b. Si s'ha pogut connectar a la base de dades. 4b1. Es passa a la tasca següent. 5a. El nom d'usuari o la contrasenya no són correctes. 5a1. El sistema mostra el corresponent missatge d'avís i passa a la tasca 2. 5b. S'ha produït alguna excepció. 5b1. Es mostra l'excepció per pantalla i es torna a la tasca 2.

2.5.2 Canviar contrasenya (Change password).

2 . Canviar contrasenya	
Resum de la funcionalitat	Un usuari per qüestions de seguretat vol modificar la seva contrasenya actual .
Actors	Operador, cap, coordinador i administrador
Casos d'ús relacionats	1.Autenticar.
Precondició	L'usuari esta a la pantalla d'autenticació vol canviar la seva contrasenya.
Postcondició	L'usuari té la nova contrasenya guardada a la base de dades, concretament a l'atribut password de la taula "users" està a la pantalla inicial "Login".
Procés normal principal	1. L'Usuari escull canviar contrasenya. 2. El sistema demana el nom d'usuari , contrasenya vella i la contrasenya nova dues vegades. 3. L'usuari introdueixen les dades. 4. El sistema accedeix a la base de dades, i fa la consulta sobre la taula "users". 5. El sistema comprova les dades introduïdes (usuari i contrasenya vella) amb les de la base de dades i identifica el rol de l'usuari . 6. El sistema executa la sentència SQL que guarda la nova contrasenya a la taula "users" . 7. El sistema executa el cas d'ús 1 "Login". 8.El sistema finalitza el cas d'ús.
Alternatives de procés i excepcions	4a. No s'ha pogut connectar amb la base de dades. 4a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús. 4b,6a. Si s'ha pogut connectar a la base de dades. 4b1,6a1. Es passa a la tasca següent. 5a. El nom d'usuari o la contrasenya vella no són correctes. 5a1. El sistema mostra el corresponent missatge d'avís i passa a la tasca 2. 5b,6b. S'ha produït alguna excepció. 5b1,6b1. Es mostra l'excepció per pantalla i es torna a la tasca 2.

2.5.3 Sortir del sistema (Logout).

3 . Sortir del Sistema	
Resum de la funcionalitat	Un usuari vol sortir del sistema un cop ha acabat les seva tasca i que no es pugui fer res a nom seu.
Actors	Operador, cap, coordinador i administrador
Casos d'ús relacionats	1.Autenticar.
Precondició	Un usuari vol sortir del sistema i està en el seu menú d'opcions.
Postcondició	El sistema mostra la pantalla Autenticar
Procés normal principal	1. L'Usuari escull sortir del sistema. 2.El sistema executa el cas d'ús 1 "Autenticar". 3.El sistema finalitza el cas d'ús.

2.5.4 Crear Fitxa (Create file).

4 . Crear Fitxa	
Resum de la funcionalitat	Un usuari crearà una nova fitxa d'un treball flexogràfic del qual se'n faran una o més comandes.
Actors	Operador, cap i coordinador
Casos d'ús relacionats	1.Autenticar, 5.Nova comanda
Precondició	Un usuari ja esta autenticat al sistema i té permís i vol crear fitxa nova.
Postcondició	Una nova fitxa (treball flexogràfic) està guardada a la base de dades.
Procés normal principal	1. L'Usuari escull crear fitxa nova del sistema. 2. El sistema mostra una pantalla amb totes les dades a omplir. 3. L'usuari entra totes les dades de la nova fitxa i accepta. 4. El sistema valida les dades (límits, formats,...) 5. El sistema guarda les dades de la fitxa i executa el cas d'ús 5 nova comanda. 6. El sistema finalitza el cas d'ús.
Alternatives de procés i excepcions	3a,6a. L'usuari cancel.la 3a1,6a1. El sistema finalitza el cas d'ús. 4a.Algun format de dada no validada o el nom ja existeix en el sistema. 4a1. El sistema mostra el missatge corresponent i torna a la tasca 2. 4b. No s'ha pogut connectar amb la base de dades o altre excepció. 4b1. El sistema mostra un missatge d'error a i finalitza el cas d'ús.

2.5.5 Nova Comanda (New Order).

5 . Nova Comanda	
Resum de la funcionalitat	Un usuari acaba de crear una fitxa d'un nou treball i vol fer la primera comanda, o bé ha recuperat una fitxa existent al sistema per fer una nova comanda.
Actors	Operador
Casos d'ús relacionats	4.Crear fitxa, 6.Recuperar fitxa
Precondició	Un usuari ha creat una fitxa nova o n'ha recuperat una d'existent.
Postcondició	El sistema haurà mostrat una pantalla amb tots els camps de la nova ordre. Alguns amb informació i d'altres que l'usuari haurà d'omplir un cop feta la comanda.
Procés normal principal	1. El sistema demana el codi de la nova comanda. 2. El sistema mostra les dades de la nova comanda i els camps que l'usuari haurà d'omplir. 3. L'usuari entra totes les dades de la comanda acabada. 4. El sistema valida les dades (límits, formats,...) 5. El sistema connecta a la BBDD i registra la nova comanda. 4.El sistema finalitza el cas d'ús.
Alternatives de procés i excepcions	1a. Si ve de recuperar fitxa existent, el sistema incrementa una unitat després del "-" del codi de comanda ja existent per aquest treball. 4a.Alguna dada no validada. 4a1. El sistema mostra el missatge corresponent i torna a la tasca 2. 5a. No s'ha pogut connectar amb la base de dades o altre excepció. 5a1. El sistema mostra un missatge d'error a i finalitza el cas d'ús.

2.5.6 Recuperar fitxa (Retrieve file).

6. Recuperar fitxa	
Resum de la funcionalitat	L'usuari ha recuperat una llista de fitxes existents (cas d'ús 8) , en vol modificar una (algun paràmetres) o bé fer una nova comanda.
Actors	Operador
Casos d'ús relacionats	5.Nova comanda, 7.Mostrar/Modificar fitxa, 8. Llistar fitxes
Precondició	Un usuari està autenticat en el sistema i té permisos per modificar alguna dada d'una fitxa existent o bé fer-ne una comanda.
Postcondició	El sistema ha executat el cas d'us 7 o bé el 8.
Procés normal principal	1. L'usuari executa opció recuperar fitxa 2. El sistema executa el cas d'ús 10. Llistar fitxes existents. 3. L'usuari escull una fitxa ja existent
Alternatives de procés i excepcions	3a . L'usuari torna al menú principal . 3a1. El sistema finalitza el cas d'ús. 3b. L'usuari escull modificar la fitxa 3b1. El sistema executa el cas d'ús 7. Modificar fitxa. 3c. L'usuari escull fer una comanda de la fitxa. 3c1. El sistema executa el cas d'ús 5. Nova comanda.

2.5.7 Mostrar/Modificar fitxa (Show/Modify file).

7. Mostrar/Modificar fitxa	
Resum de la funcionalitat	El sistema mostra les dades d'una fitxa existent que s'ha recuperat per permetre que pugui modificar les que necessiti o simplement consultar-ne les dades.
Actors	Operador
Casos d'ús relacionats	6.Recuperar fitxa
Precondició	Un usuari ha seleccionat una fitxa existent i el sistema ha executat el cas d'ús 7 i ha seleccionat una fitxa per consultar o modificar.
Postcondició	El sistema ha actualitzat la fitxa recuperada.
Procés normal principal	1. El sistema connecta a la capa de dades i recupera totes les dades de la fitxa seleccionada en el cas d'ús 6. 2. Les mostra per pantalla (algunes no es poden modificar). 3.L'Usuari modifica dades i executa l'ordre "Modificar". 4. El sistema valida les dades (límits, formats,...) 5. El sistema connecta a la BBDD i actualitza les dades de la fitxa. 6.El sistema finalitza el cas d'ús .
Alternatives de procés i excepcions	1, 5, a. No s'ha pogut connectar amb la bbdd o altre excepció. 1, 5, a1. El sistema mostra un missatge d'error i finalitza el cas d'ús. 2a L'usuari cancel.la . 2a1. El sistema torna executa el cas d'ús 9. Recuperar fitxa. 4. Alguna dada no validada el sistema torna a la tasca 2.

2.5.8 Llistar fitxes (List files).

8. Llistar fitxes	
Resum de la funcionalitat	El sistema llista per nom totes les fitxes existents que hi ha guardades a la capa de dades. També es poden filtrar per el seu nom o part del seu nom.
Actors	Operador
Casos d'ús relacionats	6.Recuperar Fitxa
Precondició	Un usuari ha executat el cas d'ús 6.Recuparar fitxa
Postcondició	L'Usuari disposa d'un llistat amb totes les fitxes existents o les que han estat demanades per el filtre (per nom).
Procés normal principal	1. El sistema connecta amb la BBDD i demana el llistat de les fitxes . 2. El sistema mostra la llista de les fitxes 3. El sistema finalitza el cas d'ús i torna al cas d'ús des de on s'ha executat aquesta funció 6.Recuperar fitxa .
Alternatives de procés i excepcions	1a. No s'ha pogut connectar amb la base de dades o altre excepció. 1a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús.

2.5.9 Llistar Comandes (List Orders).

9. Llistar comandes	
Resum de la funcionalitat	El sistema llista per codi totes les comandes existents que hi ha guardades a la capa de dades. També es poden filtrar per el seu codi o part del seu codi.
Actors	Cap i coordinador
Casos d'ús relacionats	1.Autenticar, 10. Mostrar dades comanda
Precondició	Un usuari esta autenticat i té permisos per fer aquesta consulta.
Postcondició	L'Usuari disposa d'un llistat de totes (o algunes) comandes .
Procés normal principal	1. L'usuari executa la funció llistar comandes. 2. El sistema connecta a la capa de dades, consulta el llistat de les comandes. 3. El sistema mostra la llista de les fitxes . 4. L'usuari selecciona la comanda desitjada. 5.El sistema executa el cas d'ús 10 "Mostrar dades comanda".
Alternatives de procés i excepcions	2a. No s'ha pogut connectar amb la base de dades o altre excepció. 2a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús. 3a. L'usuari decideix tornar al menú principal. 3a1. El sistema torna a la vista del menú principal

2.5.10 Mostrar dades comanda (Order data).

10. Mostrar dades comanda	
Resum de la funcionalitat	L'usuari ha seleccionat una comanda i el sistema l'hi mostra totes les seves dades de producció.
Actors	Cap i coordinador
Casos d'ús relacionats	9.Llistar comandes
Precondició	Un usuari té la llistat de les comandes .
Postcondició	L'Usuari disposa de tot els detalls de la comanda que ha seleccionat.
Procés normal principal	1. L'usuari té seleccionada una comanda i executa la opció mostrar dades comanda. 2. El sistema connecta a la capa de dades, consulta tota la informació de la comanda en qüestió i les mostra a l'usuari. 3.El sistema finalitza el cas d'ús.
Alternatives de procés i excepcions	2a. No s'ha pogut connectar amb la base de dades o altre excepció. 2a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús.

2.5.11 Mostrar metres totals entre dues dates (Show total metres between dates).

11. Mostrar metres totals entre dues dates	
Resum de la funcionalitat	L'usuari entra una data inici i una final i el sistema l'hi mostra els metres totals (qualitat i no qualitat) de totes les comandes.
Actors	Cap i coordinador
Casos d'ús relacionats	1.Autenticar
Precondició	Un usuari esta autenticat i té permisos per fer aquesta consulta.
Postcondició	L'Usuari disposa dels metres totals de qualitat i de desperdici de l'interval de temps seleccionat.
Procés normal principal	1. L'usuari executa la opció mostrar metres totals. 2. El sistema mostra els camps de data inici i data fi. 3. L'usuari entra les dates 4. El sistema connecta a la capa de dades, consulta la informació i la mostra a l'usuari. 5.El sistema finalitza el cas d'ús.
Alternatives de procés i excepcions	3a . L'usuari cancel.la 3a1. El sistema finalitza el cas d'ús. 4a. No s'ha pogut connectar amb la base de dades o altre excepció. 4a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús.

2.5.12 Registrar Usuari (Register user).

12. Registrar usuari	
Resum de la funcionalitat	L'administrador dona d'alta un nou usuari i el sistema el registra a la base de dades.
Actors	Administrador
Casos d'ús relacionats	1.Autenticar
Precondició	L'Administrador esta autenticat i té permisos per gestionar usuaris.
Postcondició	Un nou usuari està registrat a la base de dades.
Procés normal principal	1. L'administrador executa l'opció "registrar usuari" 2. El sistema mostra els atributs buits referents a un usuari. 3. L'administrador omple les dades i accepta. 4. El sistema valida les dades . 5. El sistema connecta a la capa de dades, i registra el nou usuari. 6. El sistema finalitza el cas d'ús.
Alternatives de procés i excepcions	2a . L'Administrador cancel.la 2a1. El sistema finalitza el cas d'ús. 4a.Alguna dada no ha estat validada. 4a1. El sistema mostra el missatge corresponents i torna a la tasca 2. 5a. No s'ha pogut connectar amb la base de dades o altre excepció. 5a1. El sistema mostra l'error a l'usuari i finalitza el cas d'ús.

2.5.13 Baixa Usuari (Down user).

13. Baixa usuari	
Resum de la funcionalitat	L'administrador dona de baixa un usuari existent i el sistema actualitza el camp actiu amb el valor fals.
Actors	Administrador
Casos d'ús relacionats	1.Autenticar, 14.Llistar usuaris
Precondició	L'Administrador esta autenticat i té permisos per gestionar usuaris.
Postcondició	L'Usuari esta marcat com a inactiu a la base de dades.
Procés normal principal	1. L'administrador executa l'opció "baixa usuari" 2. El sistema executa el cas d'us 14.Llistar usuaris. 3. L'administrador escull un usuari i pressiona baixa. 4. El sistema mostra un missatge i demana si esta segur. 5. El sistema connecta a la capa de dades, i actualitza l'usuari com a inactiu. 6.El sistema finalitza el cas d'ús
Alternatives de procés i excepcions	3a ,4a . L'Administrador cancel.la 3a1, 4a1. El sistema finalitza el cas d'ús. 5a. No s'ha pogut connectar amb la base de dades o altre excepció. 5a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús.

2.5.14 Llistar usuaris (List users).

14. Llistar usuaris	
Resum de la funcionalitat	El sistema llista els usuaris que hi ha guardats a la capa de dades. Si es ve del cas d'ús 13, només mostra els actius i si es ve del cas d ús 15, llavors mostra tant actius com inactius.També es poden filtrar pel seu nom.
Actors	Administrador
Casos d'ús relacionats	13.Baixa usuari, 15.Modificar usuari
Precondició	L'administrador ha executat el cas d'ús 13.Baixa usuari o bé el 15.Modificar usuari.
Postcondició	L'administrador disposa d'un llistat amb tots (o alguns) usuaris .
Procés normal principal	1. El sistema connecta amb la BBDD i consulta la llista dels usuaris que l'administrador ha demanat. 2. El sistema mostra la llista dels usuaris 3. El sistema finalitza el cas d'ús i torna al cas d'ús des de on s'ha executat aquesta funció 13.Baixa usuari o bé el 15.Modificar usuari.
Alternatives de procés i excepcions	1a. No s'ha pogut connectar amb la base de dades o altre excepció. 1a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús.

2.5.15 Mostrar/Modificar usuari (Modify user).

15. Mostrar/Modificar usuari	
Resum de la funcionalitat	L'administrador modifica un usuari existent i el sistema actualitza les dades de la capa de dades o bé només les consulta.
Actors	Administrador
Casos d'ús relacionats	1.Autenticar, 14.Llistar usuaris
Precondició	L'Administrador esta autenticat i té permisos per gestionar usuaris.
Postcondició	L'Usuari està actualitzat en el sistema
Procés normal principal	1. L'administrador executa l'opció "modificar usuari" 2. El sistema executa el cas d'us 14.Llistar usuaris. 3. L'administrador escull un usuari i pressiona Modificar. 4. El sistema connecta a la capa de dades, recupera l'usuari i mostra les dades. 5. L'administrador modifica les dades desitjades i guarda. 6.El sistema connecta a la capa de dades i actualitza les dades de l'usuari. 7. El sistema finalitza el cas d'ús.
Alternatives de procés i excepcions	3a., 5a. L'Administrador cancel.la . 3a1, 5a1, El sistema finalitza el cas d'ús. 4a, 6a. No s'ha pogut connectar amb la BBDD o altre excepció. 4a1, 6a1. El sistema mostra un missatge d'error a l'usuari i finalitza el cas d'ús.

2.6. Altres diagrames

Amb els d'activitat es mostra el flux entre els objectes i són molt útils per modelar el funcionament del sistema, el flux de control entre objectes i també tenir clares les pantalles.S'han fet el de comanda ja que és el més rellevant del projecte.

També comentar que el diagrames de seqüència s'ha decidit posar-los a la part de disseny, pel fet que tenen en compte l'arquitectura de l'aplicació (tres capes) , i això es contempla a la par de disseny.

2.6.1 Diagrama d'estats d'una comanda

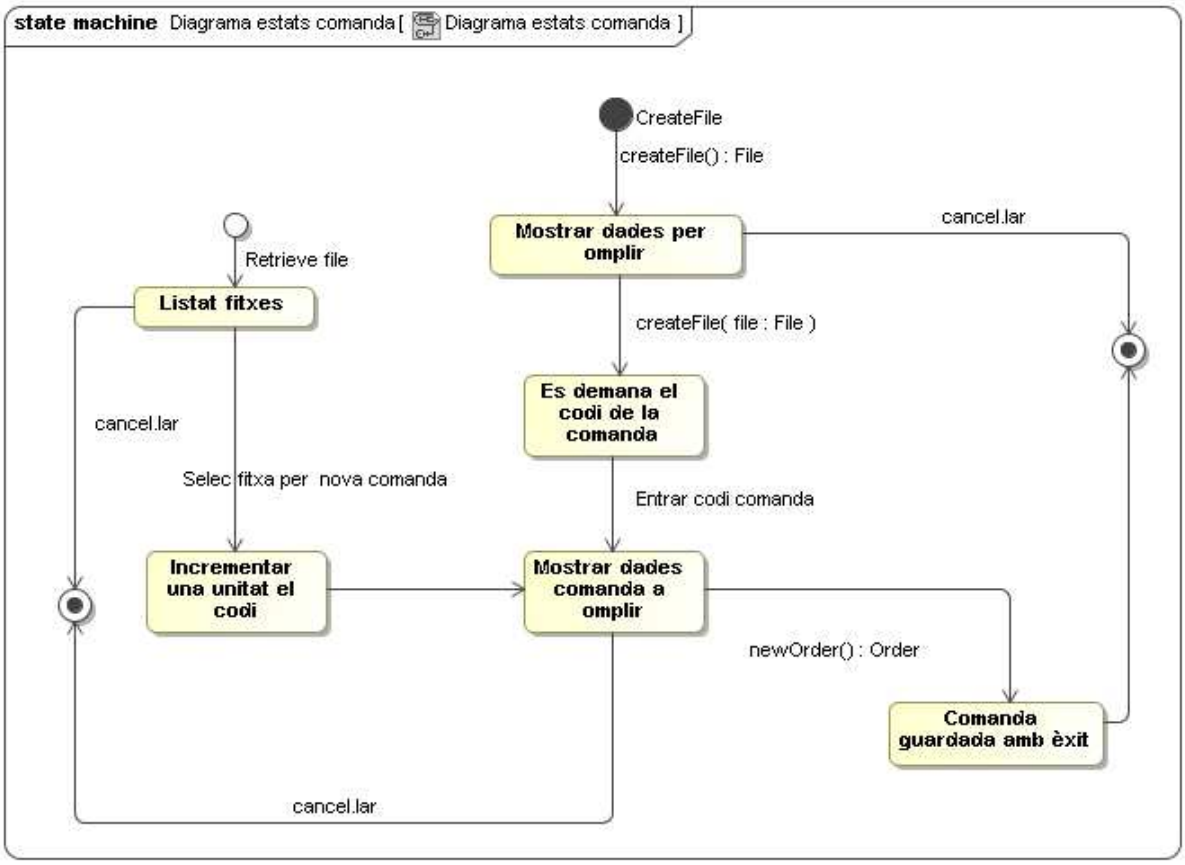

Figura 5. Diagrama d'estats d'una comanda

3. Disseny

Ja arribats aquest punt, tenim la informació suficient per passar a la part de disseny de la nostra aplicació i decidir i crear assumptes com: arquitectura de l'aplicació, model conceptual, diagrama de classes d'entitat, de components distribuïts, la interfície de l'usuari, la base de dades i alguns altres components més.

