Treball fi de carrera. Àrea d'enginyeria del programari

Anàlisi i disseny d'una aplicació d'escandall amb tècniques OO

Pep Sureda Uceda
Enginyeria Tècnica en Informàtica de Gestió

Ricard Burriel Maurel
Director del projecte

01-06-2012

A tu Marc, que l' esforç per aconseguir
aquest títol sigui un exemple perquè ningú
te pugui convèncer de que no estàs capacitat
per aconseguir els teus propòsits

AGRAÏMENTS

Seria injust no mencionar a la meva família, especialment a la meva dona, agraint la comprensió, el recolzament, la paciència i les facilitats que m'han concedit durant aquests sis anys perquè pogués aconseguir els meus objectius. De ben segur que sense ells no hagués estat possible.

També vull tenir present als consultors i alumnes de la UOC que he anat coneixent en el recorregut d'aquest camí i que cadascun d'ells, a la seva manera, han contribuït en la meva arribada a la meta. Destacar al consultor Ricard Burriel per l'ajuda i atenció prestada en tot moment, per encoratjar-me i treure el millor de mi durant el desenvolupament del TFC.

RESUM

Aquest document representa la memòria tècnica corresponent a l'assignatura TFC (treball fi de carrera) dels estudis d'Enginyeria Tècnica en Informàtica de Gestió. Es tracta d'un treball que sintetitza els coneixements adquirits en les diferents assignatures de la carrera i que requereix posar-los en pràctica en un projecte concret.

El treball té un caràcter majoritàriament pràctic i vinculat a la vida professional dels enginyers superiors i tècnics en informàtica. Concretament, és centra en una àrea fonamental present en el desenvolupament d'aplicacions informàtiques, l'enginyeria del programari.

El document està estructurat de forma que la informació s'introdueix en diferents apartats de forma ordenada, descrivint les fases, els procediments i les diferents tècniques utilitzades en la producció d'una aplicació informàtica. Començarem amb una introducció per descriure el projecte, els objectius, la metodologia que s'aplicarà i el pla de treball a seguir. Seguirem amb el bloc principal del treball detallant l'anàlisi i el disseny del sistema a desenvolupar, tractant temes com la descripció dels casos d'ús, l'arquitectura del programari, el llenguatge UML, la persistència i la usabilitat entre altres.

Per últim, però no menys important, destinarem un apartat per analitzar el costos aproximats del projecte i un altre, amb les conclusions a les que s'han arribat.

ÍNDEX

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

PRIMERA FASE: Pla de treball

1 INTRODUCCIÓ

El tema escollit per a realitzar aquest TFC es diu: *Anàlisi i disseny d'una aplicació d'escandall amb tècniques OO*, corresponent a l'àrea d'*Enginyeria del Programari*. Del títol es desprenen dos aspectes fonamentals, el primer, que el producte final serà una memòria que descrigui les fases prèvies a la implementació del sistema a desenvolupar. En segon lloc, que les diferents tècniques, mètodes i eines que s'aplicaran en el projecte corresponen al paradigma de l'orientació a l'objecte.

1.1 CONTEXT DEL TFC

L'empresa *Armaçons de fusta SL* es dedica a la fabricació de l'estructura interna del sofàs. El producte final, l'armaçó o esquelet, és el resultat de la unió de diferents elements provinents principalment de taulons de fusta i materials auxiliars com coles adhesives, grapes, cintes elàstiques, etc.

La principal dificultat que té l'empresa és realitzar el càlcul del cost dels diferents models produïts (procediment conegut com *escandall*) degut a que la quantitat de material emprat en cadascun d'ells va variant, així com també ho fan les hores invertides per part dels operaris i de la maquinària utilitzada. A més, aquest còmput, al efectuar-se de forma manual resulta una operació molt laboriosa i repetitiva.

Per una altre part, al ser una organització petita només compta amb una aplicació per gestionar els clients, els proveïdors i realitzar factures, però és un producte estàndard que no acaba d'ajustar-se als requeriments de l'entitat.

1.2 JUSTIFICACIÓ DEL TFC

En l'objectiu d'informatitzar l'escandall, unificar les tasques que realitzen amb el programa estàndard i afegir les funcionalitats no contemplades, *Armaçons de fusta SL* ha decidit encarregar el desenvolupament d'un programari que s'adeqüi a les seves exigències.

La societat, compta només amb una fàbrica des d'on es duu a terme tot el procés de producció i gestió del negoci. Per tal cosa, en un principi s'havia pensat en crear una aplicació d'escriptori i centralitzada per ser manipulada només pels usuaris de l'empresa. No obstant, després de vàries reunions i de valorar les possibilitats que oferiria un accés remot a l'aplicació, com per exemple que els clients puguin fer les seves comandes des de el seu ordinador, ha originat que la part interessada es decantés cap a un entorn distribuït.

Un altre motivació per realitzar aquest TFC es que existeix la possibilitat de comercialitzar el producte final, doncs hi ha altres empreses del mateix sector que no disposen d'una aplicació d'aquestes característiques.

Una de les primeres decisions que s'han establert, és que la implantació de la nova aplicació i la gestió del canvi es faci de forma gradual, per alterar el mínim possible el cicle de vida normal de l'empresa. Així, tant el desenvolupament com la implantació del sistema global es desglossarà en varis subprojectes.

1.3 OBJECTIUS

Seguidament, es descriuen els objectius a assolir en l'apartat personal com a estudiant d'ETIG, els del TFC i els de l'aplicació a desenvolupar.

1.3.1 PERSONALS

Aplicar els diferents coneixements adquirits en els estudis d'Enginyeria en Informàtica de Gestió (ETIG), com són l'enginyeria del programari, el disseny de bases de dades, el desenvolupament de programari, la programació orientada a l'objecte, la gestió de projectes,

les competències comunicatives, la producció i presentació de documents cientificotècnics. En resum, demostrar que com a alumne s'han assolit correctament els conceptes apresos al llarg de la carrera.

1.3.2 DEL TFC

Elaborar una memòria concisa i clara que detalli les fases d'anàlisi i disseny del cicle de vida d'un programari. Aquest document permetrà que a curt i mitjà termini es pugui realitzar la implementació d'una part de les funcionalitats de l'aplicació global. La codificació, al igual que les proves, la integració i el manteniment, són fases que queden fora de l'objectiu d'aquest TFC i que per tant, seran tractats en un altre projecte.

Els objectius específics del TFC són:

- Realitzar la recollida i documentació dels requisits.
- Aplicar una metodologia per desenvolupar programari.
- Confeccionar les etapes d'anàlisi i disseny del cicle de vida del programari.
- Conèixer i aplicar les notacions del model UML.
- Consolidar les tècniques de disseny de bases de dades relacionals.
- Dissenyar les interfícies gràfiques dels casos d'ús principals.
- Tractar i aplicar el concepte d'usabilitat.
- Conèixer les característiques de la comunicació escrita i aplicar-les als textos relacionats amb les TIC.

1.3.3 DE L'APLICACIÓ

Tot i que la meta final es obtenir un programari que compleixi amb tots els requisits que el client necessita, com s'ha comentat anteriorment els objectius s'aniran assolint de forma gradual en diferents etapes. Així doncs, aquest TFC és el primer dels projectes que es tenen prevists per cobrir l'anàlisi i el disseny dels subsistemes *Manteniment, Connexió* i *Flux de treball*.

Les funcionalitats establertes per aquesta primera fase són:

- Alta, baixa i consulta/modificació de models.
- Alta (escandall), baixa i consulta/modificació d'un article.
- Alta, baixa i consulta/modificació de tipus de material (unitat, m2, m3).
- Alta, baixa i consulta/modificació de material.
- Alta, baixa i consulta/modificació d'usuaris.
- Alta, baixa i consulta/modificació de proveïdors.
- Gestió de dades generals.
- Gestió de les comandes.
- Gestió de les compres.
- Gestió de l'estoc del material.
- Accés al sistema i gestió dels menús.

1.4 METODOLIGIA DEL PROJECTE

L'enginyeria del programari és una tecnologia multicapa[1] i al igual que altres enginyeries té que recolzar-se sobre un compromís d'organització de qualitat. En la figura 1 es pot veure una representació de l'estructura de capes.

Aquest apartat es centra en la capa corresponent als mètodes -els quals indiquen com construir tècnicament el programari- i en explicar les característiques de la metodologia que serà utilitzada en aquest TFC, tot justificant els motius de la seva elecció.