3.1. Arquitectura computacional

La part d'arquitectura de programari és molt important, ja que seria com els fonaments d'una construcció arquitectònica, que sinó estan ben pensats i ben executats, el futur edifici en patirà les conseqüències.

3.1.1 Arquitectura client-Servidor (tres capes)

Llavors s'ha pensat en una arquitectura que permeti treball en grup, reutilització, manteniment ràpid i confortable.Per aquestes raons s'ha escollit una arquitectura ***client/servidor***, i la part servidora estarà formada per ***3 capes*** (presentació, lògica negoci i dades), amb l'objectiu de tenir aquestes parts aïllades entre sí.A més, si s'ha de fer algun canvi que afecti a una part en concret del projecte, aquest no afecti a les altres.

- Presentació que permetrà la interacció dels usuaris amb el servei.
- Lògica del negoci que implementa la funcionalitat bàsica de l'aplicació.
- Integració o dades que permet interactuar amb les fonts de dades que emmagatzemen la informació persistent.

A més, si en un futur l'aplicació s'ha d'actualitzar amb noves funcionalitats, es podrà repartir la feina en diferents grups de treball permetent-los treballar per separat.

Per exemple, si algun dia es vol canviar el servidor de base de dades per una altre marca comercial o bé per una actualització de versió, ens serà molt més ràpid i còmode dur a terme el canvi tenint la part d'accés a dades aïllada de tota la resta de l'aplicació. També si en algun moment hem d'actualitzar o afegir algunes llibreries de components com per exemple Ajax ,Richfaces, GoogleMaps, facebook, etc..., només ens afectarà a la capa de presentació i les altres dues capes no s'adonaran del canvi.

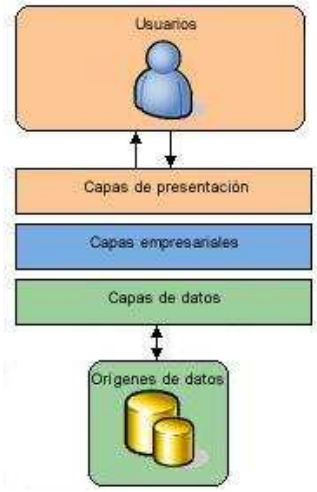

Figura 6. Arquitectura tres capes

3.1.2 Model MVC

A la capa presentació utilitzarem el patró model-vista-controlador (MVC), que dividint aquesta part amb tres components (model, vistes i controladors) , ajuda a organitzar les responsabilitats d'interacció amb l'usuari desacoblant la interfície gràfica del nostre sistema de la resta del sistema i fer una bona separació de responsabilitats.

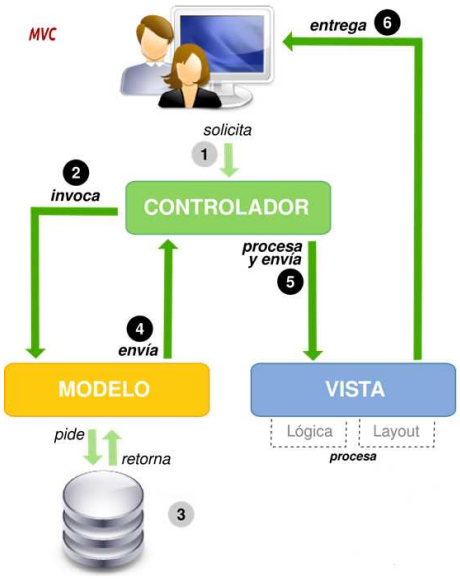

Figura 7. Model Vista Controlador (MVC)

3.1.3 Framworks i model tres capes

Per la capa de ***Presentació*** (i el patró MVC) s'utilitzarà el framework Java Server Faces (JSF 2) que ve amb la especificació de JEE. El controlador serà el servlet Faces Servlet, les accions estaran definides amb Managed Bean i s'implementaran les vistes amb Facelets.A més en aquesta capa s'hi ha integrat HTML i CSS.

A la capa de ***Lògica*** voldrem que les consultes no tinguin estat i a més les transaccions siguin gestionades per un contenidor i les peticions sempre siguin síncrones. Per tant la solució que aplicarem serà mitjançant EJB3 (Enterprise Java Beans) de sessió sense estat i accés remot seguint un patró façana. Evidentment aquesta part serà la que s'entengui en el sistema com una aplicació orientada a objectes i programada en llenguatge Java.

I per la capa de ***Dades*** s'ha escollit el PostgreSQL, que gaudeix d'una gran reputació, fins al punt que molts ja diuen que és el millor gestor de base de dades dins el programari lliure. I després d'haver utilitzat MYSQL i PostgreSQL, em quedo amb l'últim ja que m'hi vaig sentir més còmode.Així doncs per emmagatzemar les dades , seguirem l'espicificació EJB 3 i utilitzarem entitats Java percistence (JPA) que ens permetrà gestionar la persistència de dades i les seves relacions des de la part de java, on es tracten les entitats de la base de dades coma objectes.Per lligar aquestes dues tecnologies s'utilitza implicitament el framework Hibernate.

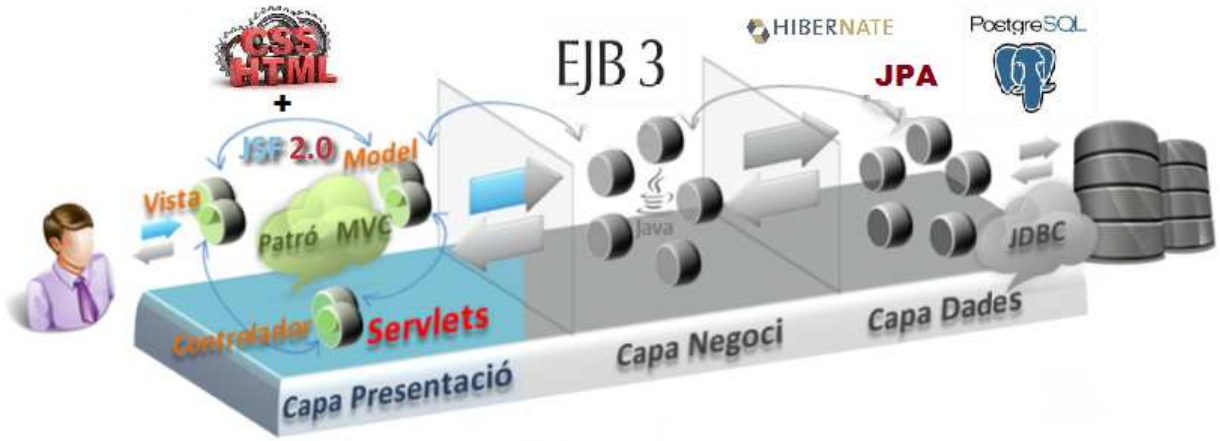

Figura 8. Frameworks i arquitectura tres capes

3.2. Diagrama de clases

A la capa de lògica negoci és on s'estableixen les regles i per tant també les relacions, per això en aquesta és on hi implementem el diagrama de classes que s'ha engendrat a partir d'un model conceptual previ.

3.2.1 Model conceptual

Primer s'ha confeccionat les relacions del model conceptual de les classes d'entitat, per després crear el diagrama de les classes d'entitat.

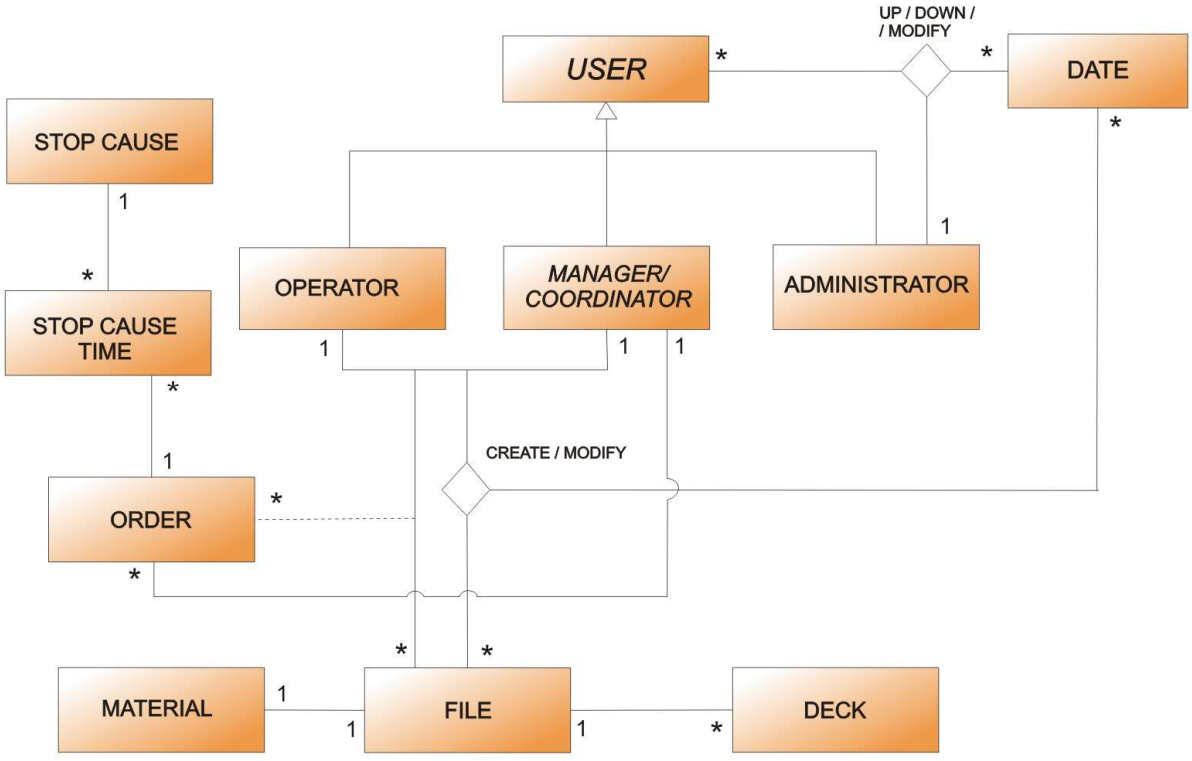

Figura 9. Model conceptual

3.2.2 Diagrama estàtic de classes

En el model conceptual hi tenim una classe associativa que s'haurà de transformar a classe normal perquè el nostre llenguatge de programació Java no les tracta.Per tant l'hi aplicarem una transformació de reificació, per tal de relacionar la nova classe amb les dues classes que participen a l'associació.Per tant la classe associativa Order (Comanda) s'ha de convertir a una classe normal i mantenint els mateixos atributs.

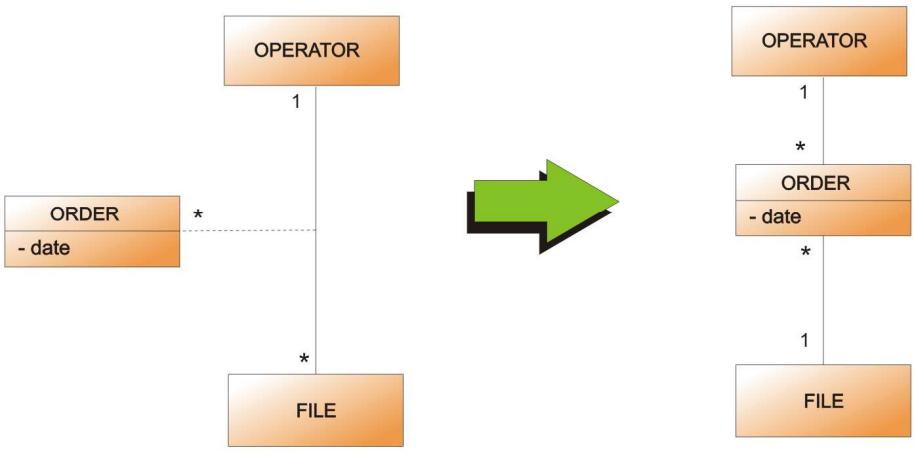

Figura 10. Reificació classe associativa Order

A més les entitats operador, cap/coordinador i administrador les reduirem només a una; Usuari. Per diferenciar el rol d'un usuari afegirem un atribut anomenat tipus_rol de tipus enter i associat a una classe anomenada rol amb els atributs identificador i descripció.

També la data deixarà de ser una entitat i passarà a ser un atribut de la classes que treballin amb dates.

El darrer pas consisteix a substituir les associacions per atributs i aquests tindran l'estereotip "ref", però en el nostre model només tindrà sentit si en un futur volem llistar els treballs creats o modificats per un determinat usuari, comandes creades, usuaris creats, modificats, o desactivats per un determinat administrador.

En el següent gràfic es pot apreciar el diagrama de classes d'entitat tal com ha quedat i que serà utilitzat tant com per la capa de negoci com per la capa de dades, ja que les entitats han de coincidir perquè utilitzem el framework JPA amb Hibernate.

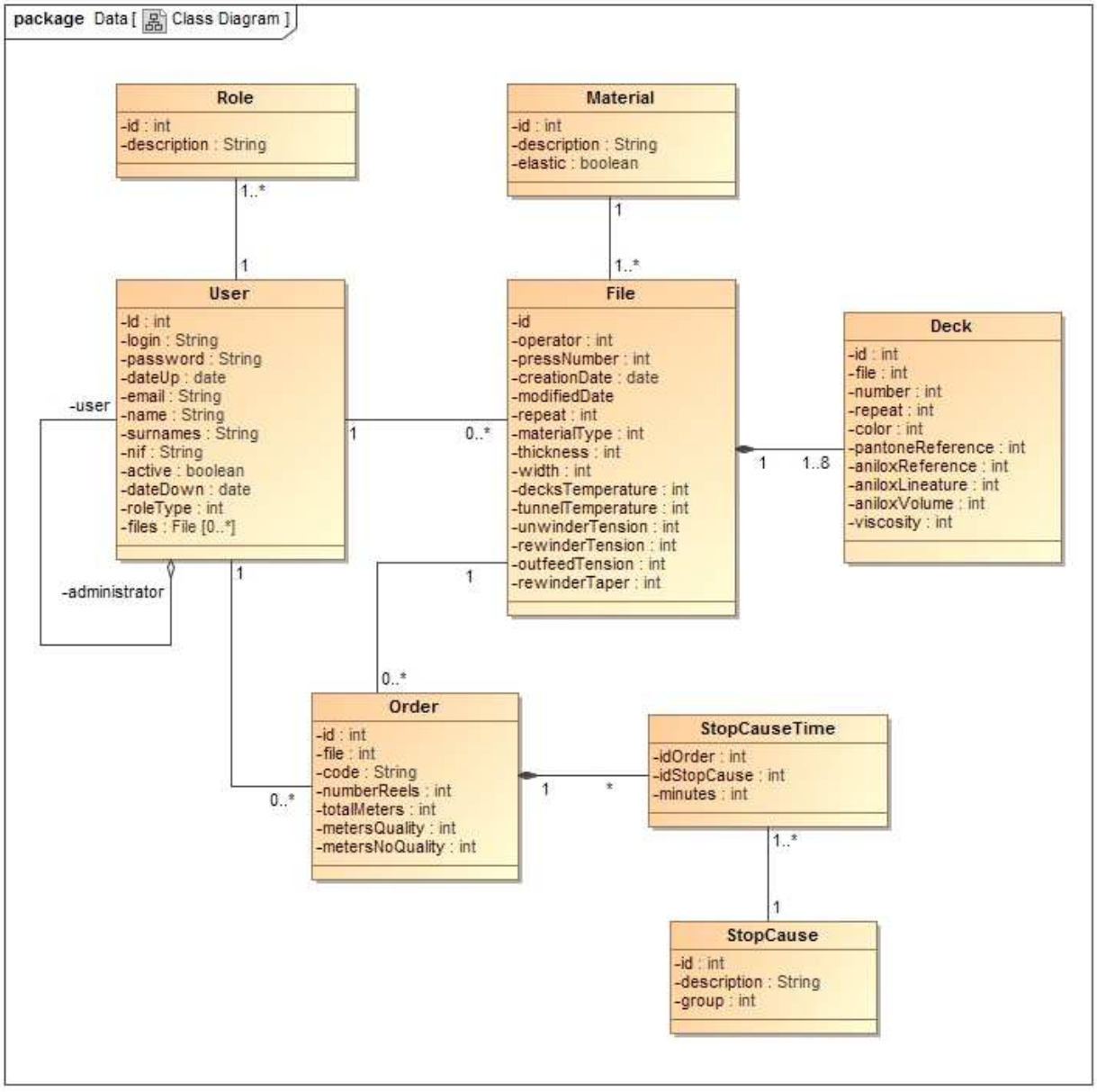

Figura 11. Diagrama de classes

3.3. Diagrames de components distribuïts

Tal com s'ha explicat a la a part d'arquitectura, s'ha escollit una arquitectura de programari de tipus client/servidor de tres capes (Presentació, lògica de negoci i integració de dades).

- Presentació que permetrà la interacció dels usuaris amb el servei.
- Lògica del negoci que implementa la funcionalitat bàsica de l'aplicació.
- Integració o dades que permet interactuar amb les fonts de dades que emmagatzemen la informació persistent.

Per representar aquest punt de vista computacional ho fem mitjançant diagrames de components.Cada capa a nivell lògic es representa com un paquet, i dins aquest paquet s'identifiquen els components lògics de gra gruixut.Aquests components lògics tenen les seves respectives interfícies que mostres la signatura dels serveis que ofereix cada component.

S'han organitzat els paquets en capes conceptuals independents: Presentació, Negoci i Integració.

S'han identificat els quatre components arrel dels subsistemes (també enteses com a divisions funcionals) que es van decidir a la primera fase del projecte : Connexió, Operació, Consulta i Estadística i Manteniment.Per tal de que es puguin visualitzar bé totes les signatures de les interfícies, hem partir el diagrama en dos parts (dos subsistemes per cada part).

Un cop obtingut aquest diagrama que representa el punt de vista de la computació , es fa el refinament i disseny aplicant el perfil JAVA EE. No ho farem de tots els components ja que sinó el volum de pàgines d'aquesta pac pot créixer considerablement, per tant s'ha escollit fer el component més rellevant que és el de Operació.

3.3.1 Diagrames de components

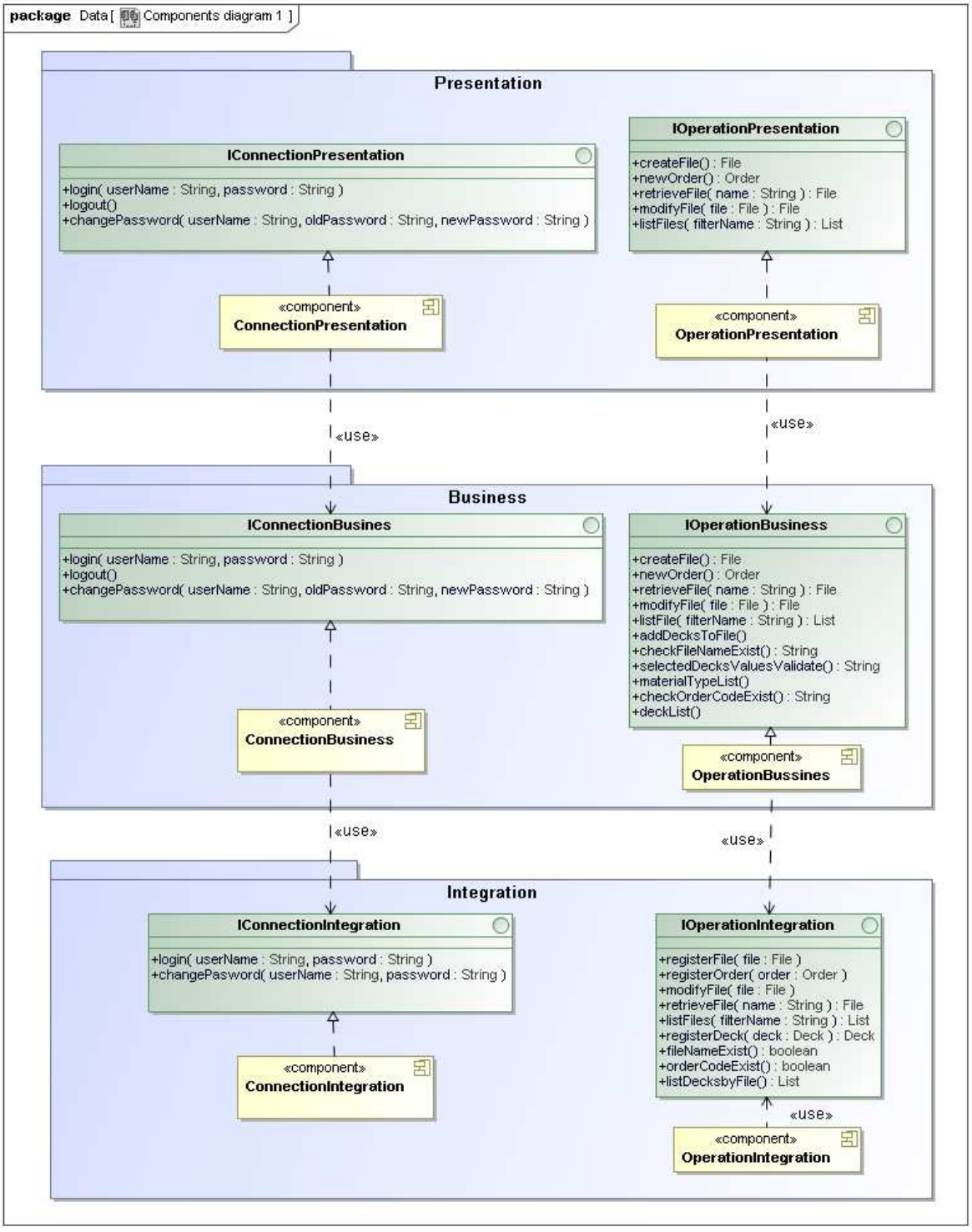

Figura 12. Diagrama de components (connexió i operació)

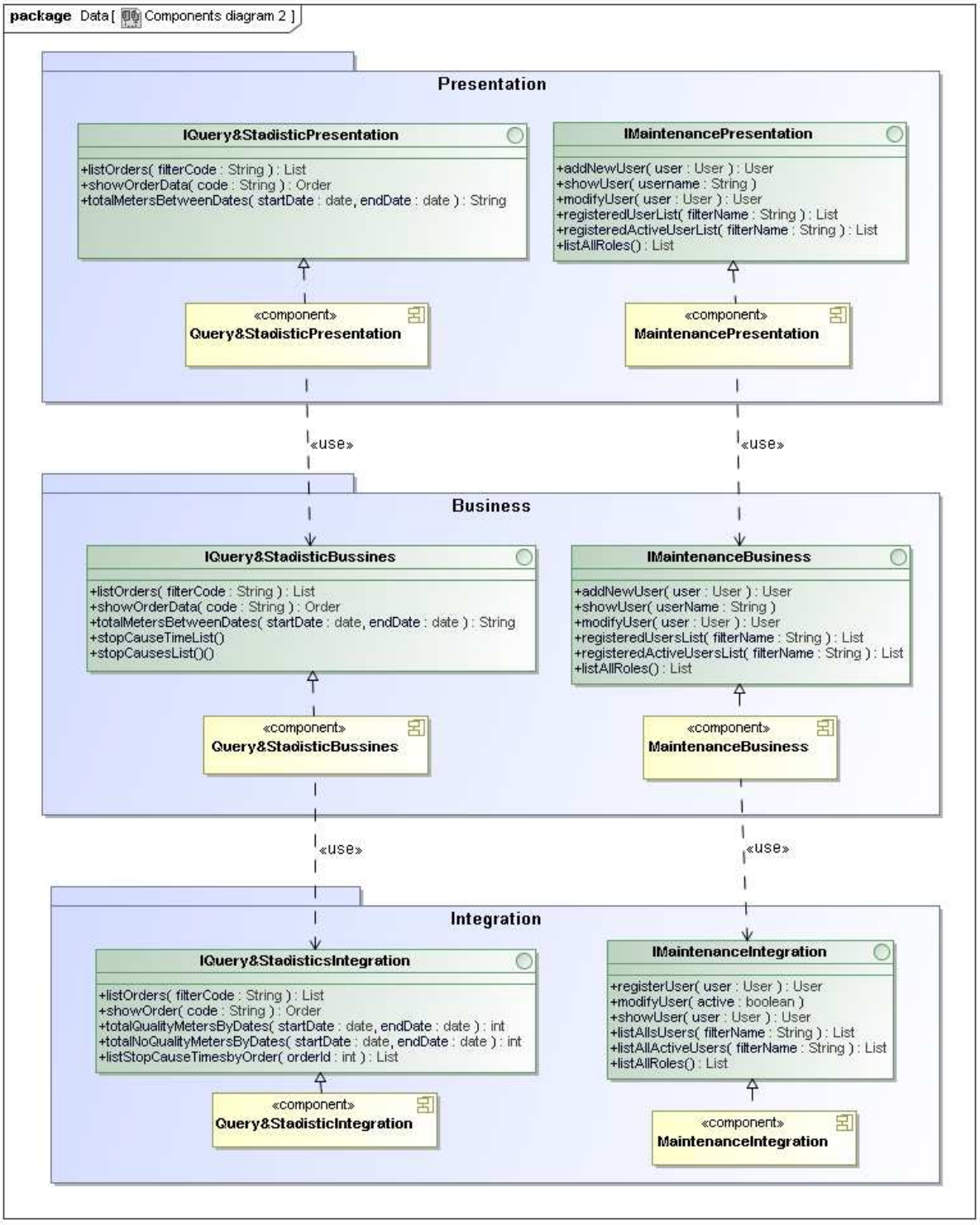

Figura 13. Diagrama de components (connexió i operació) 2

3.3.2. Refinament Component Operació a perfil Java EE capa Presentació

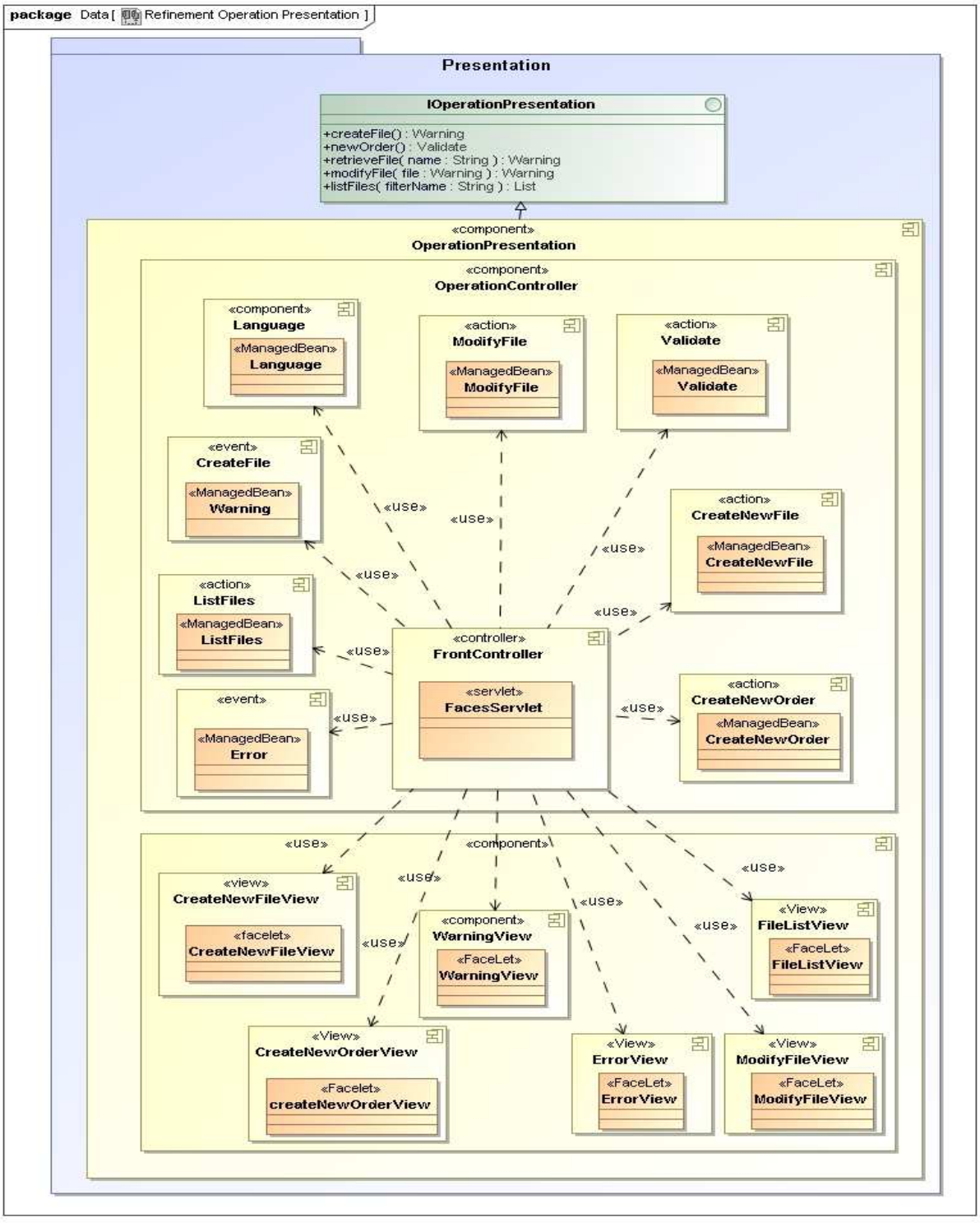

Figura 14. Refinament JavaEE Diagrama de components presentació

Com que es va decidir utilitzar el patró MVC (Model-Vista-Controlador) i que és implementada a la capa de negoci, s'ha obtingut un primer refinament del component de presentació separant el component que farà de controlador i els que faran de vistes.

S'ha creat un controlador per totes les operacions d'un component (Front controller) i que aquest no implementa cap operació, ja que és el responsable de coordinar tot el procés amb la implementació desacoblada.Cada acció que l'usuari pot realitzar per mitjà de la interfície d'usuari, és mapejada amb una vista i el controlador executa aquestes accions.

Qualsevol operació que produeixi un error o bé una excepció anirà a una vista genèrica d'error. El mateix amb els avisos i la vista Warning.

S'ha aplicat el perfil JavaEE que es va decidir a la primera fase del projecte.Per la capa de presentació utilitzem el framework Java Server Faces (JSF) i en un futur també utilitzarem Java Server Pages (JSP). El controlador no cal implementar-lo ja que ho farà el servlet Face Servlet que incorpora JSF.Les accions corresponents als Commands estaran definides amb Managed Bean.Finalment les vistes estaran implementades amb Facelets.

3.3.3 Refinament component Operació a perfil Java EE capa de Negoci.

En aquesta capa implementarem el patró façana.S'ha de donar tant un accés remot com accés local als components. A més les operacions que es realitzaran seran síncrones i sense estat ja que serà el contenidor Java EE qui es faci càrrec de les transaccions.

Per això s'ha decidit implementar el component de lògica de negoci amb EJB de sessió sense estat perquè així el contenidor resoldrà tota la complexitat de les comunicacions remotes, comportament transaccional i la gestió de la seguretat.

Arrel de totes d'aquestes decisions s'ha obtingut el següent refinament (pàgina següent).

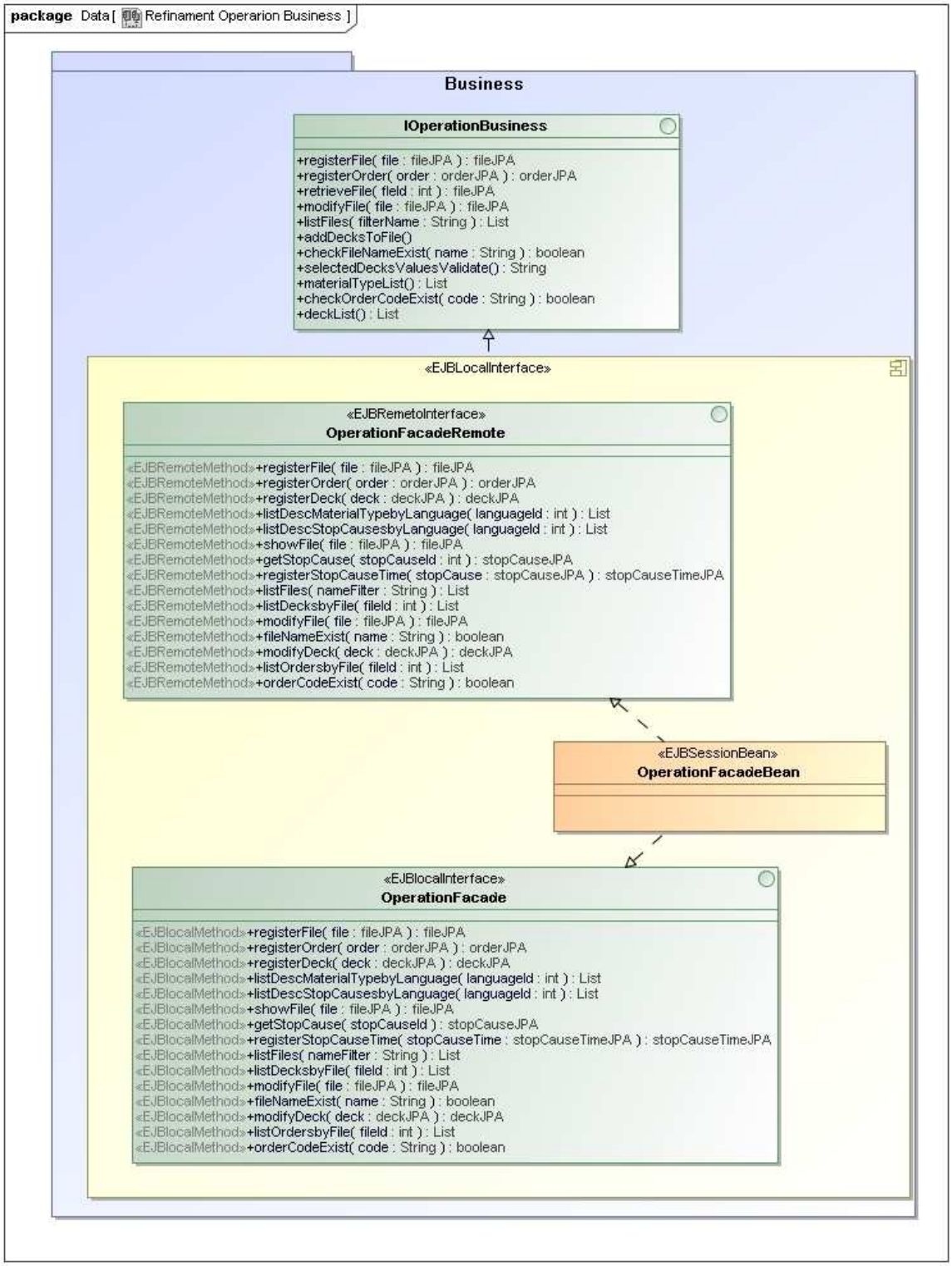

Figura 15. Refinament JavaEE Diagrama de components negoci

3.3.4 Refinament component Operació a perfil Java EE capa de Integració.

En aquesta capa s'ha optat implementar els components amb entitats JPA.Cada component estarà format per una entitat JPA.

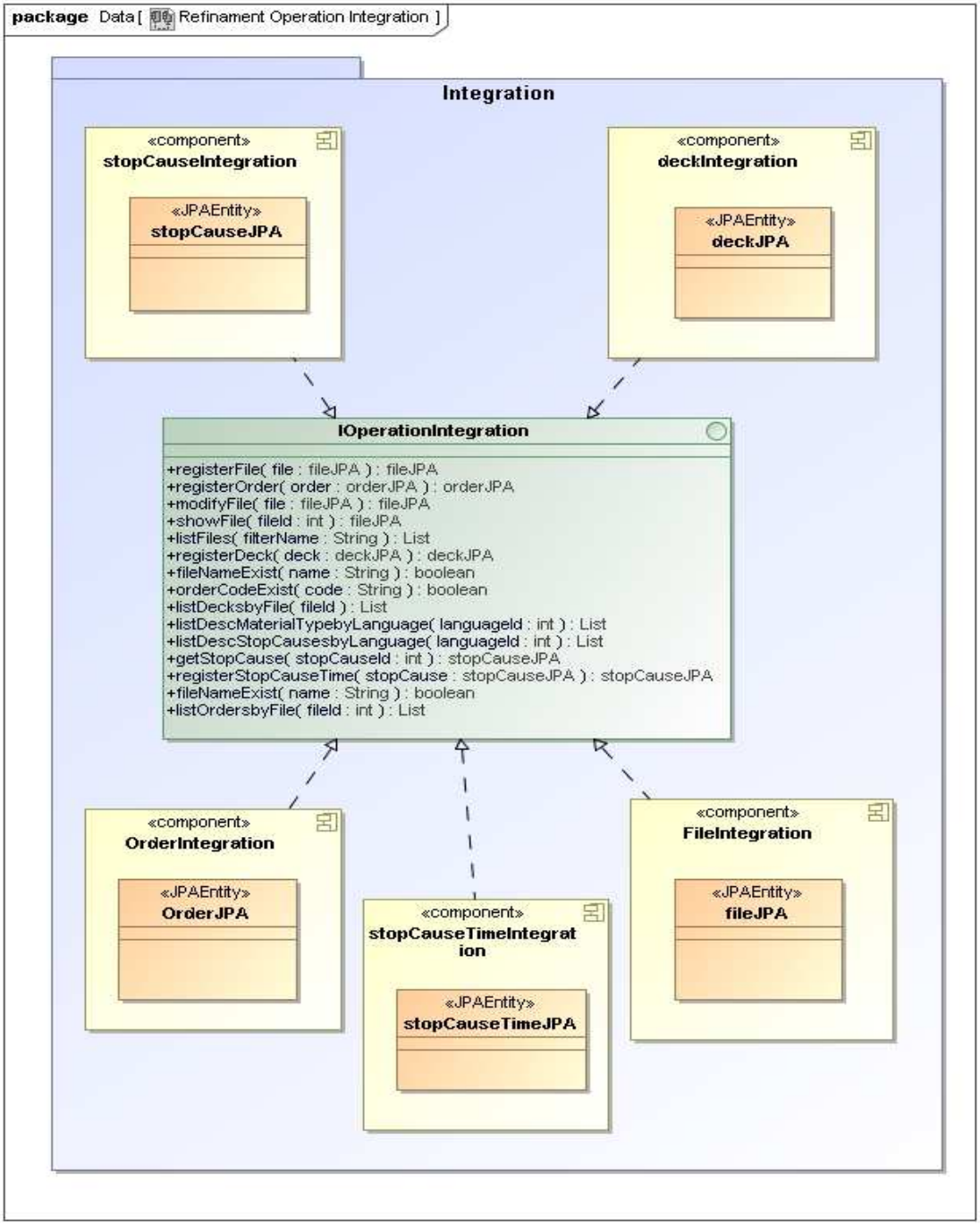

Figura 16. Refinament JavaEE Diagrama de components integració

3.3.5 Diagrama implementació SOA per futura aplicació mòbil.

En un futur no molt llunyà, voldrem implementar un aplicació nativa Android per una "tablet".Ho farem mitjançant serveis web (WS), amb el qual publicarem les consultes que un coordinador o un cap pot dur a terme des del seu dispositiu mòbil.Modificarem els EJB session existents (connection i query&stadistics) perquè a més de les funcionalitats que ofereix també permeti publicar el sevei Web.