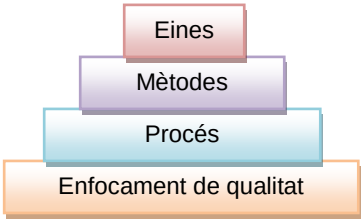


Figura 1. Capes de l'enginyeria del programari

[1] PRESSMAN, ROGER S.: Ingeniería del software: Un enfoque práctico. McGraw-Hill (2002)

1.4.1 PERQUÈ UTILITZAR UNA METODOLOGIA

El procés de creació de programari és una tasca prou complicada que requereix una sèrie de passes predicibles que serveixin com a guia en el seu desenvolupament. Aquestes passes es veuen agrupades en lo que es coneix com a metodologia i se'n poden trobar diferents tipus. Un dels objectius principals és proporcionar estabilitat, control i organització al procés de creació de programari que pot, si no és controla, tornar-se caòtic o resultar un fracàs. De forma general, els models existents estan formats per unes etapes fonamentals que són:

- Especificació i anàlisi de requisits
- Disseny
- Implementació
- Proves unitàries i d'integració
- Instal·lació
- Manteniment

1.4.2 EL MÈTODE ITERATIU INCREMENTAL

La metodologia escollida que s'aplicarà al llarg del TFC correspon al model iteratiu incremental. És un procés del tipus evolutiu que es fonamenta en la repetició de varis cicles d'una altre metodologia existent, el model en cascada realimentat[2] representat en la figura 2.

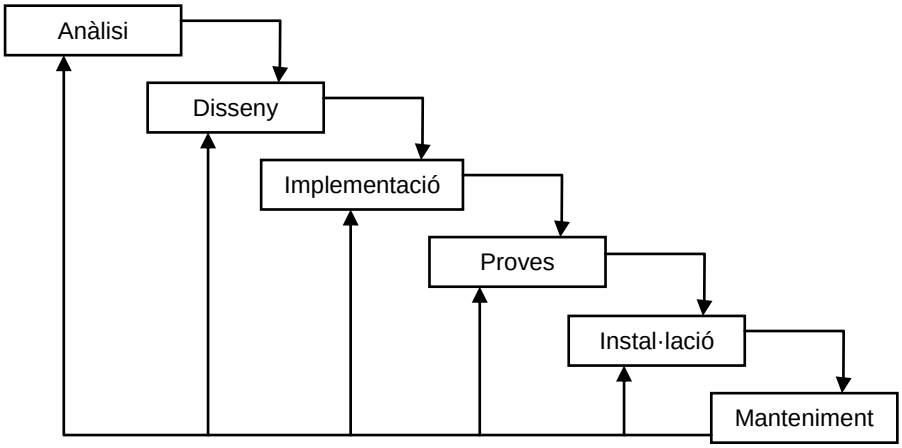


Figura 2. Model en cascada realimentat

[2] Gràfic extret de la web: http://es.wikipedia.org/wiki/Software#Modelo_cascada

El model de cascada realimentat es diferencia del model en cascada clàssic o seqüencial, perquè permet passar a una etapa anterior si és necessari, com per exemple, anar de la fase d'implementació a l'anterior per redissenyar alguna funcionalitat que no és correcte. No obstant, té una sèrie de limitacions que amb el procés iteratiu incremental es veuen minimitzats.

Dels diferents models que es poden trobar el mètode iteratiu incremental[3] és el que millor s'adequa per a aquest projecte. Els motius de la seva elecció són els següents:

- Permet lliurar un producte parcial però completament operacional en cada increment, on el procés es repeteix fins a completar el programari final complet. Aquest fet s'ajusta al desig del client de tenir una part de l'aplicació operativa cada cert temps.
- Després de cada fase d'integració es lliura un producte amb major funcionalitat que l'anterior.
- Admet l'existència d'activitats de desenvolupament de forma paral·lela o concurrent, permetent que mentre es fa el disseny o codificació en un increment ja es pugui iniciar la fase d'anàlisi del següent.
- Permet prioritzar els increments segons la urgència de les funcionalitats.
- L'enfocament incremental es adequat per quan es disposen pocs recursos humans per desenvolupar, com és el cas.
- No obliga a especificar amb precisió i de forma detallada tots els requisits del sistema al inici del projecte. Com és normal, el client en aquest cas no té clar totes les funcions en la fase inicial del projecte.
- Facilita la incorporació de nous requisits durant el desenvolupament, situació molt comú en que el client vol realitzar canvis o presentar noves funcionalitats.
- En poc temps el client obté un feedback que pot resultar útil per les pròximes iteracions en el desenvolupament de noves funcionalitats.

[3] PRESSMAN, ROGER S.: Ingeniería del software: Un enfoque práctico. McGraw-Hill (2002)

Una representació gràfica del model iteratiu incremental podria ser el que es mostra en la figura 3. Hi ha que tenir en compte que encara que el gràfic pareixi un model en cascada seqüencial o clàssic, s'ha fet per simplificar la il·lustració i realment correspon al de cascada realimentat vist anteriorment. També es pot apreciar com es pot passar a la fase d'anàlisi d'una nova iteració estant en etapes diferents.

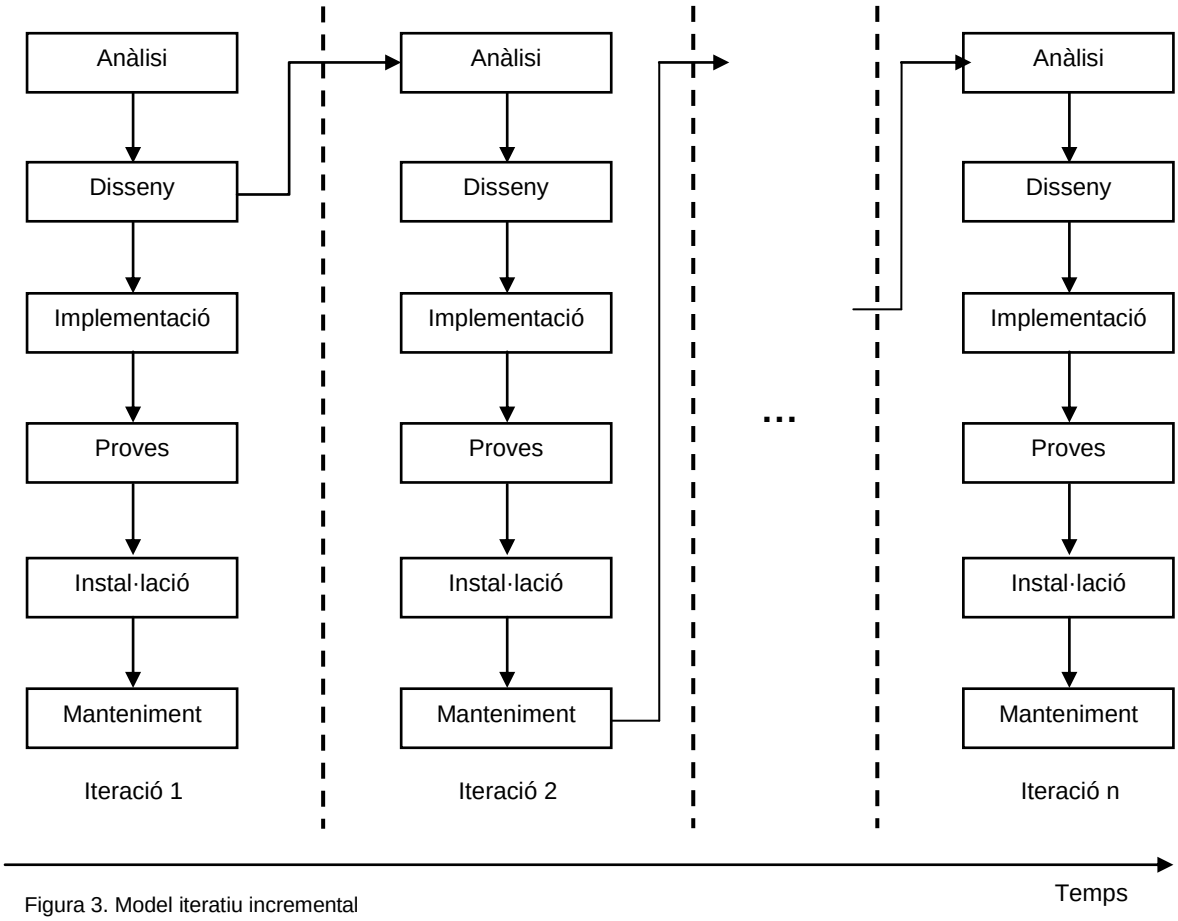


Figura 3. Model iteratiu incremental

1.4.3 COMBINACIÓ AMB EL MODEL UML

En aquest projecte, el mètode iteratiu incremental serà combinat amb el Llenguatge Unificat de Modelat (UML) que permetrà representar d'una forma visual i estàndard les fases d'anàlisi i disseny del cicle de vida del programari. Com s'ha comentat al principi, l'enfocament del TFC és orientat a l'objecte i la notació UML és un dels mètodes més emprats per realitzar l'anàlisi i disseny OO.