Aquest servei web estarà basat en SOAP. La implementació client es farà sobre una aplicació android.

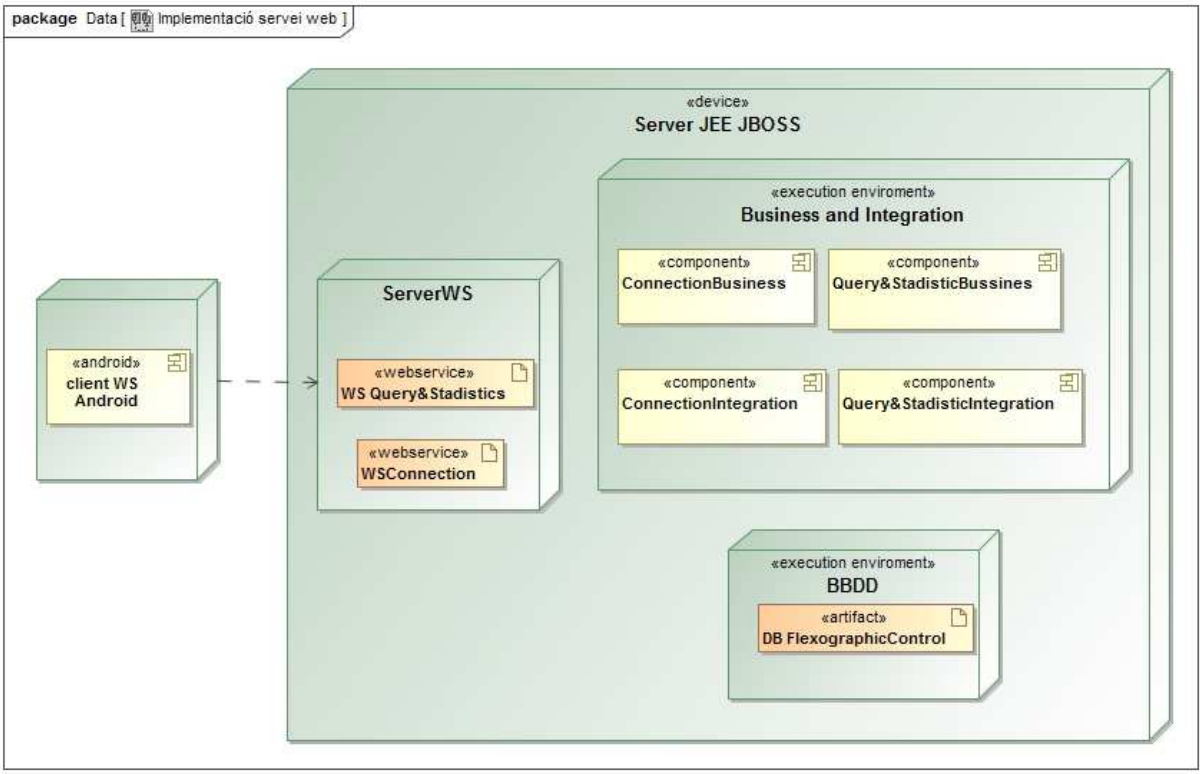

Figura 17. Implementació Web Service SOA per aplicació mòbil

3.4 Diagrama de seqüència Autenticar (Login)

Com s'ha comentat , s'han guardat els diagrames seqüència per la part de disseny ja que fa falta conèixer bé l'arquitectura de l'aplicació, i a més si es tracta d'una aplicació distribuïda. Amb els diagrames dinàmics de seqüència podem veure com l'usuari interactua amb l'aplicació.Mostren el pas de missatges entre entitats dins l'arquitectura de tres capes (presentació, lògica i integració).

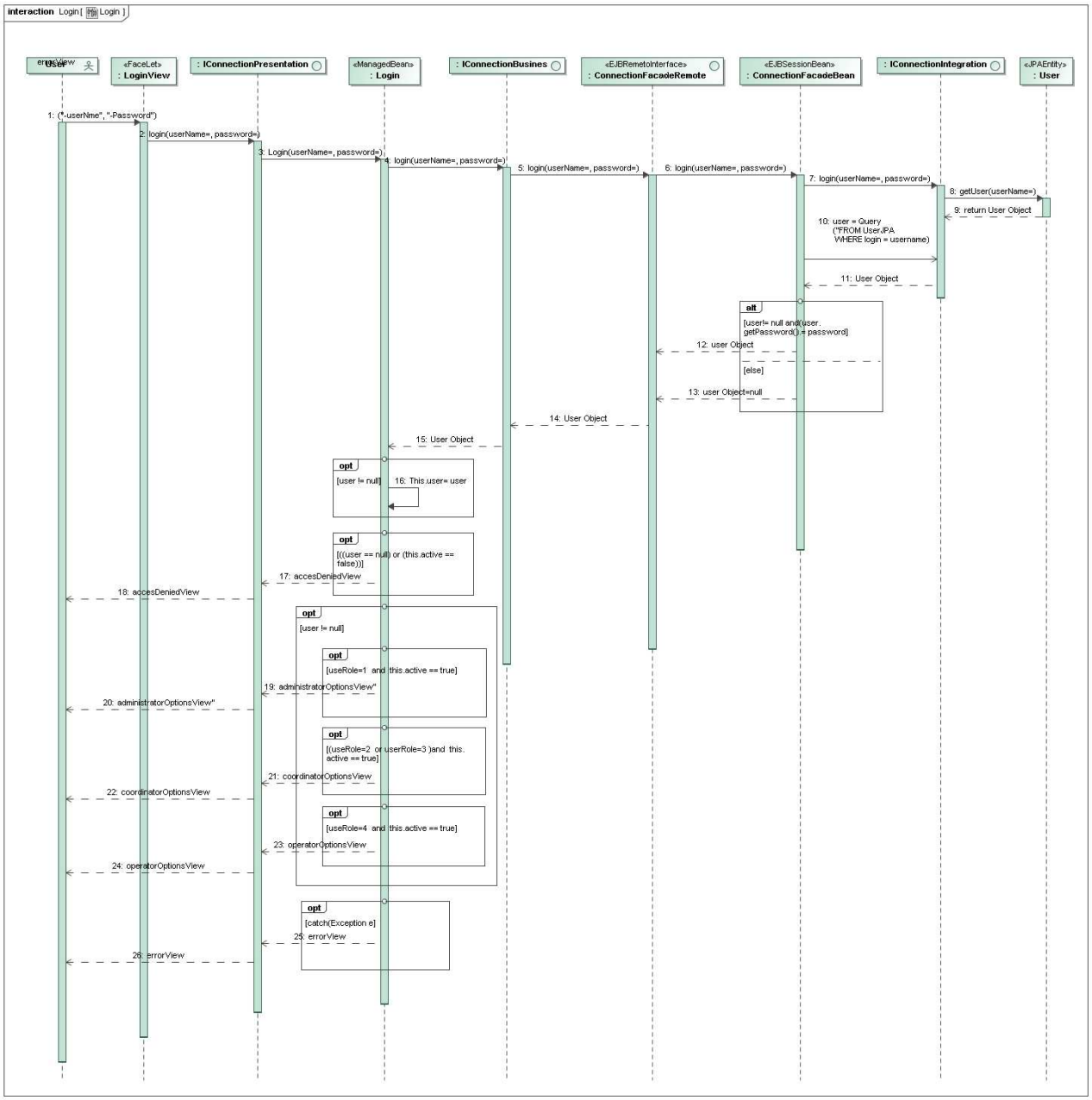

Figura 18. Diagrama de seqüència cas d'ús autenticar

3.5 Disseny Base da dades (Persistència)

Ja arribats en aquest punt, tenim prou informació per poder dissenyar la base de dades relacional, i amb cada taula obtinguda especificarem les columnes, la clau primària, les foranes i per cada clau forana a quina taula fa referència.També confeccionarem un diagrama on es mostrin les interrelacions entre taules.

Arrel del model conceptual que hem confeccionat a l'apartat anterior hem obtingut l'estructuració de la informació (entitats i relacions) de la lògica de l'aplicació i de la persistència de dades, que ha de coincidir.

L'únic que difereix de la persistència de dades respecte a la capa de lògica de programa, és la gestió de l'idioma estàtic.S'ha de crear una entitat Idioma que emmagatzemarà el codi de l'idioma i la seva descripció, i qualsevol entitat que contingui descripcions, es relacionarà amb idioma mitjançant una relació M-N i tindrà els atributs d'interrelació que són la descripció de l'atribut en qualsevol idioma.

Finalment s'ha confeccionat l'script que crea la base de dades més les entrades indispensables per tal de començar a utilitzar l'aplicació.Aquest script és el que s'executa des del SQL editor del pgAdminIII, que és l'administrador del postgreSQL .

3.5.1 Taules de la base de dades relacional

-Language (id, description);

-Role (Id,);

-User (Id, login, password, date_up, email, name,surnames, nif, active, date_down, role_type);
{ role_type } és clau forana a Role.

-Material (id, elastic);

-Desc_material_type (id_material_type, id_language, description);
{ id_material_type } és clau forana a Material.
{ id_language } és clau forana a Language.

-**File** (id, name, id_operator, press_number, creation_date, modified_date, repeat,
material_type, thickness, width, decks_temperature, tunnel_temperature,
unwinder_tension, outfeed_tensions, rewinder_tension, rewinder_taper);
{ id_operator } és clau forana a User
{ id_material_type} és clau forana a Material.

-**Deck** (id, id_file, number, repeat, color, pantone_reference, anilox_reference,
anilox_lineature, aniloxe_volume, viscosity);
{ id_file } és clau forana a File

-**Order** (id, id_file, order_name, reels_number, total_meters, metres_quality,
metres_no_quality);
{ id_file } és clau forana a File

-**Stop_cause_type** (id, group);

-**Desc_stop_cause_type** (id_stop_cause_type, id_language, description);
{ id_stop_cause_type } és clau forana a Stop_cause_type.
{ id_language } és clau forana a Language.

-**Stop_cause_time** (id_order, id_stop_cause_type, minutes);
{ id_stop_cause_type } és clau forana a Stop_cause_type.
{ id_order } és clau forana a Order.

3.5.2 Diagrama de la base de dades relacional

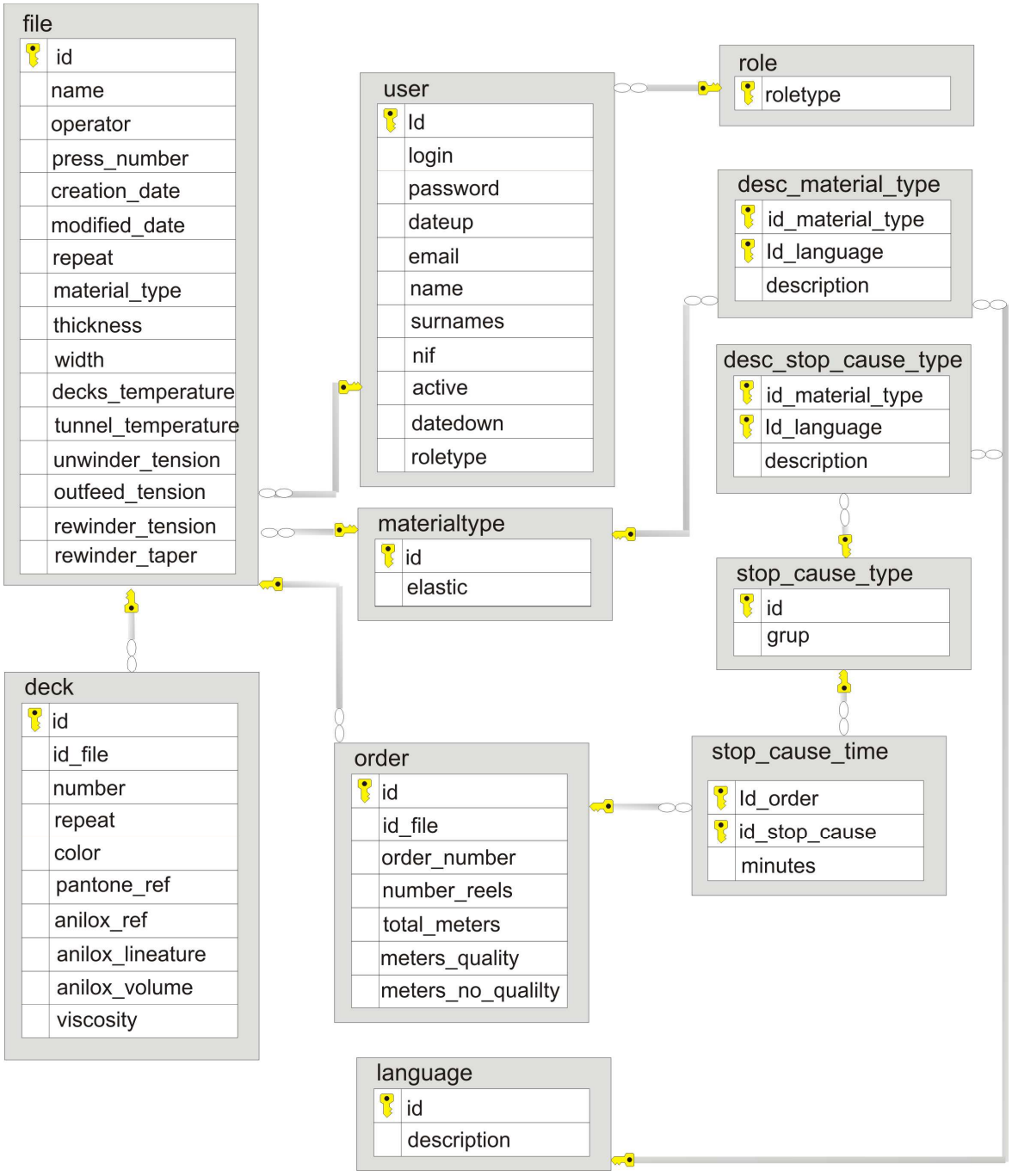

Figura 19. Diagrama base de dades relacional

3.5.3 Script de la base de dades.

Veiem a continuació com hem declarat aquestes taules, atributs, relacions i entrades inicials en un script que executem des de l'administrador del postgreSQL.

```
DROP TABLE IF EXISTS  flexographicmanagement.role CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.user CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.materialtype CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.file CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.language CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.desc_material_type CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.deck CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.order CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.stop_cause CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.stop_cause_time CASCADE;
DROP TABLE IF EXISTS  flexographicmanagement.desc_stop_cause CASCADE;


 CREATE TABLE flexographicmanagement.role (
   "role" text NOT NULL,
      CONSTRAINT "role_pkey" PRIMARY KEY (role)  );


CREATE TABLE flexographicmanagement.user (
   id integer NOT NULL,
   login character varying(20) NOT NULL UNIQUE,
   password character varying(15) NOT NULL,
   dateup date NOT NULL DEFAULT CURRENT_DATE,
   email character varying(30) NOT NULL,
   name character varying(30) NOT NULL,
   surnames character varying(30) NOT NULL,
   nif character varying(10) NOT NULL,
   active boolean NOT NULL,
   datedown date,
   role text NOT NULL,
   CONSTRAINT "user_pkey" PRIMARY KEY (id),
   FOREIGN KEY ("role")  REFERENCES flexographicmanagement.role (role) );


CREATE TABLE flexographicmanagement.materialtype (
   id integer NOT NULL,
   elastic boolean NOT NULL,
   CONSTRAINT "materialtype_pkey" PRIMARY KEY (id) );


CREATE TABLE flexographicmanagement.file (
   id integer NOT NULL,
   name character varying(30) NOT NULL,
   operator integer NOT NULL,
   pressnumber integer NOT NULL,
   creationdate date NOT NULL ,
   modificationdate date,
   repeat integer NOT NULL,
   materialtype integer NOT NULL,
   thickness integer NOT NULL,
   width integer NOT NULL,
   deckstemperature integer NOT NULL,
   tunneltemperature integer NOT NULL,
   unwindertension integer NOT NULL,
```

```
    outfeedtension integer NOT NULL,
    rewindertension integer NOT NULL,
    rewindertaper integer NOT NULL,
    CONSTRAINT "file_pkey" PRIMARY KEY (id),
    FOREIGN KEY ("operator")  REFERENCES flexographicmanagement.user (id),
    FOREIGN KEY ("materialtype")
    REFERENCES flexographicmanagement.materialtype (id)  );


CREATE TABLE flexographicmanagement.language (
    id integer NOT NULL,
    description character varying(20)NOT NULL,
    CONSTRAINT "language_pkey" PRIMARY KEY (id) );


 CREATE TABLE flexographicmanagement.desc_material_type (
    id integer NOT NULL,
    idmaterialtype integer NOT NULL,
    idlanguage integer NOT NULL,
    description character varying(20)NOT NULL,
    CONSTRAINT "desc_material_type_pkey" PRIMARY KEY (id ),
    FOREIGN KEY ("idmaterialtype")  REFERENCES flexographicmanagement.materialtype (id),
    FOREIGN KEY ("idlanguage")  REFERENCES flexographicmanagement.language (id));


CREATE TABLE flexographicmanagement.deck (
    id integer NOT NULL,
    file integer NOT NULL,
    number integer NOT NULL,
    pantone integer NOT NULL,
    aniloxref character varying(10)NOT NULL,
    aniloxlineature integer NOT NULL,
    aniloxvolume integer NOT NULL,
    viscosity integer NOT NULL,
    CONSTRAINT "deck_pkey" PRIMARY KEY (id),
    FOREIGN KEY ("file")  REFERENCES flexographicmanagement.file (id));


CREATE TABLE flexographicmanagement.order (
    id integer NOT NULL,
    file integer NOT NULL,
    creationdate date NOT NULL ,
    code character varying(15)NOT NULL,
    reelsnumber integer NOT NULL,
    totalmeters integer NOT NULL,
    metersquality  integer NOT NULL,
    metersnoquality  integer NOT NULL,
    CONSTRAINT "order_pkey" PRIMARY KEY (id),
    FOREIGN KEY ("file")  REFERENCES flexographicmanagement.file (id));


CREATE TABLE flexographicmanagement.stop_cause (
    id integer NOT NULL,
    type integer NOT NULL,
    CONSTRAINT "stop_cause_pkey" PRIMARY KEY (id));


CREATE TABLE flexographicmanagement.stop_cause_time (
        orderid integer NOT NULL,
        stopcause integer NOT NULL,
        minutes integer NOT NULL,
```

```
  CONSTRAINT "stop_cause_time_pkey" PRIMARY KEY (orderid, stopcause),
  FOREIGN KEY ("orderid")  REFERENCES flexographicmanagement.order (id),
  FOREIGN KEY ("stopcause")  REFERENCES flexographicmanagement.stop_cause (id));


CREATE TABLE flexographicmanagement.desc_stop_cause (
      idstopcause integer NOT NULL,
      idlanguage integer NOT NULL,
      description character varying(20)NOT NULL,
      CONSTRAINT "desc_stop_cause_pkey" PRIMARY KEY (idstopcause, idlanguage  ),
      FOREIGN KEY ("idstopcause")  REFERENCES flexographicmanagement.stop_cause (id),
      FOREIGN KEY ("idlanguage")  REFERENCES flexographicmanagement.language (id));



insert into flexographicmanagement.role (role) values ('1');
insert into flexographicmanagement.role (role) values ('2');
insert into flexographicmanagement.role (role) values ('3');
insert into flexographicmanagement.role (role) values ('4');

insert into flexographicmanagement.materialtype (id, elastic) values (1, true);
insert into flexographicmanagement.materialtype (id, elastic) values (2, true);
insert into flexographicmanagement.materialtype (id, elastic) values (3, false);
insert into flexographicmanagement.materialtype (id, elastic) values (4, false);
insert into flexographicmanagement.materialtype (id, elastic) values (5, false);

insert into flexographicmanagement.language (id, description) values (1, 'English' );
insert into flexographicmanagement.language (id, description) values (2, 'Spanish' );
insert into flexographicmanagement.language (id, description) values (3, 'Catalan' );

insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (1, 1, 1, 'Polytilene');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (2, 1, 2, 'Politileno');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (3, 1, 3, 'Politilé');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (4, 2, 1, 'Polytilene Hd');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (5, 2, 2, 'Politileno Hd');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (6, 2, 3, 'Politilé Hd');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (7, 3, 1, 'Polypropylene');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (8, 3, 2, 'polipropleno');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (9, 3, 3, 'polipropié');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (10, 4, 1, 'Paper');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (11, 4, 2, 'Papel');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (12, 4, 3, 'Paper');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (13, 5, 1, 'Aluminium');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (14, 5, 2, 'Aluminio');
insert into flexographicmanagement.desc_material_type (id, idmaterialtype,idlanguage, description) values (15, 5, 3, 'Alumini');

insert into flexographicmanagement.stop_cause ( id, type) values (1, 1);
insert into flexographicmanagement.stop_cause ( id, type) values (2, 1);
insert into flexographicmanagement.stop_cause ( id, type) values (3, 1);
insert into flexographicmanagement.stop_cause ( id, type) values (4, 1);
insert into flexographicmanagement.stop_cause ( id, type) values (5, 1);
insert into flexographicmanagement.stop_cause ( id, type) values (6, 2);
insert into flexographicmanagement.stop_cause ( id, type) values (7, 2);
insert into flexographicmanagement.stop_cause ( id, type) values (8, 2);
insert into flexographicmanagement.stop_cause ( id, type) values (9, 2);
insert into flexographicmanagement.stop_cause ( id, type) values (10, 2);
insert into flexographicmanagement.stop_cause ( id, type) values (11, 3);
insert into flexographicmanagement.stop_cause ( id, type) values (12, 3);
insert into flexographicmanagement.stop_cause ( id, type) values (13, 3);
insert into flexographicmanagement.stop_cause ( id, type) values (14, 3);
insert into flexographicmanagement.stop_cause ( id, type) values (15, 3);
```

```
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 1, 1,'Setup');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 1, 2,'Ajuste Inicial');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 1, 3,'Ajust Inicial');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 2, 1,'Timetable');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 2, 2,'Horario Fábrica');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 2, 3,'Horari Fábrica');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 3, 1,'Change Job');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 3, 2,'Cambio Trabajo');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 3, 3,'Canvi Treball');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 4, 1,'Plate Cleaning');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 4, 2,'Limpieza Cliché');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 4, 3,'Neteja Clitxé');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 5, 1,'No Ink');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 5, 2,'Falta Tinta');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 5, 3,'Falta Tinta');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 6, 1,'Adjustments');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 6, 2,'Ajustes');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 6, 3,'Ajustos');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 7, 1,'Anilox Change');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 7, 2,'Cambio Anilox');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 7, 3,'Canvi Anilox');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 8, 1,'Dr. Balde Change');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 8, 2,'Cambio Rasqueta');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 8, 3,'Canvi Rasqueta');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 9, 1,'No Material');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 9, 2,'Falta Material');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 9, 3,'Falta Material');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 10, 1,'Splice Fail');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 10, 2,'Fallo Cambio');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 10, 3,'Fallo Canvi');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 11, 1,'Maintenance');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 11, 2,'Mantenimiento');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 11, 3,'Manteniment');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 12, 1,'Emergency Stop');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 12, 2,'Parada Emergéncia');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 12, 3,'Parada Emergència');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 13, 1,'Break Material');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 13, 2,'Rotura Material');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 13, 3,'Trencada Material');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 14, 1,'Operator Stepout');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 14, 2,'Descanso Operario');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 14, 3,'Descans Operari');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 15, 1,'Webpath');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 15, 2,'Enfilar Material');
insert into flexographicmanagement.desc_stop_cause ( idstopcause, idlanguage, description)  values ( 15, 3,'Enfilar Material');
```

```
insert into flexographicmanagement.user(id ,login, password, dateup, email, name, surnames, nif, active, datedown, role )
values (1,'msegura','abril99' ,'01-12-2010','marc.segura@comexigroup.com','Marc','Segura Valls','436772266G', true ,null, '1');
insert into flexographicmanagement.user(id, login, password, dateup, email, name, surnames, nif, active, datedown, role )
values (2,'jmolas','pandorarules' ,'01-10-2008','josep.molas@comexigroup.com','Josep','Molas Valls','14456722T', true ,null, '2');
insert into flexographicmanagement.user(id, login, password, dateup, email, name, surnames, nif, active, datedown, role )
values (3,'mjonas','nilemo' ,'11-08-2010','miquel.jonas@comexigroup.com','Miquel','Jonas Ulbert','34566678G', true ,null, '3');
insert into flexographicmanagement.user(id, login, password, dateup, email, name, surnames, nif, active, datedown, role )
values (4,'llfontas','printoac' ,'20-08-2011','lluis.fontas@comexigroup.com','Lluís','Fontàs Pallarols','23341123T', true ,null, '4');
insert into flexographicmanagement.user(id, login, password, dateup, email, name, surnames, nif, active, datedown, role )
values (5,'jroure','pinatellrules' ,'20-08-2009','joan.roure@comexigroup.com','Joan','Roure Pirlu','89721155D', true ,null, '4');
```

3.6 Interfície Gràfica

Ja clares i desgranades totes les funcionalitats a partir de l'anàlisi i els dissenys estàtics i dinàmics, s'han confeccionat les pantalles que formen par de la interfície gràfica. No posem aquí tots els missatge d'error, excepcions i de verificacions.

Com ja s'ha comentat, a la primera alliberació d'aquest projecte hi haurà només una plataforma web, i en una pròxima fase una plataforma mòbil Android.Per tant, s'han realitzat també els prototips per l'aplicació mòbil.

3.6.1 Pantalla Autenticació (Login)

A més en aquesta pantalla es podrà seleccionar el idioma de l'aplicació.

Figura 20. Pantalla autenticació

3.6.2 Pantalla Canviar Contrasenya.

Figura 21. Pantalla canvi de contrasenya

3.6.3 Pantalla Principal

Aquesta pantalla ofereix un menú d'opcions/permisos que l'usuari pot fer en funció del seu rol en el sistema.Vegem l'exemple del menú del rol de l'administrador.

Figura 22. Menú principal (exemple perfil administració)

3.6.4 Pantalla Crear fitxa i Modificar fitxa

La pantalla crear fitxa és molt semblant a la de Mostar/Modificar fitxa només difereixen algunes dades com que nomes sortiran els tinters que s'utilitzen (que tenen desenvolupament).També hi han missatges de la pantalla avis per si no s'ha seleccionat cap tinter o bé no s'ha omplert algun camp d'algun tinter seleccionat.

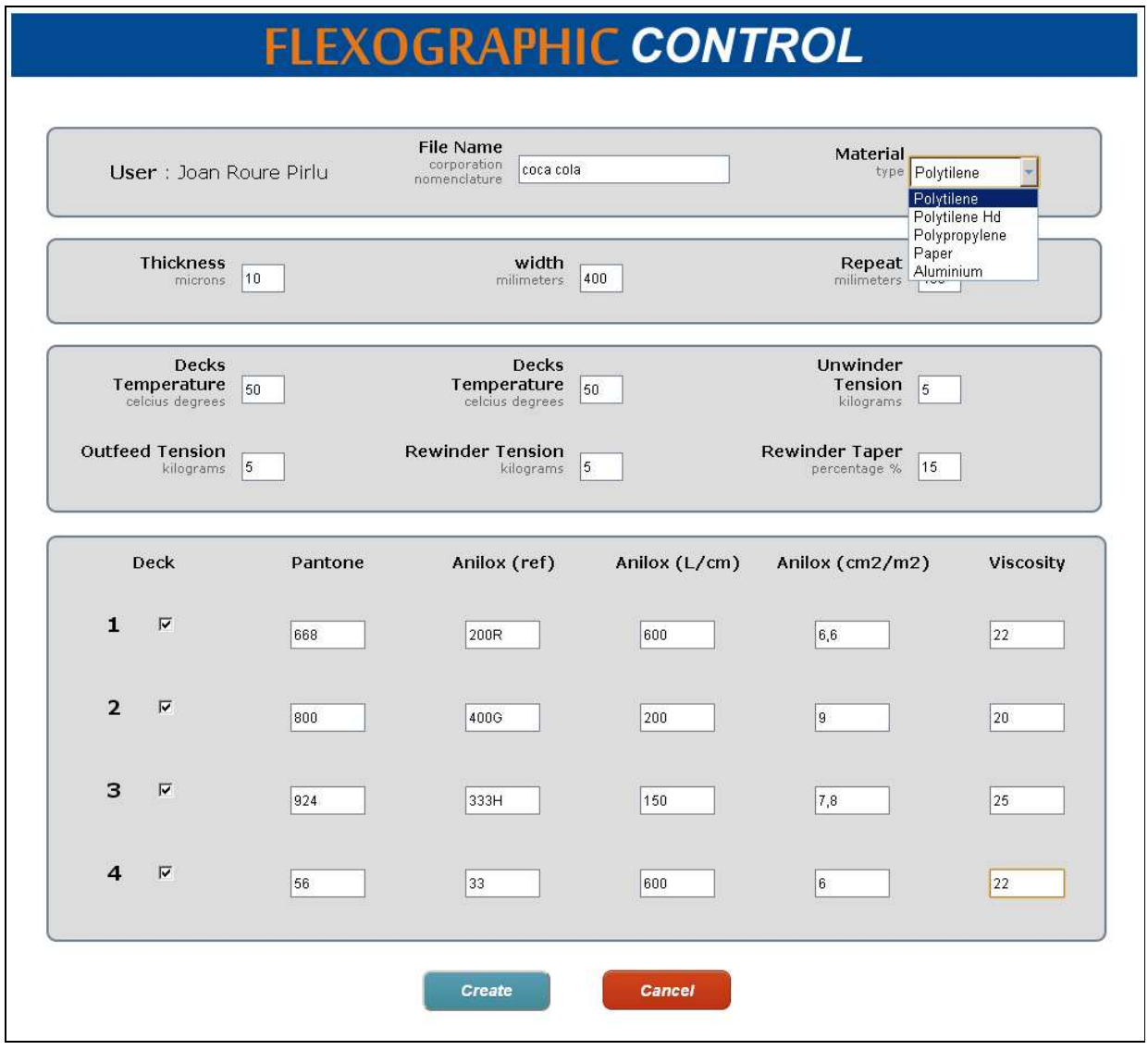

Figura 23. Pantalla crear fitxa

Figura 24. Missatges avís pantalla crear fitxa

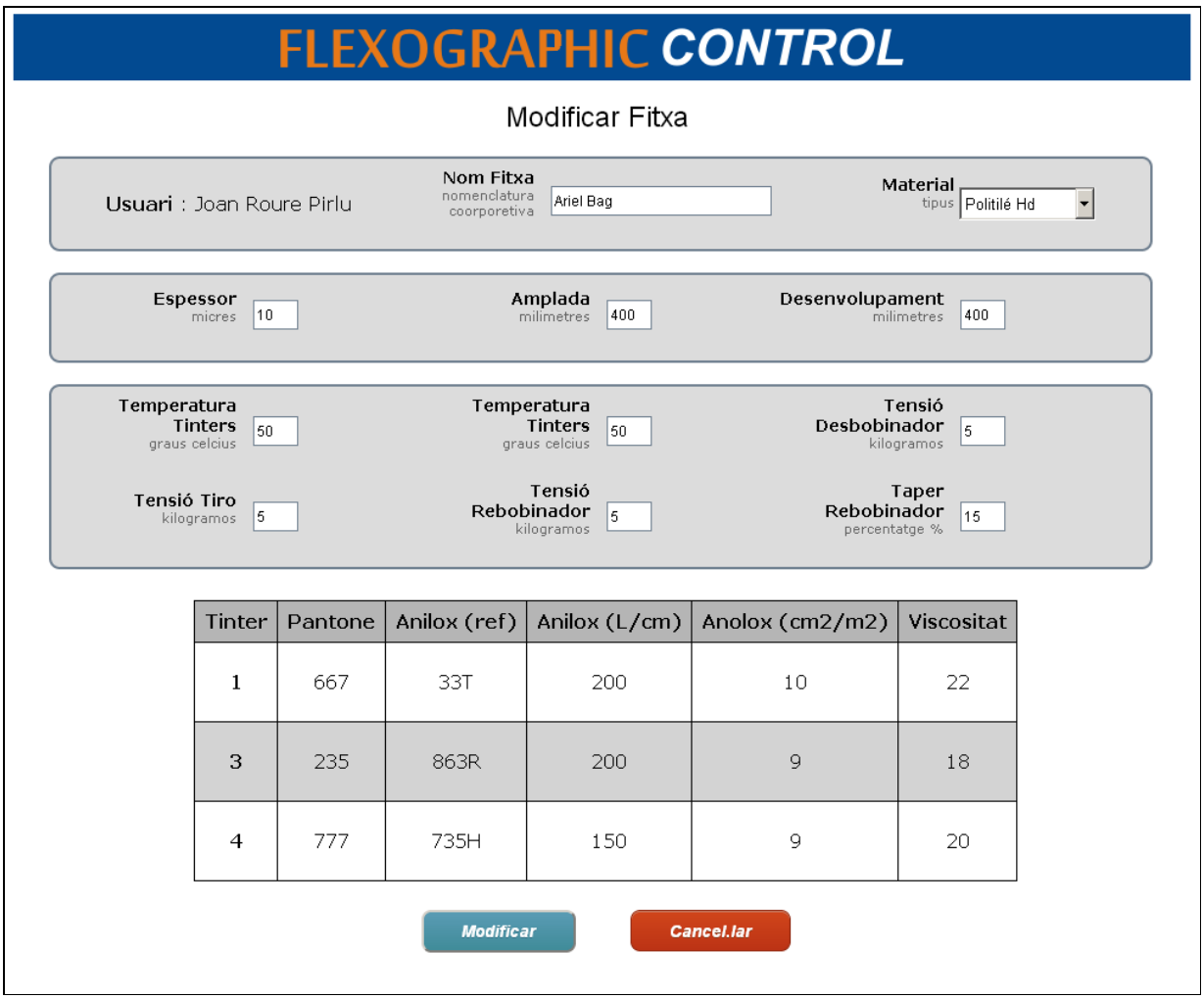

Figura 25. Pantalla mostrar/modifica fitxa

Figura 26. Avís fitxa existent.

3.6.5 Pantalla llistar fitxes

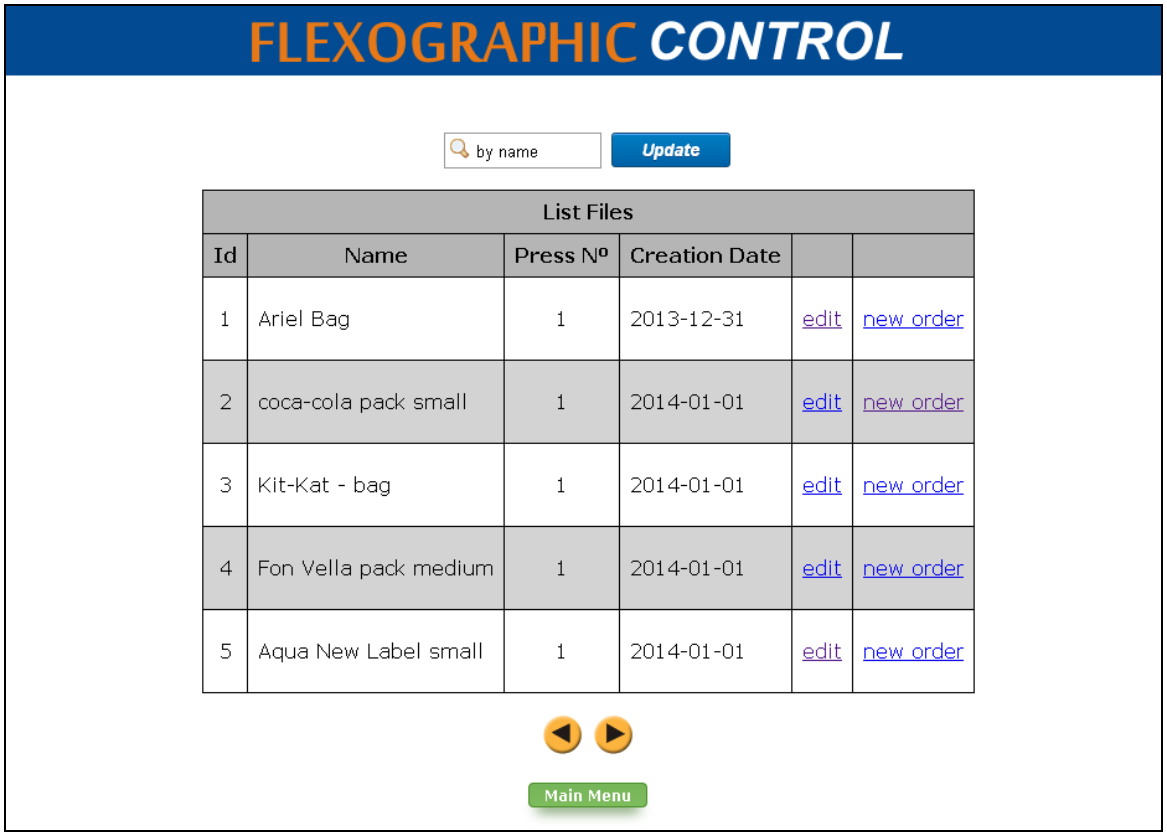

Figura 27. Pantalla Llistar fitxes.

3.3.6 Pantalles Nova comanda i dades comanda

Quan es fa una fitxa per primera vegada, un cop enviada el sistema demana a l'usuari per el codi d'aquesta comanda.

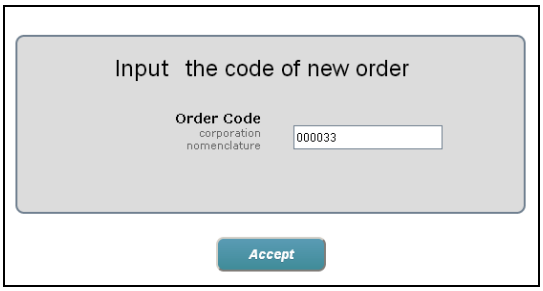

Figura 28. Pantalla primera comanda.

Com que la pantalla de crear comanda és quasi idèntica a la de mostrar les dades d'una comanda que utilitza el coordinador, se'n mostra una.