1.5 PLANIFICACIÓ

La planificació s'ha dividit en quatre fases que coincideixen amb els lliuraments de tres PAC, més la memòria i la presentació a realitzar durant el semestre. A partir de la descripció de les fases que componen el TFC s'ha confeccionat el diagrama de Gantt on es pot veure la planificació de cada tasca en el temps. Cal destacar que al ser aquesta la fase d'inici, el pla de treball es pot veure alterat en etapes successives degut a l'aparició de noves tasques i funcionalitats o bé, eliminar-ne alguna. La dedicació que es pot destinar al TFC és d'unes 3-4 hores diàries de dilluns a divendres. No obstant, pot haver-hi moments puntuals que requereixin una major dedicació, sent un motiu més que pot variar el pla de treball.

1.5.1 FASES DE LA PLANIFICACIÓ

Primera fase: Pla de treball. Correspon al lliurament de la PAC 1, on es durà a terme una reunió amb el client, fer la descripció del TFC, establir els objectius, estudiar les diferents metodologies del cicle de vida i elegir la més adequada, dissenyar la planificació de tasques.

Segona fase: Anàlisi de requeriments. Es presentarà en la PAC 2, on es realitzarà una altre reunió amb el client, es definiran els subsistemes, s'identificaran els actors, es confeccionarà el diagrama de casos d'ús i la descripció dels casos d'ús.

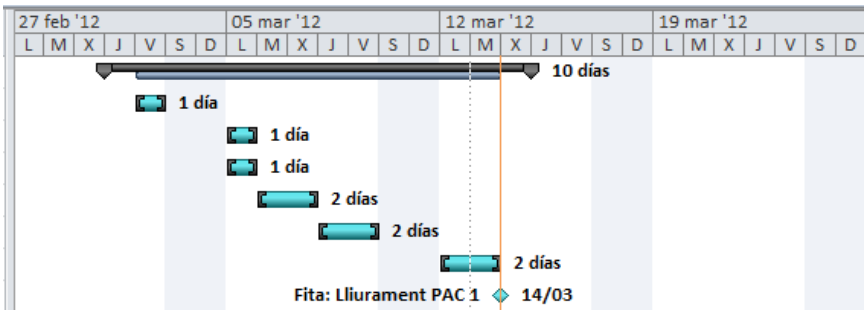
Tercera fase: Disseny. Correspon a la PAC 3 i es detallarà l'arquitectura de l'aplicació, el diagrama de classes, el diagrama de seqüència i el disseny d'interfícies d'usuari. En aquest punt es farà una reunió amb el client per entre altres coses mostrar els resultats de les GUI. Es seguirà amb el diagrama ER i la descripció dels atributs de les entitats per la persistència.

Quarta fase: Memòria i presentació. Correspon a l'últim lliurament on es realitzarà la revisió i finalització de la memòria, així com l'elaboració de la presentació.

1.5.2 DIAGRAMA DE GANTT

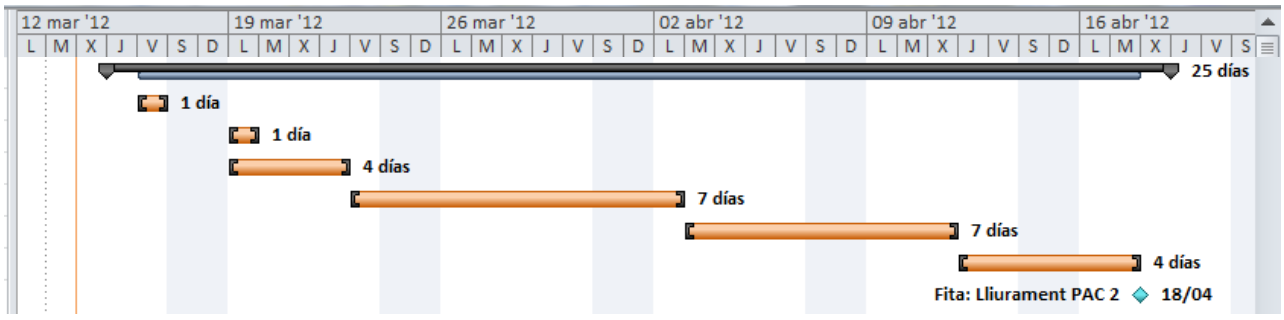
- Detall de les tasques i planificació setmanal de la fase 1:

	Nombre de tarea	Comienzo	Fin
1	**Primera fase: Pla de treball**	**jue 01/03/12**	**mié 14/03/12**
2	Reunió amb el client	vie 02/03/12	vie 02/03/12
3	Resum del TFC	lun 05/03/12	lun 05/03/12
4	Establir objectius	lun 05/03/12	lun 05/03/12
5	Estudi i elecció metodologia	mar 06/03/12	mié 07/03/12
6	Planificació de tasques	jue 08/03/12	vie 09/03/12
7	Elaboració document a lliurar	lun 12/03/12	mar 13/03/12
8	Fita: Lliurament PAC 1	mié 14/03/12	mié 14/03/12



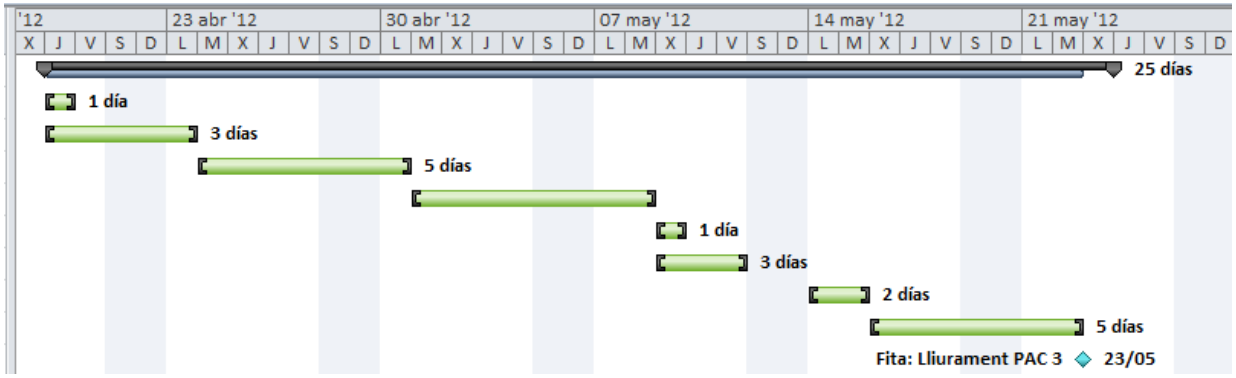
- Detall de les tasques i planificació setmanal de la fase 2:

	Nombre de tarea	Comienzo	Fin
9	**Segona fase: Anàlisi**	**jue 15/03/12**	**mié 18/04/12**
10	Reunió amb el client	vie 16/03/12	vie 16/03/12
11	Identificar actors del sistema	lun 19/03/12	lun 19/03/12
12	Definir subsistemes, revisar funcionalitats	lun 19/03/12	jue 22/03/12
13	Diagrama de casos d´ús	vie 23/03/12	lun 02/04/12
14	Descripció dels casos d'ús	mar 03/04/12	mié 11/04/12
15	Elaboració document a lliurar	jue 12/04/12	mar 17/04/12
16	Fita: Lliurament PAC 2	mié 18/04/12	mié 18/04/12



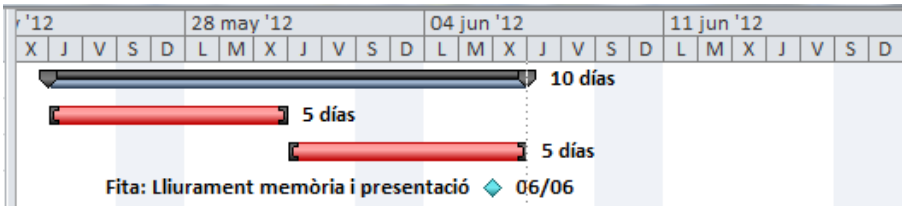
- Detall de les tasques i planificació setmanal de la fase 3:

	Nombre de tarea	Comienzo	Fin
17	**Tercera fase: Disseny**	**jue 19/04/12**	**mié 23/05/12**
18	Arquitectura de l'aplicació	jue 19/04/12	jue 19/04/12
19	Diagrama de classes	jue 19/04/12	lun 23/04/12
20	Diagrama de seqüència	mar 24/04/12	lun 30/04/12
21	Disseny Interfícies d'usuari	mar 01/05/12	mar 08/05/12
22	Reunió amb el client	mié 09/05/12	mié 09/05/12
23	Diagrama ER (persistència)	mié 09/05/12	vie 11/05/12
24	Descripció atributs (persistència)	lun 14/05/12	mar 15/05/12
25	Elaboració document a lliurar	mié 16/05/12	mar 22/05/12
26	Fita: Lliurament PAC 3	mié 23/05/12	mié 23/05/12



- Detall de les tasques i planificació setmanal de la fase 4:

	Nombre de tarea	Comienzo	Fin
27	**Quarta fase: Memòria i presentació**	**jue 24/05/12**	**mié 06/06/12**
28	Revisió i finalització memòria	jue 24/05/12	mié 30/05/12
29	Elaboració presentació	jue 31/05/12	mié 06/06/12
30	Fita: Lliurament memòria i presentació	mié 06/06/12	mié 06/06/12



SEGONA FASE: Anàlisi

2 RECOLLIDA DE REQUISITS I ANÀLISI

S'inicia aquesta segona fase del TFC amb el procés d'anàlisi del programari a desenvolupar, anomenat a partir d'ara *Wooden Framework Managament*, que simplificarem amb les sigles WFM per a facilitar la seva referència. La primera tasca realitzada ha sigut mantenir una segona reunió amb el client per a verificar, modificar i/o ampliar els requisits que es varen detectar en la primera sessió.

2.1 DESCRIPCIÓ TEXTUAL

Armaçons de fusta SL fabrica els esquelets dels sofàs que posteriorment són venuts a altres empreses (clients) que realitzen el procés final de revestiment o tapiçat. Cada esquelet o armaçó és un article que pertany a una família o model, es a dir, que un model és una col·lecció d'un o varis articles. Els articles estan compostos de diferents materials (barres de fusta i altres elements auxiliars). Els materials es poden mesurar de varies formes, però les més comuns són unitats, m^2 i m^3.

Armaçons de fusta SL realitza compres als proveïdors, els quals li serveixen el material necessari perquè pugui produir els seus articles. El material es guarda en un magatzem que hi ha dins la fàbrica. Per un altre costat estan els clients que realitzen comandes d'un o varis articles. Al dur a terme una compra o una venda (comanda) l'estoc del material es regularitza (augmenta o disminueix) per tal de tenir un coneixement exacte de les existències. No cal controlar l'estoc dels articles degut a que només es produeixen les unitats demandades pel client.

L'empresa ha decidit que hi hagi un administrador que efectuï les tasques de manteniment dels usuaris, dels rols, els magatzems i dels paràmetres generals de l'aplicació. L'auxiliar administratiu s'encarregarà de la gestió dels models, els materials i els seus tipus, els articles, els proveïdors i el flux del negoci (compres, comandes, facturació). Finalment, els clients també podran realitzar comandes de forma remota. En definitiva, cada usuari tindrà uns permisos que delimitaran les parts del programari a les quals poden accedir una vegada identificats en el sistema.

2.2 REQUISITS FUNCIONALS

Les converses mantingudes amb el client han servit per detectar els requisits funcionals que descriuen el comportament del sistema. A partir d'aquesta informació s'ha assignat un codi a cadascun dels requeriments, facilitant així la seva referència en altres fases com ara el disseny o la implementació.

Taula 1. Requisits funcionals

ID	DESCRIPCIÓ
RF_01	*Identificació. Es necessari que l'usuari realitzi un procés d'autentificació per accedir al sistema*
RF_02	*Gestió dels usuaris del sistema. L'administrador serà l'encarregat de l'alta, baixa, i consulta (inclou modificació) dels usuaris.*
RF_03	*Gestió dels rols dels usuaris. Tasca realitzada per l'administrador que consisteix en donar d'alta, baixa o consulta (inclou modificació) dels rols dels usuaris.*
RF_04	*Permisos i restriccions. Segons el rol de l'usuari identificat, el sistema li permetrà accedir a unes o altres opcions de l'aplicació.*
RF_05	*Gestió dels articles. L'auxiliar administratiu podrà realitzar l'alta (escandall), baixa i consulta (inclou modificació) dels articles.*
RF_06	*Gestió dels materials. L'auxiliar administratiu podrà realitzar alta, baixa i consulta (inclou modificació) dels materials.*
RF_07	*Gestió de tipus de material. Podrà ser de tres tipus, unitari, m^2 i m^3. L'auxiliar administratiu realitzarà l'alta, baixa i consulta (inclou modificació)*
RF_08	*Gestió dels models. Tasca de l'auxiliar administratiu que consisteix en l'alta, baixa i consulta (inclou modificació) dels models.*
RF_09	*Gestió de proveïdors. L'auxiliar administratiu podrà donar d'alta, baixa i consulta (inclou modificació) dels proveïdors.*
RF_10	*Gestió de les compres realitzades als proveïdors. L'auxiliar administratiu realitzarà l'alta, consulta (inclou modificació) i devolucions de compres.*
RF_11	*Gestió de les comandes. Inclou alta, cancel·lació i consulta (inclou modificació) que principalment realitzaran els clients, però també ho podrà fer l'auxiliar administratiu.*

RF_12	*Gestió dels magatzems. L'administrador podrà donar d'alta, baixa i consultar (inclou modificació) un magatzem.*
RF_13	*Gestió de l'estoc. El sistema regularitzarà l'estoc del material a mesura que es vagin realitzant compres i comandes. L'administrador podrà consultar l'estoc i modificar-ho si hi ha un desquadrament.*
RF_14	*Gestió dels paràmetres generals. Són unes constants que l'aplicació utilitzarà com ara el preu d'hora del treballador, percentatges de despeses, etc. L'administrador serà l'encarregat d'introduir-les una vegada instal·lat el programari i també de consultar-les (inclou modificació).*
RF_15	*Canvi d'estat de les entitats. El sistema realitzarà un canvi d'estat a les entitats que tenen l'operació de baixa. Realment les dades no seran eliminades sinó que s'aplicarà un canvi al seu estat, que pot ser ACTIVE o INACTIVE, per tant, seguiran estant de forma persistent a la base de dades.*
RF_16	*Canvi d'estat de les comandes. Les comandes podran tenir diferents estats segons sigui la seva situació, podent ser PENDING, IN PROCESS, FINISHED o CANCELED.*
RF-17	*Canvi de contrasenya. Els usuaris del sistema han de poder modificar la seva contrasenya per motius de seguretat*.

2.3 REQUISITS TÈCNICS I EQUIP DESENVOLUPADOR

Per dur a terme l'anàlisi i el disseny del projecte és necessari reunir una sèrie de requisits tècnics a nivell de programari, maquinari i evidentment, l'equip humà que organitzi i efectuï les tasques especificades en el pla de treball.

2.3.1 MAQUINARI I PROGRAMARI

- L'equip desenvolupador compta amb un ordinador de sobretaula, dos portàtils, una impressora, fax, escàner i unitats de disc externes per realitza un *backup* per assegurar tota la informació del projecte.

- Connexió dels equips mitjançant una xarxa tipus LAN.

- Connexió a Internet de banda ampla ADSL.

- Aplicació ofimàtica per la redacció i confecció de la documentació del projecte (memòria, pla de treball, informes, diagrames, manuals, etc.)

- Eines CASE per la realització dels diagrames UML. En aquest cas s'ha optat per l'aplicació MagicDraw UML Personal Edition.

- L'IDE Netbeans 7.0.1 per el disseny de les interfícies gràfiques i la implementació del programari, tot i que aquesta fase no està inclosa en el projecte.

- Sistema gestor de bases de dades PostgreSQL 9.0 per l'anàlisi, disseny i gestió de la persistència.

- Aplicació Skype per la comunicació síncrona entre els membres de l'equip i poder mantenir reunions virtuals de forma periòdica.

- Correu electrònic i fòrum del campus virtual de la UOC per la comunicació asíncrona entre els membres.