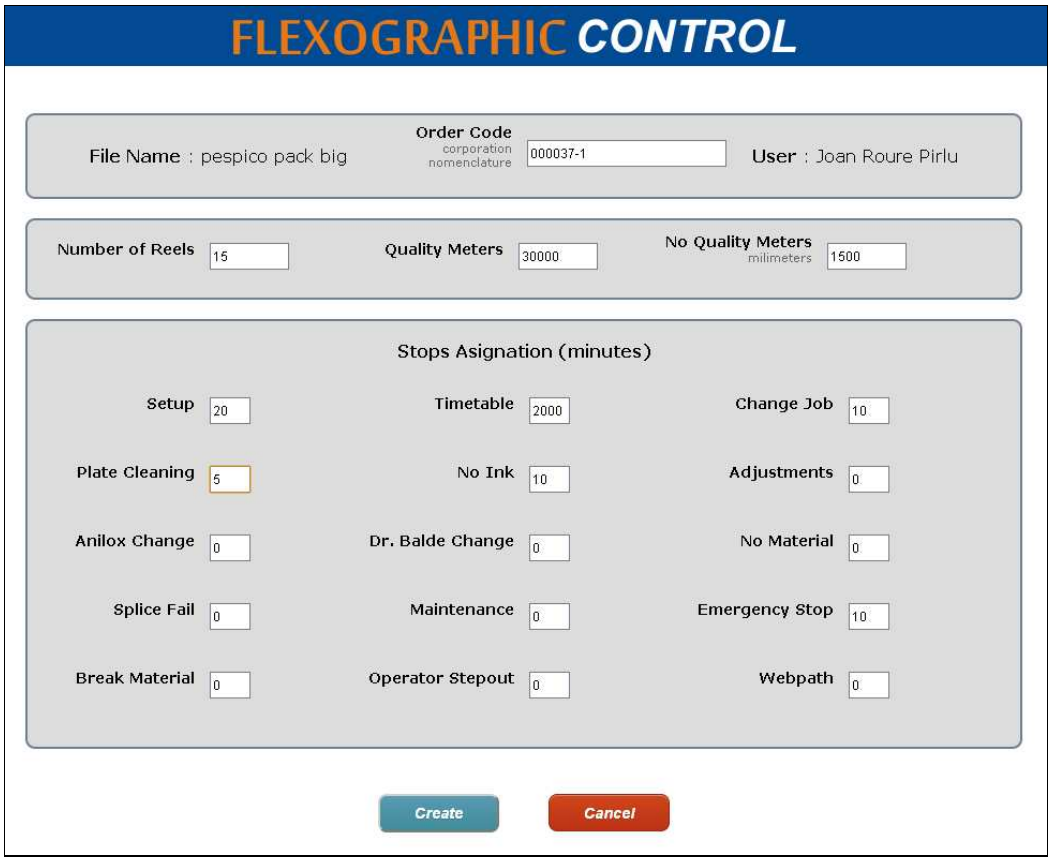

Figura 29. Pantalla Crear Comanda i Mostrar Comanda.

3.3.7 Pantalla Llistar comandes

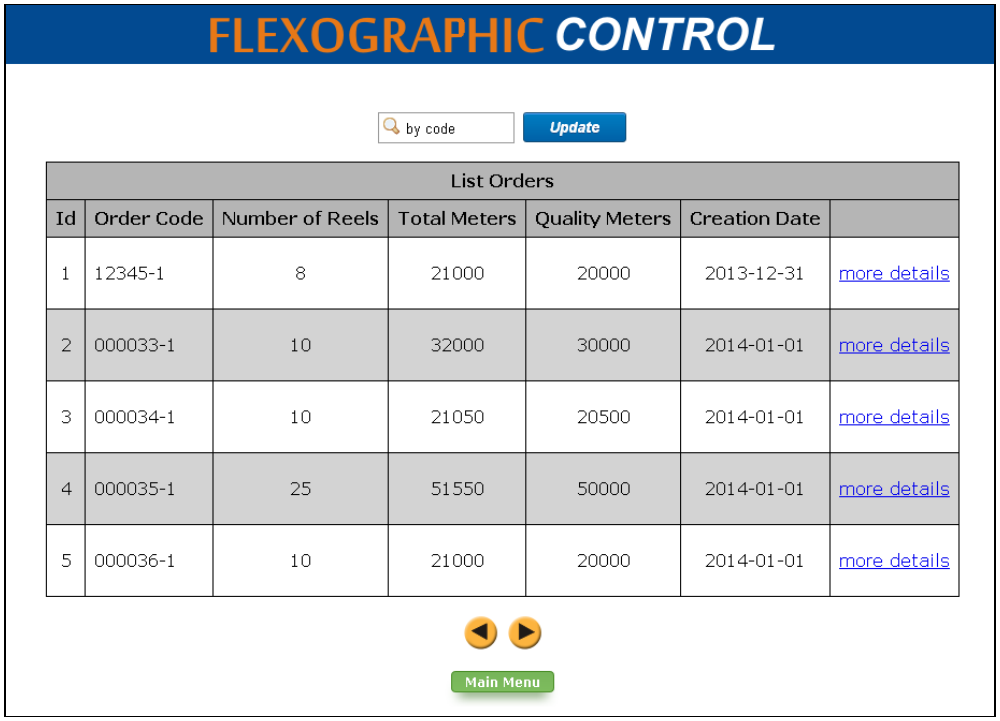

Figura 30. Pantalla Llistar Comandes.

3.3.8 Pantalla Consulta metres totals entres dues dates

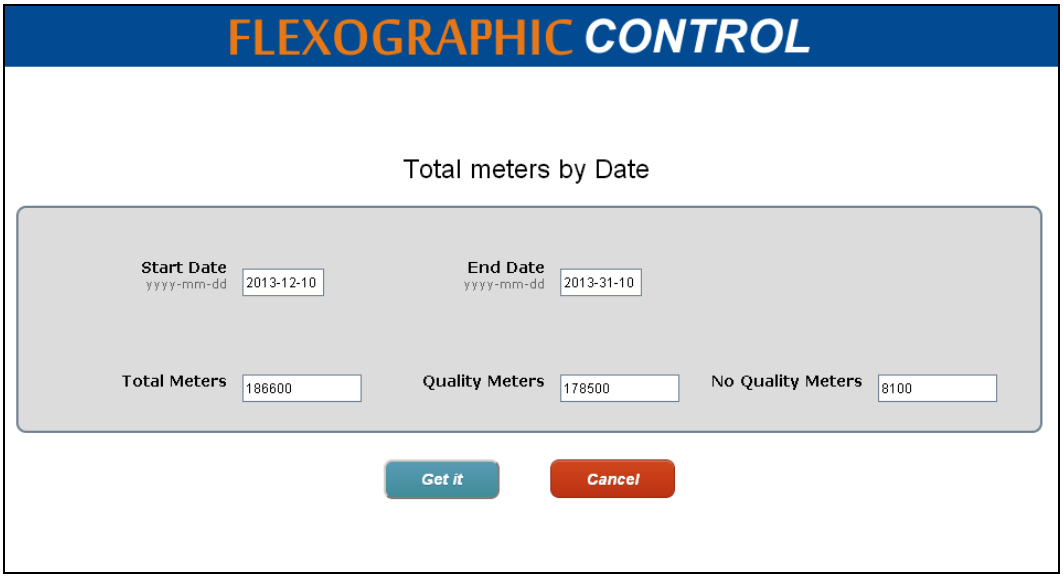

Figura 31. Consulta metres totals entre dates.

3.3.9 Pantalla crear usuari i modificar usuari.

La pantalla de mostrar i modificar difereix molt poc de la de crear i només es mostra aquesta última.

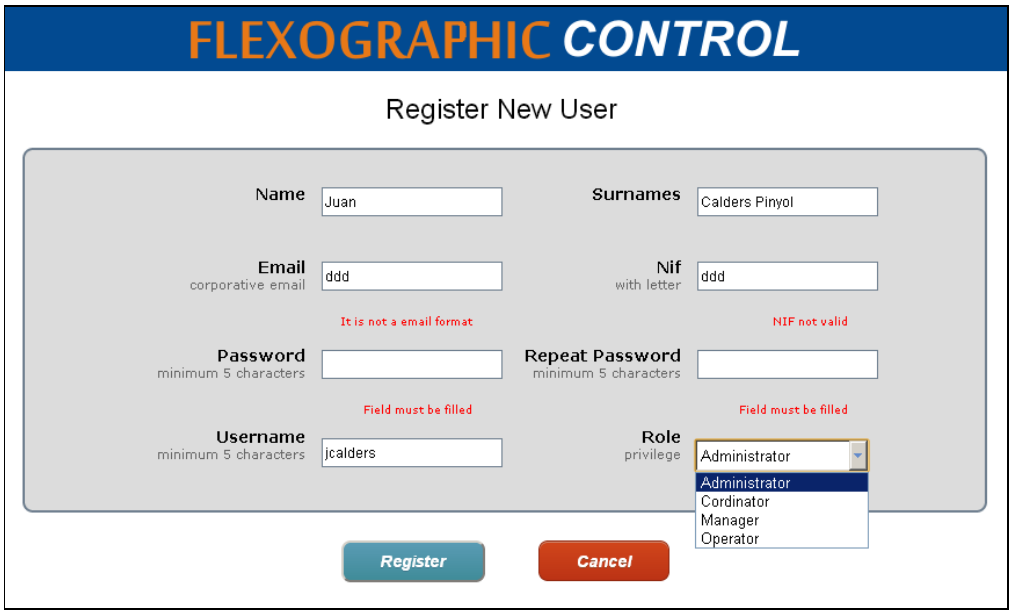

Figura 32. Crear usuari.

3.3.10 Llistar usuaris (per donar de baixa i modificar)

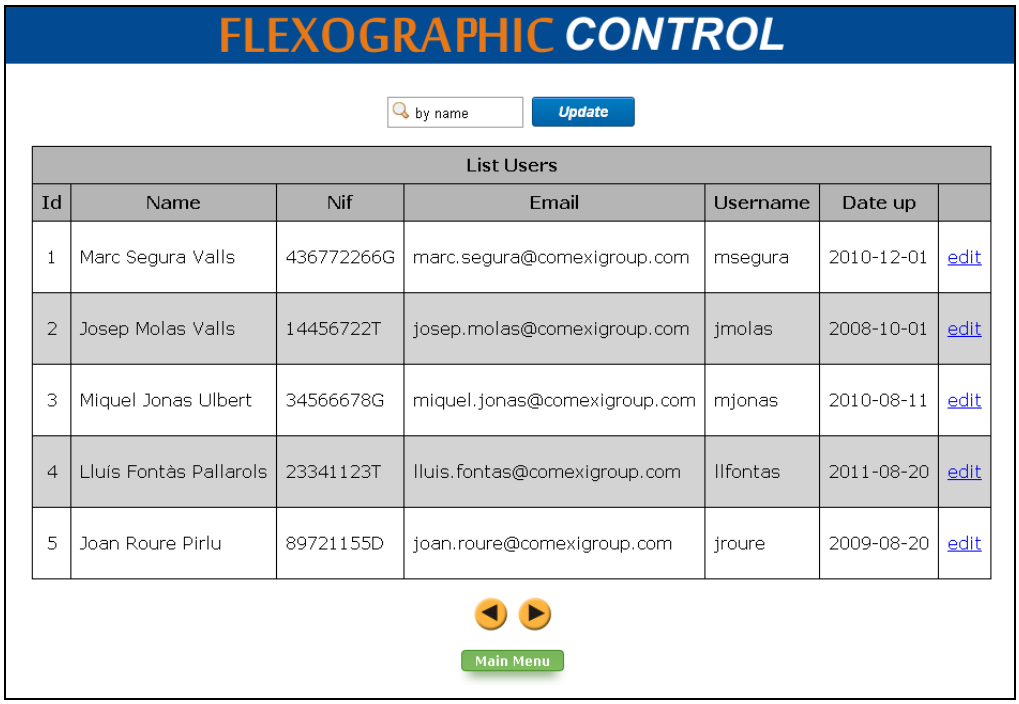

Figura 33. Pantalla llistar usuaris.

El cas que sigui la pantalla llistar usuaris actius per donar-ne un de baixa, un cop seleccionat l'usuari en qüestió el sistema mostrarà el missatge següent

Figura 34. Missatge donar de baixa usuari.

3.3.11 Prototips pantalles futura aplicació per dispositiu mòbil

Figura 35. Prototip pantalles Login i menú aplicació mòbil.

Figura 36. Prototip pantalles llistar fitxes/comandes i dades comanda aplicació mòbil.

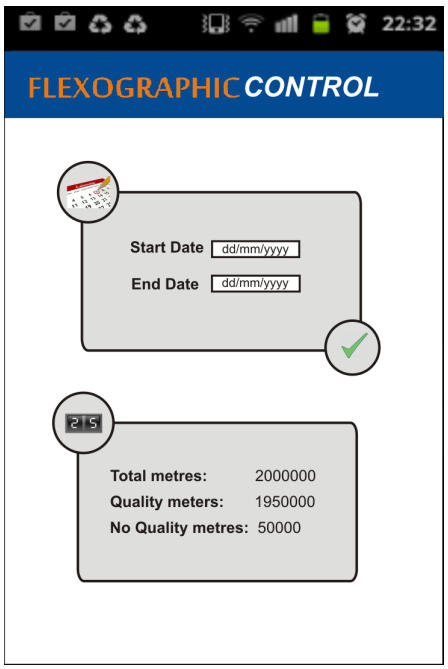

Figura 37. Prototip pantalla consulta metres totals aplicació mòbil.

4. Implementació

En aquest apartat s'explica com s'ha implementat l'aplicació, mostrant la seva estructura i exemples de com s'han codificat vistes, EJB, ManagedBean,etc... També com s'ha fet la gestió dels idiomes i les validacions.

4.1 Implementació arquitectura client-servidor tres capes

Gracies a al estudi i disseny previ, la implementació del projecte s'ha fet d'una manera clara i estructurada, que permet aplicar tots els avantatges i prestacions que ofereix l'arquitectura client/servidor amb tres capes.A les figures següents es mostra tal com ha quedat l'estructura del nostre model implementat en el nostre projecte creat amb l'IDE Ecplise. S'ha separat per package els EJB , les entitats JPA i els Managed Beans. A més els JSF queden aïllats per pàgines .xhtml a la carpeta docroot. El package TO de moment no té res a veure amb el projecte actual, ja que està pensat en un futur immediat utilitzar serveis web per aplicació mòbil, i per porta-ho a termes necessita fer una traducció de les entitats JPA.

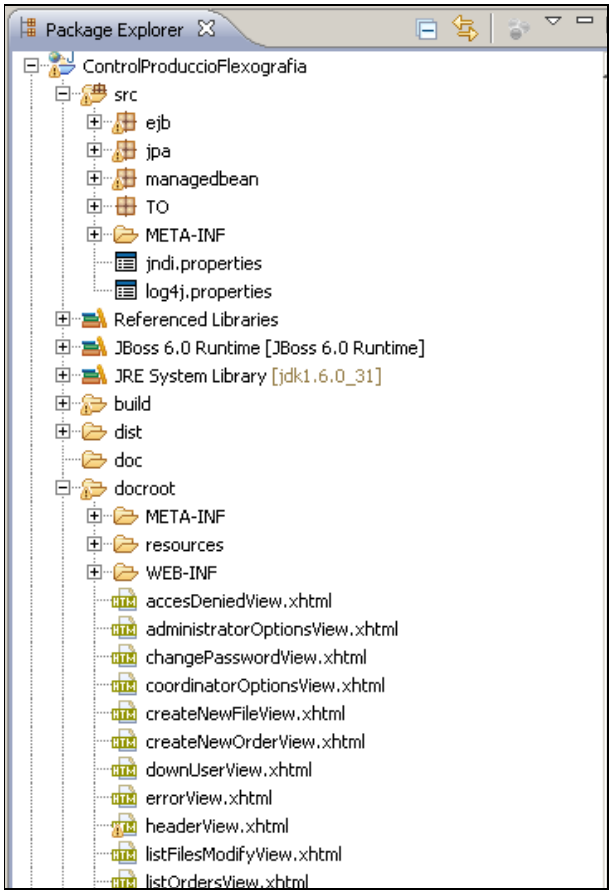

Figura 38. Estructura implementació projecte amb Eclipse

La figura següent desglossa els paquets i mostra tal com han quedat organitzats els quatre subsistemes i les seves façanes.També totes les entitats JPA i els ManagedBean.

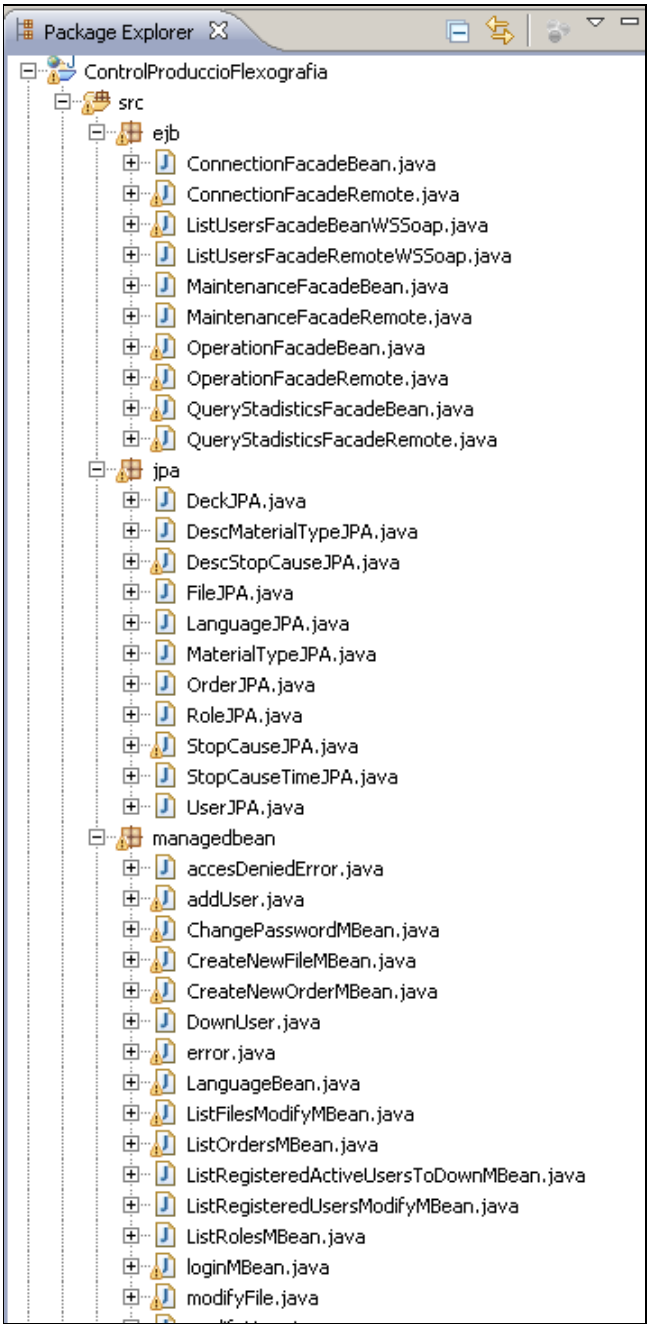

Figura 39. Estructura implementació projecte amb Eclipse desglossada

4.2 Exemples Codificacions

És important també fixar-nos com s'han codificat els elements dels principals frameworks utilitzats i també altres aspectes importants de l'aplicació com la gestió del idioma i les validacions.

4.2.1 Codificació vistes Facelets + HTML + CSS

Com s'ha comentat en el primer apartat, amb els Facelets elaborem la interfície amb l'usuari juntament amb HTML (HiperText Markup Language) i CSS (Cascading Style Sheets). Cal destacar la importància de la fulla d'estils CSS , ja que ha agafat tota la responsabilitat del disseny i des de la fulla d'estils es controlen els efectes d'animació i disseny.Aquesta fulla d'estils la invoquem des de la capçalera, que és una pàgina omnipresent.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=http://www.w3.org/1999/xhtml
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:f="http://java.sun.com/jsf/core" xmlns:h="http://java.sun.com/jsf/html">

<h:head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
       <h:outputStylesheet library="css" name="styles.css" />
      <title>FlexoGraphic Control</title>
</h:head>

<h:body>
      <div id="layout">
            <div id="top" class="top">
      <ui:insert name="top">
            <table width="100%" cellspacing="0" border="0"
            background="#{resource['images:header_logo_background.png']}">
              <tr>
                  <td width="30"><img border="0"
                  src="#{resource['images:header_logo.png']}"/></td>
              </tr>
            </table>

      </ui:insert>
            </div>
            <div id="content">
      <           ui:insert name="content">Content</ui:insert>
            </div>
      </div>
</h:body>
</html>
```

Aquesta fulla d'estils es troba dins la carpeta resources on estan ubicades totes les vistes.En aquesta carpeta també s'hi guarden totes les imatges.

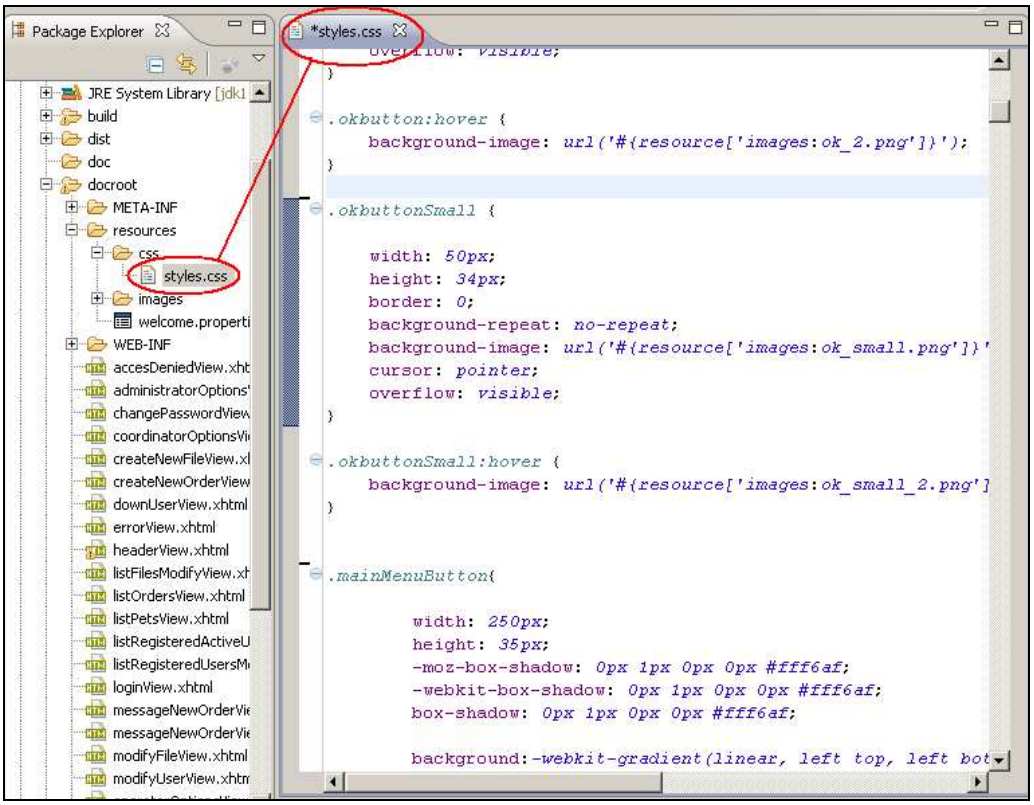

Figura 40. Fulla d'estils CSS

Seguidament tenim un exemple de com s'ha codificat una vista (pàgina xhtlm). Aquesta es tracta de la vista crear nou usuari (figura32), i les altres segueixen la mateixa mecànica, amb diferències com per exemple de les vistes que mostres llistes tenen la instrucció dataTable

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=http://www.w3.org/1999/xhtml
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:f="http://java.sun.com/jsf/core" xmlns:h="http://java.sun.com/jsf/html"
xmlns:p="http://primefaces.org/ui">
<f:metadata>
      <f:viewParam name="idCustomer" value="#{registeredCustomershow.idCustomer}"/>
</f:metadata>

<ui:composition template="./headerView.xhtml">

<ui:define name="content">
      <p align="center" >
      <h:outputText value="#{language.init}" style="padding-
      top:2px;color:White;font-size:xx-small;"/>
      <h:outputText value="#{addUser.init}" style="padding-
      top:2px;color:White;font-size:xx-small;"/>
```

```
        </p>
        <br/>

        <h:outputText align="center" value="#{msg['new.user']}"  style="font-size:x-
        large;font-family:Arial;"/>
        <br/><br/>

<f:view>
<h:form >

<div id="form">

<table align="center">
        <tr>
                <td><label><h:outputText value="#{msg['name']}" /></label></td>
                <td><h:inputText id="name" value="#{addUser.name}"
                validator="#{validate.validate}" size="20" maxlength="30"
                styleClass="input"></h:inputText>
                <span class="validate"><h:message for="name"
                style="color:Red;"></h:message></span> </td>

                <td><label><h:outputText value="#{msg['surnames']}" /></label></td>
                <td><h:inputText id="surnames" value="#{addUser.surNames}"
                validator="#{validate.validate}" size="20" maxlength="40"
                styleClass="input"></h:inputText>
                <span class="validate"> <h:message for="surnames"
                style="color:Red;"></h:message> </span></td>
        </tr>

        <tr>
                <td><label><h:outputText value="#{msg['email']}" />
                <span class="detail"><h:outputText
                value="#{msg['corporative.email']}"/></span></label></td>
                <td><h:inputText id="email" value="#{addUser.email}"
                validator="#{validate.validate}" size="20" maxlength="40"
                styleClass="input"></h:inputText>
                <span class="validate"> <h:message for="email"
                style="color:Red;"></h:message> </span></td>

                <td><label><h:outputText value="#{msg['nif']}" />
                <span class="detail"><h:outputText
                value="#{msg['with.letter']}"/></span></label></td>
                <td><h:inputText id="nif" value="#{addUser.nif}"
                validator="#{validate.validate}" size="9" maxlength="9"
                styleClass="input"></h:inputText>
                <span class="validate"> <h:message for="nif"
                style="color:Red;"></h:message> </span></td>
        </tr>

        <tr>
                <td><label><h:outputText value="#{msg['password']}" />
                <span class="detail"><h:outputText
                value="#{msg['minimum.five.caracters']}"/></span></label></td>
                <td><h:inputSecret id="password" value="#{addUser.password}"
                validator="#{validate.validate}" size="20" maxlength="40"
                styleClass="input"></h:inputSecret>
                <span class="validate"><h:message for="password"
                style="color:Red;"></h:message>  </span></td>

                <td><label><h:outputText value="#{msg['repeat.password']}" />
                <span class="detail"><h:outputText
                value="#{msg['minimum.five.caracters']}"/></span></label></td>
                <td><h:inputSecret id="passwordRetyped"
                value="#{addUser.passwordRetyped}"
```

```
                validator="#{validate.validate}" size="20" maxlength="40"
                styleClass="input"></h:inputSecret>
                <span class="validate"><h:message for="passwordRetyped"
                style="color:Red;"></h:message>  </span></td>

        </tr>

        <tr>
                <td><label><h:outputText value="#{msg['username']}" />
                <span class="detail"><h:outputText
                value="#{msg['minimum.five.caracters']}"/></span></label></td>
                <td><h:inputText id="username" value="#{addUser.login}"
                validator="#{validate.validate}" size="20" maxlength="30"
                styleClass="input"></h:inputText>
                <span class="validate"><h:message for="username"
                style="color:Red;"></h:message>  </span></td>

                <td><label><h:outputText value="#{msg['role']}" />
                 <span class="detail"><h:outputText
                value="#{msg['privilege']}"/></span></label></td>
                <td>
                <h:selectOneMenu id="role"  value="#{addUser.roleType}"
                style="height:28px;width:150px;">
                <f:selectItems value="#{roles.roleList}" />
                </h:selectOneMenu>
                <span class="validate"><h:message for="role"
                style="color:Red;"></h:message>  </span></td>
        </tr>

    </table>



    </div>

    <table align="center" cellpadding="20">
        <tr>
                <td><h:commandButton value="#{msg['register']}"
                styleClass="submitButton"  action ="#{addUser.addNewUser}"   /></td>
                <td><h:commandButton value="#{msg['cancel']}"
                styleClass="CancelButton" immediate="true"  action
                ="administratorOptionsView"   /></td>
        </tr>
    </table>


</h:form>


</f:view>


</ui:define>
 </ui:composition>
</html>
```

4.2.2 Codificació ManagedBean

Els ManagedBean són els objectes responsables de la lògica de l'aplicació, que responen als esdeveniments generats pels components JSF dels facelets i que controlen la navegació entre pàgines.A més gràcies als Faces Servlet que actuen de manera transparent, s'examinen les peticions rebudes, s'actualitza la representació de la interfície del client i les dades dels Managed Beans, i s'invoquen els controladors de esdeveniments i les accions, mitjançant els mètodes dels Managed Bean.

En el següent exemple de Managed Bean de la nostra aplicació (Mostrar Comanda), ens hem de fixar amb les anotacions que declaren i configuren aquest components.

- *@ManagedBean*: Amb aquesta anotació indiquem que la classe es declara com a *Managed Bean*, i amb l'atribut *"name"* li donem un nom a aquest *Managed Bean*. Aquest nom és el que s'utilitza per cridar-lo des del facelet.

- *@SessionScoped*: s'està indicant que el *ManagedBean* que s'acaba de declarar, es vol que estigui disponible per a totes les peticions que formin part de la mateixa sessió de un client.

- *@EJB*: especifica la referència a un EJB.

- @ ManagedProperty: S'utilitza per intercanviar dades amb un altre ManagedBean, d'aquesta manera ja no es necessiten variables de sessió i es fa la codificació més simple.

Anem a veure el codi:

```
package managedbean;
import java.io.Serializable;
import java.sql.Date;
import java.util.*;
import javax.ejb.EJB;
import javax.faces.bean.*;
import javax.naming.Context;
import javax.naming.InitialContext;
import jpa.DescStopCauseJPA;
import jpa.FileJPA;
import jpa.OrderJPA;
import jpa.StopCauseJPA;
import jpa.StopCauseTimeJPA;
import jpa.UserJPA;
import ejb.MaintenanceFacadeRemote;
import ejb.OperationFacadeRemote;
import ejb.QueryStadisticsFacadeRemote;

/**
 * Managed Bean ShoUserMBean
 */
@ManagedBean(name = "showOrder")
@SessionScoped
public class ShowOrder implements Serializable{

        @ManagedProperty("#{language}")
         private LanguageBean languageMbean;
        @ManagedProperty("#{login}")
        private loginMBean loginMbean;

        private static final long serialVersionUID = 1L;
```

```
@EJB
private QueryStadisticsFacadeRemote QueryStadisticsRemote;
private MaintenanceFacadeRemote  maintenanceRemote;
private OperationFacadeRemote operationRemote;

//stores OrderJPA instance
protected OrderJPA dataOrder;
//stores RegisteredUserJPA number id
protected int  idOrder = 1;
private int id;
private String code;
private FileJPA file;
private Date creationDate;
private String stringReelsNumber;
private String stringTotalMeters;
private String stringMetersQuality;
private String stringMetersNoQuality;
ArrayList<String> stopCauses = new ArrayList<String>();
ArrayList<StopCauseJPA> stopCausesJPA = new ArrayList<StopCauseJPA>();
ArrayList<String> stopCausesMinutes = new ArrayList<String>();
private String userName;
private String userSurnames;
private String fileName;
private int languageId ;
private UserJPA operator;

/**
 * Constructor
 */

public ShowOrder() throws Exception
{
}

public String getInit() throws Exception {

        //this.code = "";
        this.stringReelsNumber = "0";
        this.stringTotalMeters = "0";
        this.stringMetersQuality = "0";
        this.stringMetersNoQuality = "0";
        //We clear all array list each facelet load
        this.stopCauses.clear();
        this.stopCausesJPA.clear();
        this.stopCausesMinutes.clear();
        //We call the local method to get all stop causes descriptions
        stopCausesList();
        //We get order data
        this.showDataOrder();
        return "initialitzed";
}

/**
 *  Methods get/set the fields of database
 */
public int getIdOrder()
{
        return idOrder;
}
public void setIdFile(int idOrder) throws Exception
{
        this.idOrder = idOrder;
        showDataOrder();
}
public OrderJPA getDataOrder()
{
        return dataOrder;
}
public int getId() {
        return id;
}
public  void setId(int id) {
        this.id = id;
}
```

```
public int getLanguageId() {
	return languageId;
}
public void setLanguageId(int languageId) {
	this.languageId = languageId;
}
public LanguageBean getLanguageMbean() {
	return languageMbean;
}
public void setLanguageMbean(LanguageBean languageMbean) {
	this.languageMbean = languageMbean;
}
public loginMBean getLoginMbean() {
	return loginMbean;
}
public void setLoginMbean(loginMBean loginMbean) {
	this.loginMbean = loginMbean;
}
public void setDataOrder(OrderJPA dataOrder) {
	this.dataOrder = dataOrder;
}
public String getUserName() {
	return userName;
}
public void setUserName(String userName) {
	this.userName = userName;
}
public String getUserSurnames() {
	return userSurnames;
}
public void setUserSurnames(String userSurnames) {
	this.userSurnames = userSurnames;
}
public String getCode() {
	return code;
}
public void setCode(String code) {
	this.code = code;
}

public FileJPA getFile() {
	return file;
}
public void setFile(FileJPA file) {
	this.file = file;
}
public Date getCreationDate() {
	return creationDate;
}
public void setCreationDate(Date creationDate) {
	this.creationDate = creationDate;
}
public String getStringReelsNumber() {
	return stringReelsNumber;
}
public void setStringReelsNumber(String stringReelsNumber) {
	this.stringReelsNumber = stringReelsNumber;
}
public String getStringTotalMeters() {
	return stringTotalMeters;
}
public void setStringTotalMeters(String stringTotalMeters) {
	this.stringTotalMeters = stringTotalMeters;
}
public String getStringMetersQuality() {
	return stringMetersQuality;
}
public void setStringMetersQuality(String stringMetersQuality) {
	this.stringMetersQuality = stringMetersQuality;
}
```

```
public String getStringMetersNoQuality() {
	return stringMetersNoQuality;
}
public void setStringMetersNoQuality(String stringMetersNoQuality) {
	this.stringMetersNoQuality = stringMetersNoQuality;
}
public ArrayList<String> getStopCauses() {
	return stopCauses;
}
public void setStopCauses(ArrayList<String> stopCauses) {
	this.stopCauses = stopCauses;
}
public ArrayList<StopCauseJPA> getStopCausesJPA() {
	return stopCausesJPA;
}
public void setStopCausesJPA(ArrayList<StopCauseJPA> stopCausesJPA) {
	this.stopCausesJPA = stopCausesJPA;
}
public ArrayList<String> getStopCausesMinutes() {
	return stopCausesMinutes;
}
public void setStopCausesMinutes(ArrayList<String> stopCausesMinutes) {
	this.stopCausesMinutes = stopCausesMinutes;
}
public void setIdOrder(int idOrder) {
	this.idOrder = idOrder;
}
public String getFileName() {
	return fileName;
}
public void setFileName(String fileName) {
	this.fileName = fileName;
}

/**
 * Method to show the data of determinate user
 * @throws Exception
 */

public String showDataOrder() throws Exception
{

	Properties props = System.getProperties();
	Context ctx = new InitialContext(props);

	try{
		QueryStadisticsRemote = (QueryStadisticsFacadeRemote)
		ctx.lookup("FlexographicManagement/QueryStadisticsFacadeBean/remote");
		dataOrder = (OrderJPA) QueryStadisticsRemote.showOrder(idOrder);
		//We get the operator data who made created this file
		this.setDataUser(dataOrder.getFile().getOperator().getId());
		this.userName = this.operator.getName();
		this.userSurnames = this.operator.getSurNames();
		// We get a fileName of this order
		this.fileName = dataOrder.getFile().getName();
		this.code = dataOrder.getCode();
		this.stringReelsNumber = Integer.toString(dataOrder.getReelsNumber());
		this.stringTotalMeters = Integer.toString(dataOrder.getTotalMeters());
		this.stringMetersQuality = Integer.toString(dataOrder.getMetersQuality());
		this.stringMetersNoQuality = Integer.toString(dataOrder.getMetersNoQuality());
		//We get all stop cause times
		this.stopCauseTimeList();
	}
	catch(Exception e){
		return "errorView";
	}
	return "";
}
```

```
/**
 * Method that gets a list of instances all FIle Decks
 * @throws Exception
 */
@SuppressWarnings("unchecked")
private void stopCauseTimeList() throws Exception
{
	Properties props = System.getProperties();
	Context ctx = new InitialContext(props);


		QueryStadisticsRemote = (QueryStadisticsFacadeRemote)
		ctx.lookup("FlexographicManagement/QueryStadisticsFacadeBean/remote");

	//We fille stopCausesMinutes  array list with string fot the texts boxs

	Collection<StopCauseTimeJPA> stopCauseTimes =
	(Collection<StopCauseTimeJPA>)QueryStadisticsRemote.listStopCauseTimesbyOrder(this.idO
	rder);

	for (Iterator<StopCauseTimeJPA> iter = stopCauseTimes.iterator(); iter.hasNext();)
	{
		StopCauseTimeJPA stopCausetime = (StopCauseTimeJPA) iter.next();
		stopCausesMinutes.add(Integer.toString(stopCausetime.getMinutes()));
	}
}

/**
 * Method that takes a collection of instances of DescStopCausesJPA and StopCauseJPA calling
 * method listAllMaterialType of the EJB Session
 * @throws Exception
 */

public void stopCausesList() throws Exception
{
	languageId = this.languageMbean.getLanguageId();
	Properties props = System.getProperties();
	Context ctx = new InitialContext(props);
	operationRemote = (OperationFacadeRemote)
	ctx.lookup("FlexographicManagement/OperationFacadeBean/remote");
	@SuppressWarnings("unchecked")
	Collection<DescStopCauseJPA> descStopCauseCollection = (Collection<DescStopCauseJPA>)
	operationRemote.listDescStopCausesbyLanguage(languageId);

	for (Iterator<DescStopCauseJPA> iter2 = descStopCauseCollection.iterator();
	iter2.hasNext();)
	{
		DescStopCauseJPA DescStopCause = (DescStopCauseJPA) iter2.next();

		StopCauseJPA stopCause =
		operationRemote.getStopCause(DescStopCause.getIdStopCause().getId());
		this.stopCausesJPA.add(stopCause);
		this.stopCauses.add(DescStopCause.getDescription());
	}
}
/**
 * Method to show the data of determinate user
 * @throws Exception
 */
public void setDataUser(int userId) throws Exception
{

	Properties props = System.getProperties();
	Context ctx = new InitialContext(props);
	maintenanceRemote= (MaintenanceFacadeRemote)
	ctx.lookup("FlexographicManagement/MaintenanceFacadeBean/remote");
	this.operator = (UserJPA) maintenanceRemote.showUser(userId);

}
```

4.2.3 Codificació EJB

Aquestes classes fant de façana de cares als clients quan aquests volen tenir accès a la base de dades.Els EJB també diposen d'anotacions anotacions que declaren i configuren aquest components.

- *@Stateless*: s'està indicant que es tracta de un EJB Session sense estat.