2.3.2 RECURSOS HUMANS

L'equip que desenvoluparà les fases d'anàlisi i disseny està compost de les següents persones:

- **Ricard Burriel Maurel**: *Project manager* (cap del projecte)
- **Pep Sureda Uceda**: *Programmer analyst* (anàlisi, disseny, implementació)

En posteriors projectes, on es realitzaran altres tasques del cicle de vida del programari, com per exemple la implementació o el manteniment, es revisarà la necessitat d'ampliar el personal i si s'han de reestructurar les competències de cada membre.

2.4 MODEL DE NEGOCI

El model de negoci permet descriure els processos i actors que intervenen en un sistema amb l'objectiu de comprendre el context i l'activitat del programari. El modelat del negoci es pot expressar mitjançant un diagrama de casos d'ús, que està compost d'una sèrie d'actors, les interaccions o casos d'ús i les relacions.

2.4.1 IDENTIFICACIÓ DELS ACTORS

En UML, un actor és un paper que interpreta qualsevol entitat externa que interactua amb el programari, enviant-li o rebent informació. Al parlar d'una entitat es pot referir a una persona, una màquina, un procés o una altra aplicació. A continuació es detallen els actors que interaccionen amb el sistema a desenvolupar:

- **User**. Representa qualsevol usuari que vol accedir al sistema i per fer-ho, es necessari que prèviament s'hagi identificat.

- **Actor Customer (client)**. Hereta d'*User* i la principal funció que pot exercir és gestionar les comandes (crear, consultar i cancel·lar)

- **Actor Administrative assistant (auxiliar administratiu)**. Hereta de *Customer*, per tant, a més de les operacions del client pot realitzar la gestió dels models, dels articles, els proveïdors, les compres, els materials i els seus tipus.

- **Actor Administrator (administrador)**. Hereta d'*User*, una vegada autenticat podrà gestionar els usuaris, els rols, els magatzems, l'estoc i les dades generals (paràmetres específics que l'empresa utilitza per fer alguns càlculs)

2.4.2 DIAGRAMA DE CASOS D'ÚS

El model de casos d'ús descriu les formes en les que es pot emprar un sistema. Un cas d'ús[4] és un procés autònom iniciat per una actor o per un altre cas d'ús i constitueix un flux complet d'esdeveniment, que especifica la interacció que té lloc entre un actor i el sistema.

En la figura 4 es pot observar el diagrama de casos d'ús general corresponent al model de negoci del TFC. Per no augmentar la seva complexitat i enteniment s'han agrupat en un mateix cas d'ús aquells processos que pertanyen a una entitat comú. Per exemple, l'alta, baixa, consulta i cerca d'un article formen el conjunt *Article management* (gestió d'articles). També s'han inclòs les relacions *extend* que existeixen. Aquest tipus de relació indica que un cas d'ús A amplia les funcionalitats del B, quan en el B es compleix una determinada condició. El cas d'ús que estén pot ser executat directament per un actor primari.

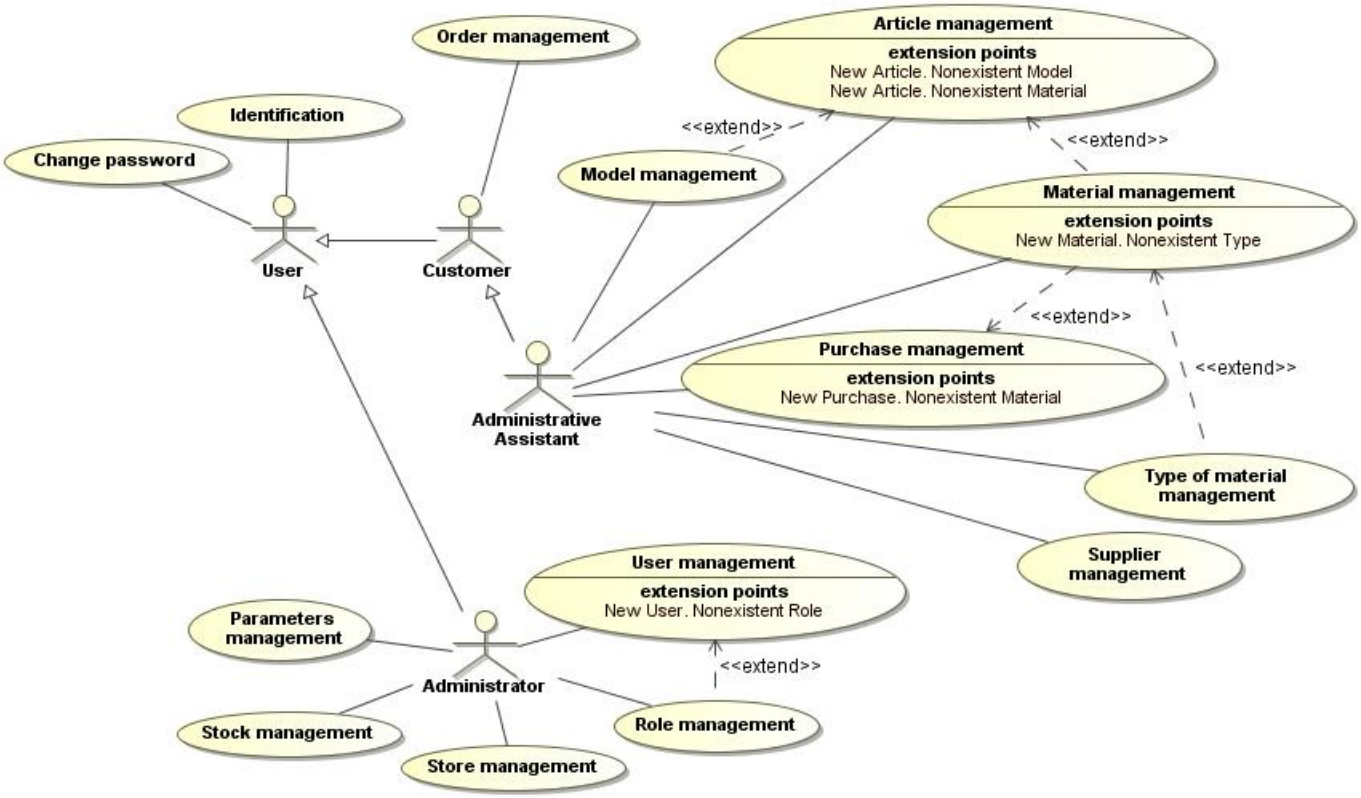


Figura 4. Diagrama general de casos d'ús

[4] CAMPDERRICH FALQUERAS, BENET. *Enginyeria del programari*. Barcelona: UOC (2004)

A continuació, en les figures 5, 6 i 7 s'il·lustren de forma desglossada totes les interaccions que pot realitzar cada actor per tenir una visió més completa del comportament. En aquest cas s'indiquen les relacions *include*, que és quan els casos d'ús A, B, C... inclouen a un altre, quan aquest últim és una part comuna d'A, B, C... El cas d'ús inclòs no té actor primari, per tant, no pot ser executat directament per l'actor.

- Actor: Customer (client)

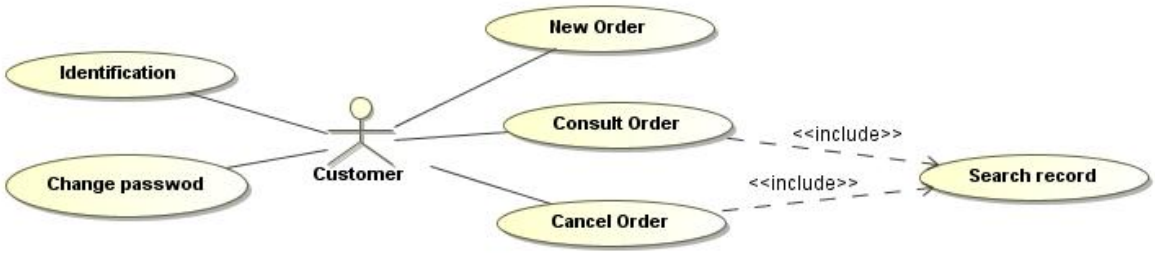


Figura 5. Diagrama de casos d'ús actor Customer

- Actor: Administrative assistant (auxiliar administratiu)

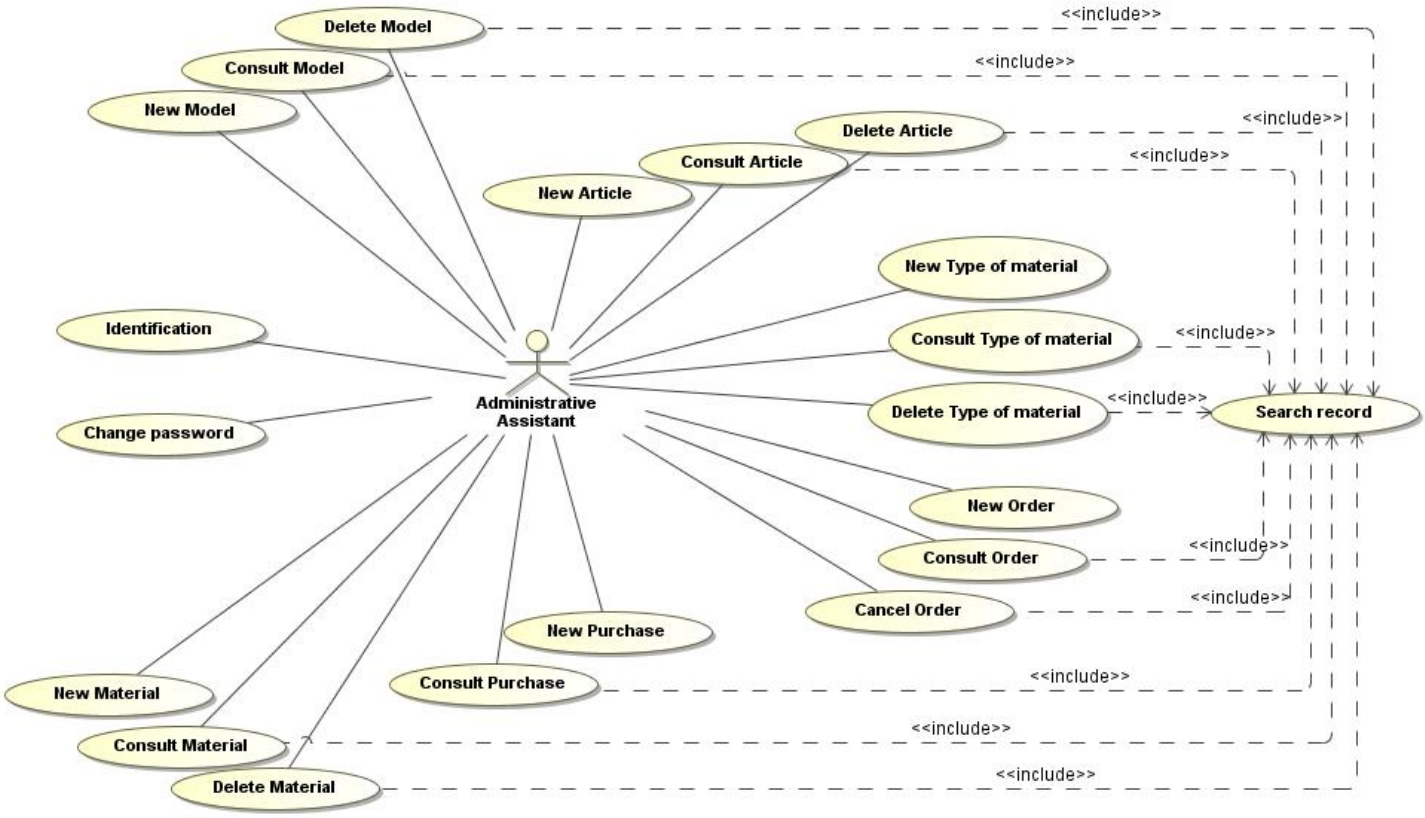


Figura 6. Diagrama de casos d'ús actor Administrative assistant

- Actor: Administrator (administrador)

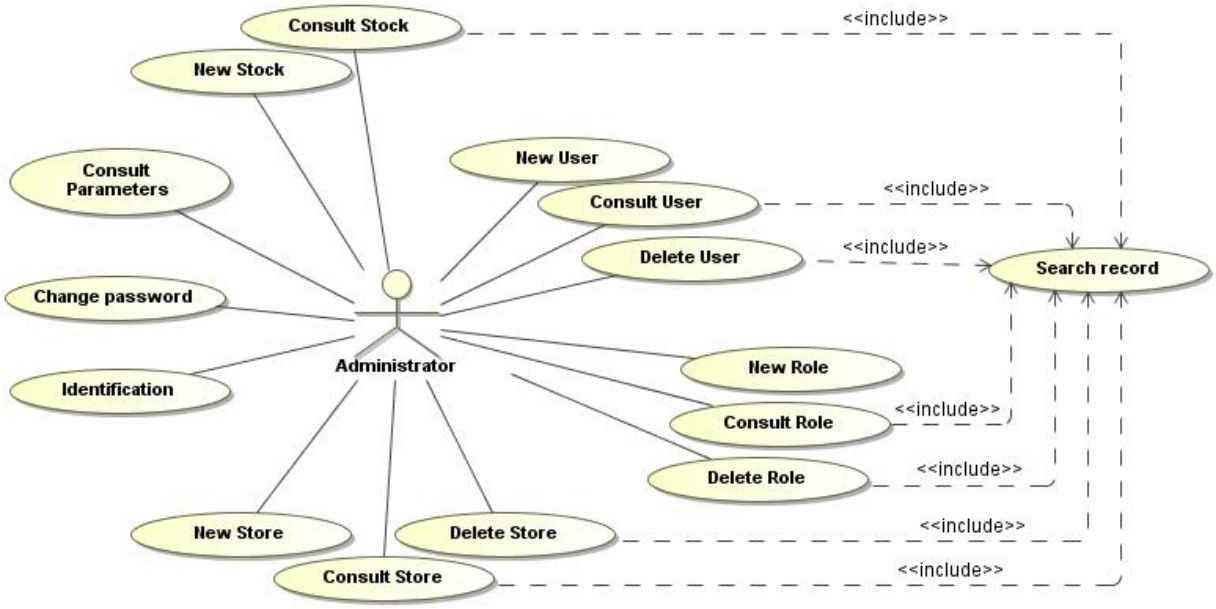


Figura 7. Diagrama de casos d'ús actor Administrator

2.4.3 DESCRIPCIÓ DELS CASOS D'ÚS

Juntament amb la representació gràfica dels casos d'ús detectats, es recomana incloure una descripció textual de cadascun d'ells, enumerant i especificant mitjançant unes taules, els detalls que ajuden a la seva comprensió, incloent entre altres coses la funcionalitat, la precondició, la postcondició i el procés principal a realitzar.

El nombre de casos d'ús per aquest TFC és de 35. Al ser aquest un document limitat en la seva extensió, s'exposen alguns dels exemples més representatius perquè el lector pugui veure com s'ha tractat aquest apartat. La resta de casos d'ús es poden consultar en el document d'annexos que acompanya la memòria, concretament en l'annex 1.

En les següents taules s'ha inclòs la descripció d'una alta, baixa i consulta/modificació d'un objecte, perquè pot servir de mostra per altres casos que utilitzen aquestes funcionalitats.

Taula 2. Cas d'ús identificació

CU-01: Identification	
Resum funcionalitat	El sistema valida l'usuari mitjançant un login i password per tenir accés a l'aplicació.
Actors	*User, Administrator, Administrative Assistant, Customer.*
Casos d'ús relacionats	
Precondició	L'usuari està donat d'alta al sistema i no es està autenticat.
Postcondició	L'usuari s'ha identificat correctament i ha accedit al sistema.
Procés principal	1-Es presenta la pantalla d'identificació. 2-L'usuari introdueix el login i el password. 3-L'usuari prem *Accept*. 4-El sistema verifica les dades de l'usuari. 5-L'usuari s'ha autenticat.
Alternatives de procés i excepcions	1-L'ususari prem *Exit* i es tanca l'aplicació. 2.1-El login i/o el password no són correctes. 2.2-El sistema informa a l'usuari. 2.3-L'usuari pot identificar-se de nou o sortir.

Taula 3. Cas d'ús nou rol

CU-02: New role (Role management)	
Resum funcionalitat	Permet donar d'alta un nou rol.
Actors	*Administrator*.
Casos d'ús relacionats	Estén New user
Precondició	El rol no existeix a la base de dades.
Postcondició	El rol s'ha emmagatzemat a la base de dades.
Processos principals	1-Apareix la pantalla d'alta d'un rol. 2-L'usuari emplena els camps. 3-L'usuari indica els permisos i restriccions del rol. 4-L'usuari prem *Save*. 5-El sistema guarda les dades en la base de dades.