- *@PersistenceContext*: Defineix quin context per la persistència utilitzarà en *En-tityManager*, en aquest cas es: *unitName="FlexographicManagement"*

A més un EJB va complementat per una interfície que és el component que es comunicarà amb l'exterior.Aqui mostrem l'EJB i la seva interfície del subsistema de consulta i estadística.

```
/*
 * TFG Informàtica  UOC
 * @author Marc Segura Valls, 2013
 */
package ejb;

import java.sql.Date;
import java.util.Collection;
import javax.ejb.Remote;
import jpa.OrderJPA;

/**
 * Session EJB Maintenance system Remote Interfaces
 */
@Remote
public interface QueryStadisticsFacadeRemote {

        /**
           * Remotely invoked method.
           */
          public Collection<?> listOrders(String nameFilter);
          public OrderJPA showOrder(int orderId);
          public Collection<?> listStopCauseTimesbyOrder(int orderId);
          public int totalQualityMetersByDates(Date date1 , Date date2);
          public int totalNoQualityMetersByDates(Date date1 , Date date2);
}


/*
 * TFG Informàtica  UOC
 * @author Marc Segura Valls, 2013
 */
package ejb;
import java.sql.Date;
import java.util.*;
import javax.ejb.Stateless;
import javax.persistence.*;
import jpa.OrderJPA;
import jpa.StopCauseTimeJPA;

/**
 * EJB Session Bean Class of Maintenance subsystem
 */
@Stateless
public class QueryStadisticsFacadeBean implements QueryStadisticsFacadeRemote {
        //Persistence Unit Context
        @PersistenceContext(unitName="FlexographicManagement")
        private EntityManager entman;
```

```
/**
* Method that returns Collection of files acorging filter
*/
public java.util.Collection<OrderJPA> listOrders(String codeFilter) {

	try
	{
		if (codeFilter.equals("")){
			@SuppressWarnings("unchecked")
			Collection<OrderJPA> allOrders = entman.createQuery("from OrderJPA ORDER
			BY id").getResultList();
			return allOrders;
		}
		else {
			System.out.println("EJB llistar usuaris per filtre");
			Query query = entman.createQuery("from OrderJPA o WHERE o.code LIKE
			:codeFilter ORDER BY id");
			query.setParameter("codeFilter", "%"+codeFilter+"%");
			@SuppressWarnings("unchecked")
			Collection<OrderJPA> orders = query.getResultList();
			return orders;
		}
	}catch (PersistenceException e) {
		System.out.println(e);
	}
	   return null;
	}

/**
 * Method that returns instance of the class order
 */
public OrderJPA showOrder(int orderId)throws PersistenceException {

	OrderJPA order = null;
	try
	{

		@SuppressWarnings("unchecked")

		Collection<OrderJPA> orders = entman.createQuery("FROM OrderJPA o WHERE o.id =
		?1").setParameter(1, new Integer(orderId)).getResultList();
		if (!orders.isEmpty() || orders.size()==1)
		{
			Iterator<OrderJPA> iter = orders.iterator();
			order = (OrderJPA) iter.next();
		}
	}catch (PersistenceException e) {
		System.out.println(e);
	}
    return order;
}


/**
* Method that returns Collection of all Decks by file
*/

public Collection<?> listStopCauseTimesbyOrder(int orderId)throws PersistenceException {

	Collection<StopCauseTimeJPA> allStopeCauseTimes = null;
	try
	{
		@SuppressWarnings("unchecked")
			Collection<OrderJPA> orders = entman.createQuery("FROM OrderJPA o WHERE
			o.id = :id").setParameter("id", orderId).getResultList();
			if (!orders.isEmpty() || orders.size()==1)
			{
			Iterator<OrderJPA> iter = orders.iterator();
			OrderJPA element = (OrderJPA) iter.next();
			allStopeCauseTimes = element.getStopCauseTimesbyOrder();
```

```
                }
            }catch (PersistenceException e) {
                    System.out.println(e);
            }
              return allStopeCauseTimes;
        }


        /**
         * Method that returns total quality meters betwen  dates
         */
        public int totalQualityMetersByDates(Date date1 , Date date2) {
            try
            {
                    @SuppressWarnings("unchecked")
                    Collection<OrderJPA> orders = entman.createQuery("SELECT e " +
                     "FROM OrderJPA e " +
                     "WHERE e.creationdate BETWEEN :date1 AND :date2")
                     .setParameter("date1", date1, TemporalType.DATE)
                     .setParameter("date2", date2, TemporalType.DATE)
                     .getResultList();
            int metersQuality =0 ;

                    for (Iterator<OrderJPA > iter =  orders.iterator(); iter.hasNext();)
                    {
                            OrderJPA order = (OrderJPA) iter.next();
                            metersQuality = metersQuality + order.getMetersQuality();
                    }
                    return metersQuality;

            }catch (PersistenceException e) {
                    System.out.println(e);
            }
        return 0 ;
        }

        /**
         * Method that returns total quality meters betwen  dates
         */
        public int totalNoQualityMetersByDates(Date date1 , Date date2) {

            try
            {
                    @SuppressWarnings("unchecked")
                    Collection<OrderJPA> orders = entman.createQuery("SELECT e " +
                   "FROM OrderJPA e " +
                   "WHERE e.creationdate BETWEEN :date1 AND :date2")
                     .setParameter("date1", date1, TemporalType.DATE)
                     .setParameter("date2", date2, TemporalType.DATE)
                     .getResultList();

            int metersNoQuality =0 ;

                    for (Iterator<OrderJPA > iter =  orders.iterator(); iter.hasNext();)
                    {
                            OrderJPA order = (OrderJPA) iter.next();
                            metersNoQuality = metersNoQuality + order.getMetersNoQuality();

                    }

                    return metersNoQuality;

            }catch (PersistenceException e) {
                    System.out.println(e);
            }

        return 0 ;
        }
}
```

4.2.4 Codificació entitat JPA

Seguint l'especificació Java Persistence API, s'ha d'implementar un mètode get/set per cada camp de la taula i per cada camp la relació que pugui tenir la taula, a més també s'utilitzen anotacions especials.

- @Entity: Amb ella estem indicant al contenidor que aquesta classe es de tipus JPA, i per tant serà utilitzada per accedir a les dades.

- @Table: S'està indicant el nom de la taula amb la que la classe esta rela-cionada.

- @Id: Indica quin es el camp de la taula que es la clau primària.

- @OneToMany: Ens indica que hi ha una relació One-to-Many i quins són els mètodes que la implementen. Get....By.....().

- @ManyToOne: Indica la relació Many-to-One .

- @JoinColumn: indica quin nom te el camp que implementa la relació a la taula.

Exemple del codi de la classe LanguageJPA:

```
/*
 * TFG Informàtica  UOC
 * @author Marc Segura Valls, 2013
 */

package jpa;
import java.io.Serializable;
import java.util.Collection;
import javax.persistence.*;
import org.hibernate.annotations.LazyCollection;
import org.hibernate.annotations.LazyCollectionOption;

/**
 * JPA Class UserJPA
 */

@Entity
@Table(name="FlexographicManagement.language")

public class LanguageJPA implements Serializable {
private static final long serialVersionUID = 1L;

	private int id;
	private String description;
	private Collection<DescMaterialTypeJPA> descMaterialTypes;
	private Collection<DescStopCauseJPA> descStopCauses;

	/**
	 * Class constructor methods
	 */

	public LanguageJPA() {
		super();

	}
```

```
	public LanguageJPA(int id, String description) {
		super();
		this.id = id;
		this.description = description;
	}


	/**
	 *  Methods get/set of database fields
	 *  Id Primary Key field
	 */
	@Id

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getDescription() {
		return description;
	}

	public void setDescription(String description) {
		this.description = description;
	}
	/**
	 * Methods get/set persistent relationships
	 */

	@OneToMany
	@LazyCollection(LazyCollectionOption.FALSE)

	//(cascade = CascadeType.ALL, fetch = FetchType.EAGER )
	//(fetch = FetchType.EAGER, cascade = {CascadeType.PERSIST, CascadeType.MERGE})

	@JoinColumn(name = "idLanguage")

	public Collection<DescMaterialTypeJPA> getDescMaterialTypebyLanguage() {
		return descMaterialTypes;
	}
	public void setDescMaterialTypebyLanguage(Collection<DescMaterialTypeJPA> descMaterialTypes) {
		this.descMaterialTypes = descMaterialTypes;
	}


	@OneToMany
	@LazyCollection(LazyCollectionOption.FALSE)
	@JoinColumn(name = "idLanguage")

	public Collection <DescStopCauseJPA> getDescStopCausebyLanguage() {
		return descStopCauses;
	}
	public void setDescStopCausebyLanguage(Collection<DescStopCauseJPA> descStopCauses) {
		this.descStopCauses= descStopCauses;
	}



}
```

4.2.5 Implementació gestió del idioma.

Preparar aquesta aplicació perquè sigui una plataforma multidioma ha estat un punt prioritari perquè si no es pensa des del principi, implementar-ho quan ja l'aplicació està acabada pot arribar a ser una feinada important i s'haurien de dedicar moltes hores de manteniment, cosa que es pot evitar si es pensa des d'un bon inici.

La part d'idioma consta de dues parts, la informativa de la interfície web i la informació que l'usuari guarda a la base de dades . La primera són els missatges dels facelets com etiquetes, títols , missatges, etc... i l'altre és la informació que l'usuari gradarà a la base de dades i va creixent a mesura que l'aplicació és utilitzada per varis usuaris. La manera com es guarda aquesta informació d'idioma persistent ja s'ha comentat a l'apartat de disseny de la base de dades.

Els missatges en varis idiomes són guardats en uns fitxers de tipus propierties dins la carpeta docroot/Web-Inf.

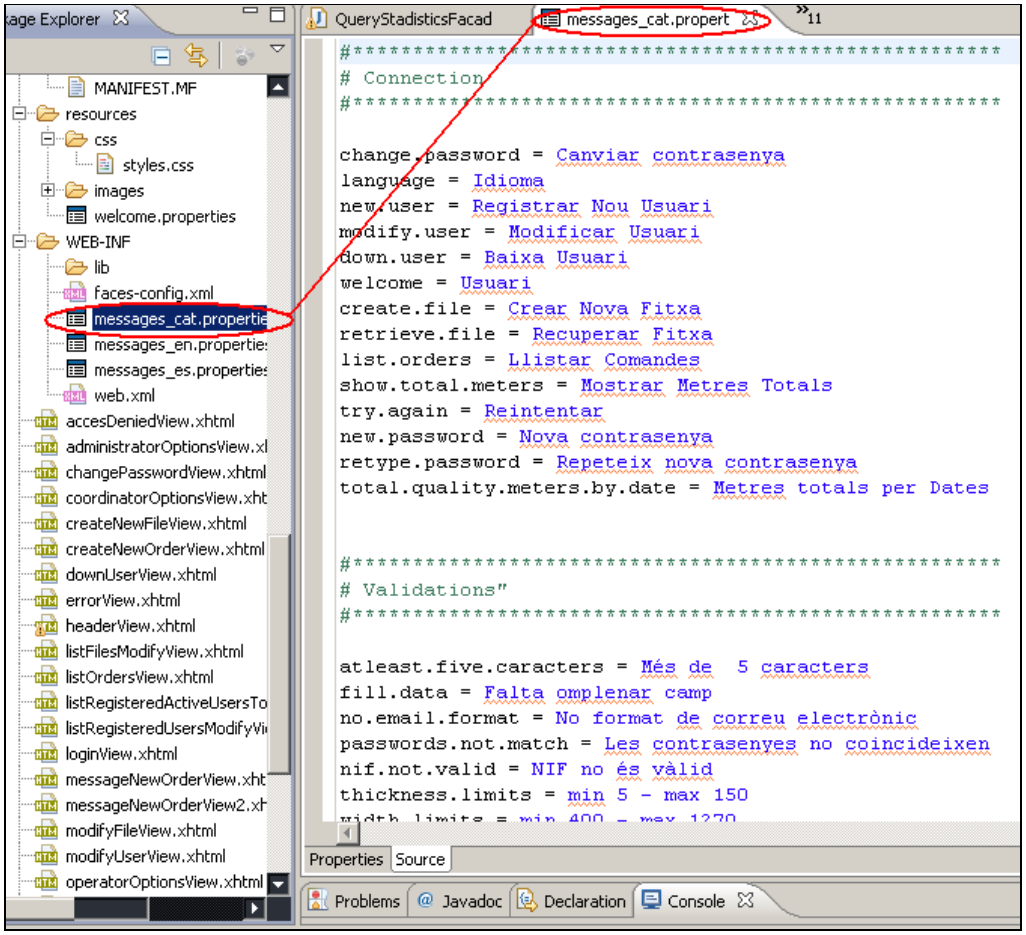

Figura 41. Missatges multidioma

Aquests missatges són cridats i gestionats des del ManagedBean LanguageMbean.Aqui és on es declaren els idiomes que es volen, i el facelet autenticar es comunica amb aquest pe quan es vol canviar de idioma.En el LanguageMbean utilitzem la classe java.util.Locale del Api de Java preparat per la internacionalització de projectes.També s'utilitza la classe import java.util.ResourceBundle per la localització dels fitxers on s'emmagatzemen els missatges.

A més configurem que els missatges a part de ser cridats des d'un facelet també puguin ser cridats des de un managedBean com per exemple el d'error o warning.

Crida d'un missatge des de la classe warningMbean:

```
this.message = LanguageBean.getMessage("all.deck.values.filled");
```

Crida d'un missatge des de un facelet:

```
<h:commandButton value="#{msg['update']}" />
```

4.2.6 Implementació validacions

El control de validacions de les dades que entra un usuari (dates, nif, email, límits, contrasenya,...) en el sistema es controlen també des de un ManagedBean, que quan és cridat des del facelet recull els arguments que l'hi donen la informació de quin element i quina informació ha de validar.A l'exemple que es mostra a continuació podem a veure com validem el correu electrònic primer des del facelet i després des del managedBean Validate.

En el facelet registerUserView.xhtml:

```
<td><h:inputText id="email" value="#{addUser.email}" validator="#{validate.validate}"size="20"
maxlength="40" styleClass="input"></h:inputText><span class="validate"> <h:message for="email"
style="color:Red;"></h:message> </span></td>
```

En el ManagedBean Validate.java:

```
@SuppressWarnings("unused")
@ManagedBean (name="validate")
@SessionScoped

public class Validates implements Serializable {


   private static final String EMAIL_PATTERN =
                  "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
                  + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";


   public void validate(FacesContext arg0, UIComponent arg1, Object arg2)
         throws ValidatorException {
```

```
//Email validate

        if (arg1.getId().equals("email")) {

                    // Compiles the given regular expression into a pattern.
                    Pattern pattern = Pattern.compile(EMAIL_PATTERN);
                    // Match the given input against this pattern
                    Matcher matcher = pattern.matcher((String)arg2);

                    if (matcher.matches()== false) {
                        throw new ValidatorException(new
                       FacesMessage(LanguageBean.getMessage("no.email.format")));

                    }

           }
```

5. Conclusions i treball futur

Després d'estar dos mesos treballant amb la plataforma JavaEE, i amb la finalitat de crear una aplicació per un sistema distribuït des de zero, crec que disposo d'una petita llicència que em permet aportar la meva opinió personal sobre aquesta disciplina.

Penso que és una plataforma molt completa, ja que m'ha permès implementar tot el que he dissenyat previament.Abans estava convençut que en el sector de les aplicacions web no hi havia lloc per el llenguatge Java perquè des de fa ja molt temps estan establertes plataformes comercials i no comercials com .NET i PHP que són molt potents i deduïa que Java dins aquest camp encara no oferia totes les prestacions que el mercat demanava.He pogut comprovar que estava equivocat i que Java EE està entrant fort i ofereix cada vegada més diversitat de frameworks molt forts i innovadors.

Jo he preferit utilitzar els frameworks més clàssics i de moment deixar de banda els més nous com Spring, struts,... així fer una bona base i ja en un pròxim projecte, fer un pas endavant.

Com a punts en contra he de manifestar que la curva d'aprenenetage ha estat molt més lenta del que esperava.Ha estat lenta perquè la informació didàctica que es pot trobar és molt dispersa per la gran diversitat d'eines i components que ofereix la plataforma.Hi han diferents IDE i servidors, i això suposa una gran quantitat de combinacions diferents a l'hora de crear un nou entorn, i totes elles amb les seves configuracions especials.Arrel d'això, he dedicat moltes hores en la instal·lar i configurar el meu entorn, tant com per la connexió entre les diferents capes com per el desplagament de l'aplicació en el servidor.Un cop tot engranat, la resta del desenvolupament ja ha estat més fluid.

Un cop contrastats els pros i els contres, aquesta la plataforma m'ha convençut i la tornaria a utilitzar en un proper projecte.

Aquest projecte tant sols és un germen del que pot arribar a ser.Per això s'ha preparat d'una manera modular i s'hi poden crear més subsistemes i també afegir més funcionalitats en els ja existents.

El subsistema de consulta i estadística és el més susceptible a ser explotat ja que actualment només ofereix una petita part del que podria arribar a oferir. S'hi poden arribar a afegir moltes funcionalitats que ajudint a controlar i millorar la producció de les màquines com per exemple consultes sobre els temps de parada i les seves causes, posar preu als materials i consultar les pèrdues dels metres de no qualitat, conèixer quins són els operadors que causes més pèrdues, etc...

En aquest control de producció també s'hi pot integrar un gestor de manteniment preventiu i predictiu de tots els compoments i actuadors (elèctrics, mecànics i pnuemàtics) de les màquines. En aquest cas s'afegiria un subsistema per el departament de manteniment, compres i producció per tal de tenir els recanvis abans de fer les intervencions, i poder coordinar la parada de màquina i els recursos per dur a terme les tasques de manteniment preventiu/predictiu.

També registrar valors actuals de paràmetres que poden repercutir a la qualitat de la impressió (temperatures, viscositats, tensions,...) en funció dels metres de cada bobina. D'aquesta manera un cop fet el control de defectes posterior, es podria relacionar el defecte de la impressió en uns determinats metres amb algun valor d'aquests paràmetres que no haguessin estat treballant correctament.

Un mòdul dedicat a l'eficiència energètica pot ser de gran ajuda per conèixer el valor econòmic de les energies utilitzades per la producció del producte (electrica, pneumàtica, gas, aigua, etc..).

Com ja s'ha comentat al llarg de la memòria, el següent pas serà el de crear una aplicació per a dispositus mòbils que mitjançant serveis web es comunicaran amb el servidor web.

Per tant el ventall de possibilitats de creixement d'aquesta aplicació és molt ampli, i a més amb el canvi d'algunes dades i entitats però mantenint la mateixa estructura, es podria implementar un control de producció per un altre tipus de maquinaria arriban ha crear una espècie de programari estàndard per el sector industrial.

6. Blibliografia

Eclipse in Action, A Guide for the Java Developer David Gallardo, Ed Burnette , Robert McGvern (2003) , Manning Publications

Java Server Faces in Action Kito D.Mann (2005) , Manning Publications

Head first EJB Kathy Sierra / Bert Bates (2003) , O'Reilly

Material didàctic asignatura Enginyeria de Programari de components i de sistemes distribuïts (2012) UOC.

Fòrums, tutorials i manuals a internet:

http://www.javaprogrammingforums.com/

http://docs.jboss.org/jbosside/tutorial/build/en/html_single/

http://www.jboss.org/jdf/examples/ticket-monster/tutorial/Introduction/

http://www.postgresql.org/docs/8.0/static/tutorial.html

http://stackoverflow.com/questions/6033905/create-the-perfect-jpa-entity

http://docs.oracle.com/javaee/5/tutorial/doc/bnbqa.html

http://www.objectdb.com/java/jpa/query