Alternatives de procés i excepcions	1.1-L'usuari no ha introduït cap dada i prem *Exit*. 1.2-Es torna a la pantalla principal. 2.1-Després d'introduir algunes dades l'usuari prem *Exit*. 2.2-El sistema avisa que les dades es perdran. Sol·licita confirmació (*Yes*/*No*). 2.3-Si contesta *Yes*, es descarten les dades i tanca l'opció d'alta. 2.4-Si escull *No* es torna a la pantalla d'alta. 3.1-Alguna dada no és vàlida o té el format incorrecte. 3.2-El sistema informa a l'usuari de l'error. 4.1-Falta emplenar algun camp obligatori. 4.2-El sistema informa a l'usuari dels camps que falten.

Taula 4. Cas d'ús consultar rol

CU-03: Consult role (Role management)	
Resum funcionalitat	Permet consultar/modificar un rol existent així com activar-ho si es troba en estat *Inactive*.
Actors	*Administrator*.
Casos d'ús relacionats	Inclou Search record
Precondició	El rol existeix a la base de dades.
Postcondició	El rol s'ha consultat/modificat.
Processos principals	1-Es mostra la pantalla de consultar/modificar un rol. 2-L'usuari fa una cerca del rol que vol consultar/modificar. 3-L'usuari selecciona el registre del llistat. 4-L'usuari consulta/modifica les dades. 5-L'usuari prem *Save* si ha realitzat modificacions.
Alternatives de procés i excepcions	1.1-L'usuari prem *Exit*. 1.2-Es torna a la pantalla principal. 2.1-El sistema informa que el rol no existeix a la base de dades. 2.2-L'usuari prem *Exit*. 2.3-Es torna a la pantalla principal. 3.1-Alguna dada no és vàlida o té el format incorrecte. 3.2-El sistema informa a l'usuari de l'error.

Taula 5. Cas d'ús cercar registre

CU-34: Search record	
Resum funcionalitat	Permet realitzar la cerca d'un registre dins una base de dades.
Actors	*Administrator, Administrative assistant, Client*.
Casos d'ús relacionats	Consult role, Delete role, Consult user, Delete user, Consult store, Delete store, Consult model, Delete model, Consult type of material, Delete type of material, Consult material, Delete material, Consult article, Delete article, Consult supplier, Delete supplier, Consult purchase, Consult order, Delete order, Consult stock.
Precondició	El registre a buscar existeix a la base de dades.
Postcondició	El registre a buscar s'ha trobat.
Processos principals	1-Es mostra la pantalla de cerca. 2-L'usuari elegeix si vol cercar entitats *Active* o *Inactive*. 3-L'usuari introdueix l'ID o descripció del registre a cercar. 4-El sistema mostra el registre indicat.
Alternatives de procés i excepcions	1.1-L'usuari prem *Exit*. 1.2-Es torna a la pantalla principal. 2.1-El registre a cercar no es troba a la base de dades. 2.2-L'usuari prem *Exit*. 2.3-Es torna a la pantalla principal. 3.1-Alguna dada no és vàlida o té el format incorrecte. 3.2-S'informa a l'usuari de l'error.

2.5 ANÀLISI DE RISCOS I PLA DE CONTINGÈNCIA

L'anàlisi i la gestió del risc[5] dins el context de l'enginyeria del programari, són una sèrie de passes que ajuden a l'equip implicat en desenvolupar un projecte a entendre i gestionar la incertesa. Un risc és un problema potencial que pot succeir o no, però sense cap dubte el millor es tenir-ho identificat, estimar el seu impacte sobre el projecte i establir un pla de contingència per si ocorre.

2.5.1 IDENTIFICACIÓ DELS RISCOS

Una forma de descriure els riscos és mitjançant una quadricula que proporcioni a l'equip desenvolupador una tècnica senzilla per la projecció de cada risc. En la taula 6 es poden observar els riscos més rellevants en aquest TFC i que tot seguit descrivim les columnes que la componen:

- **Risc**. Descripció del risc.

- **Categoria**. Valor que indica el impacte del risc en cas de que ocorri i que es classifica en: catastròfic (1), quan el projecte queda interromput o es cancel·la. Crític (2) en els casos en que el projecte queda temporalment interromput; Marginal (3) quan el risc no és greu però alenteix el ritme normal del pla de treball; Menyspreable (4), són inconvenients de fàcil solució que no alenteixen o interrompen el projecte.

- **PC**. És l'identificador d'un apunt del pla contingència que descriu les mesures previstes davant un risc en particular.

[5] PRESSMAN, ROGER S.: Ingeniería del software: Un enfoque práctico. McGraw-Hill (2002)

Taula 6. Riscos identificats

Risc	Categoria	PC
Avaria dels recursos de maquinari	2	PC02-1
Fallada o pèrdua dels recursos de programari	2	PC02-2
Pèrdua dels arxius i dades del projecte	1	PC01-1
El programari no s'adequa als requisits del client	1	PC01-2
Fallada temporal del subministrament elèctric	4	PC04-1
Baixa temporal d'algun membre de l'equip	2	PC02-3
Personal sense experiència	2	PC02-4
Falta de formació del personal	2	PC02-5
Detecció tardana d'errors en la implementació del programari	1	PC01-3
Inadaptació dels usuaris amb l'aplicació	3	PC03-1
Retalls pressupostaris en el projecte	2	PC02-6
Falta de comunicació amb el client	3	PC03-2
Utilització de tècniques i eines antiquades per desenvolupar el projecte	3	PC03-3
El client desitja canviar o afegir funcionalitats	2	PC02-7

2.5.2 PLA DE CONTINGÈNCIA

Durant el procés de desenvolupament de programari és quasi impossible que no apareguin problemes i incidències que poden fer que el projecte fracassi. L'anàlisi de riscos per si mateix no és suficient per combatre els efectes de les incidències que poden succeir. Per aquest motiu, a més de l'anàlisi de riscos hi ha que elaborar un pla de contingència que contengui les mesures tècniques, humanes i organitzatives necessàries per garantir la continuat del projecte.

En la taula 7 es detallen les accions a seguir en cas que aparegui algun dels riscos identificats en la taula anterior.

Taula 7. Pla de contingència

ID	Pla de contingència/Acció
PC01-1	Es realitzaran varies còpies de resguard del projecte de forma setmanal
PC01-2	Al utilitzar una metodologia iterativa incremental, l'aplicació s'anirà completant en vàries fases, d'aquesta forma el client disposarà de noves funcionalitats en cada versió lliurada que podrà provar sense esperar a tenir el sistema complet. Aquest fet, juntament amb les reunions que es realitzaran de forma periòdica permetrà detectar requisits malinterpretats
PC01-3	En cada iteració, al final de la fase d'implementació de la versió corresponent es realitzaran les tasques de *testing*
PC02-1	L'equip compta amb un equip de reserva per continuar amb el projecte
PC02-2	El programari utilitzat per desenvolupar el TFC es de llicència lliure i es pot aconseguir en qualsevol moment en les webs dels fabricants
PC02-3	L'equip al comptar només amb un analista/programador, aquest podrà recuperar les hores perdudes ampliant la jornada o en caps de setmana
PC02-4	Els membres de l'equip que no comptin amb la suficient experiència seran supervisats i guiat per personal més experimentat, en aquest cas per el *Projecte manager*
PC02-5	L'empresa facilitarà la formació per el reciclatge dels seus empleats
PC02-6	Aquest risc afectaria sobretot a la quantitat de personal, ja que el principal cost de l'empresa està en el salari dels empleats. En cas de prescindir d'algun treballador les seves tasques tindrien que ser assumides per la resta de l'equip
PC02-7	Es tindrà que aplicar un nova fase d'anàlisis i disseny i estudiar els efectes de les noves funcionalitats. Al ser una exigència no contemplada el client tindrà que assumir el costos econòmics, la possible alteració en el pla de treball i les variacions que puguin sofrir algunes funcions ja implementades
PC03-1	El sistema serà implantat de forma progressiva, així els usuaris no tindran que aprendre a menjar moltes funcions en poc temps. A més, s'entregarà un manual d'usuari on s'expliqui el funcionament de l'aplicació
PC03-2	Es mantindran les reunions que facin falta en cada fase per tal de mantenir un contacte continu amb el client i estar al corrent de les exigències o dubtes que puguin sorgir
PC03-3	L'equip utilitzarà les tècniques més adequades segons l'arquitectura del programari
PC04-1	Entre els recursos de maquinari l'equip disposarà de almenys una unitat SAI

2.6 ACTUALITZACIÓ DEL PLA DE TREBALL

El pla de treball del TFC s'ha vist alterat degut a una reorganització de les tasques realitzades en la fase d'anàlisi. Aquests canvis també han afectat a l'etapa del disseny i per tant s'ha dut a terme una actualització i reestructuració de la planificació. En la figura 8 es pot observar com queden distribuïdes les tasques per la segona i tercera fase del projecte.

	Nombre de tarea	Comienzo	Fin	Duración
1	**Segona fase: Anàlisi**	**jue 15/03/12**	**mié 18/04/12**	**25 días**
2	Reunió amb el client (recollida requisits)	vie 16/03/12	vie 16/03/12	1 día
3	Identificar actors del sistema	lun 19/03/12	lun 19/03/12	1 día
4	Revisar funcionalitats	lun 19/03/12	mar 20/03/12	2 días
5	Diagrama de casos d´ús	mié 21/03/12	lun 26/03/12	4 días
6	Descripció dels casos d'ús	mar 27/03/12	vie 06/04/12	9 días
7	Anàlisi de riscos	lun 09/04/12	mar 10/04/12	2 días
8	Pla de contingència	mié 11/04/12	mié 11/04/12	1 día
9	Elaboració document a lliurar	jue 12/04/12	mar 17/04/12	4 días
10	Reunió client (revisar estat actual projecte)	mié 18/04/12	mié 18/04/12	1 día
11	Fita: Lliurament PAC 2	mié 18/04/12	mié 18/04/12	0 días
12	**Tercera fase: Disseny**	**jue 19/04/12**	**mié 23/05/12**	**25 días**
13	Arquitectura de l'aplicació	jue 19/04/12	jue 19/04/12	1 día
14	Definir subsistemes	jue 19/04/12	jue 19/04/12	1 día
15	Diagrama de classes	vie 20/04/12	mar 24/04/12	3 días
16	Diagrama de seqüència	mié 25/04/12	mar 01/05/12	5 días
17	Disseny Interfícies d'usuari	mié 02/05/12	mié 09/05/12	6 días
18	Reunió client (revisar estat actual projecte)	mié 09/05/12	mié 09/05/12	1 día
19	Disseny persistència	jue 10/05/12	mar 15/05/12	4 días
20	Elaboració document a lliurar	mié 16/05/12	mar 22/05/12	5 días
21	Fita: Lliurament PAC 3	mié 23/05/12	mié 23/05/12	0 días

Figura 8. Nou pla de treball per les fases d'anàlisi i disseny

En concret l'actualització d'ambdues fases consisteix en:

- Anàlisi:
  - Incloure un anàlisi de riscos
  - Incloure un pla de contingència
  - Afegir una segona reunió amb el client per revisar la situació del projecte
- Disseny
  - La definició dels subsistemes passa a realitzar-se de l'anàlisi al disseny

TERCERA FASE: Disseny

3 DISSENY

Amb aquest document s'inicia la fase de disseny de l'aplicació WFM que fa la funció de pont entre l'anàlisi i la implementació. Si l'etapa anterior consistia en plantejar el problema a resoldre definint "que s'ha de fer" mitjançant els requisits, ara es tracta d'elaborar un model que especifiqui "com s'ha de fer" i que a més serveixi de guia per a la codificació del sistema.

En el disseny OO es dedica especial atenció a la definició dels objectes del programari i com col·laboren entre ells per a satisfer els requisits detectats en l'anàlisi.

Els temes principals que es tractaran en els pròxims apartats són:

- L'arquitectura del programari.
- La descripció dels subsistemes.
- Disseny del model de domini.
- Disseny de la persistència.
- Disseny de les interfícies gràfiques d'usuari.
- Criteris d'usabilitat

3.1 ARQUITECTURA DEL PROGRAMARI

Una vegada compresos els requeriments que el projecte ha de satisfer, la primera decisió plantejada en l'etapa de disseny ha estat definir l'arquitectura en la qual el sistema s'ha de fonamentar. El fet d'elegir un tipus en concret és un punt molt important, doncs determina les decisions de disseny, l'estructura i el funcionament de l'aplicació a desenvolupar.

Seguidament es detallen les diferents tecnologies i models que compondran l'arquitectura del programari a produir en el TFC.

3.1.1 ENTORN DISTRIBUÏT

En la construcció d'una nova aplicació s'utilitzen principalment dos tipus d'entorns, els centralitzats i els distribuïts. Les particularitats de cada projecte marquen l'opció a escollir, elegint evidentment, aquella que s'ajusti millor a les necessitats detectades. Per el cas que ens ocupa, l'alternativa més adequada és un medi distribuït. El principal motiu per fer aquesta elecció es que els clients d'*Armaçons de Fusta, S.L.* han de poder accedir a l'aplicació de forma remota, entre altres coses, per realitzar les seves comandes.

Per una altre part, existeixen altres raons importants per fer servir aquest entorn, per exemple l'escalabilitat, facilitant així l'ampliació del sistema a mesura que sigui necessari. També es guanya en transparència a nivell de dades, els usuaris poden accedir a elles sense saber on estan ubicades i també, a nivell d'execució. Els processos es poden executar indistintament en diversos ordinadors sense que els usuaris sàpiguen a quin s'executen.

3.1.2 MODEL CLIENT/SERVIDOR

Actualment, l'entorn distribuït més comú és el model client/servidor. En concret, el que s'aplicarà en el TFC té una arquitectura de tres capes basada en les funcions -veure figura 9- la qual està composta de les següents parts:

- **Clients**. Realitzen les sol·licituds de serveis i mostren els resultats. Segons la càrrega que se'ls vulgui assignar es pot instal·lar i configurar part de l'aplicació, amb la intenció de reduir les tasques que el servidor ha de realitzar. Així, podem parlar de clients lleugers o pesats.

- **Servidors d'aplicacions**. És on resideix la major part de la lògica de l'aplicació i manipulació de dades.

- **Servidors de dades**. Emmagatzemen les bases de dades que l'aplicació i l'usuari utilitzen, les quals poden estar en un equip independent o bé al mateix servidor d'aplicacions.

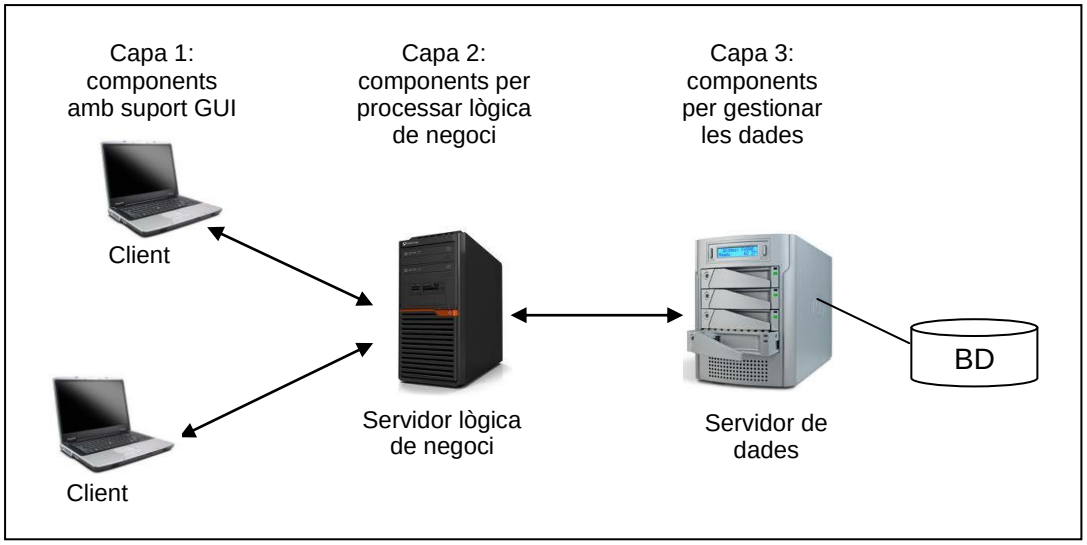


Figura 9. Model client-servidor de tres capes

3.1.3 LES TECNOLOGIES JAVA I RMI

Tot i que en la fase de disseny no és imprescindible saber quin llenguatge s'utilitzarà per implementar l'aplicació, s'ha decidit des de un principi que l'opció escollida per efectuar aquesta tasca sigui Java. El motiu principal es deu a que és la tecnologia utilitzada durant tota la carrera i en conseqüència, la que millor domina el personal encarregat de fer la codificació del sistema. Finalment això es pot traduir en poder obtenir un producte de qualitat més alta.

Un altre raó per haver fet aquest elecció, és que Java proporciona la seva pròpia solució per a la invocació remota de mètodes, coneguda com RMI (*Remote Method Invocation*). Al ser una part estàndard de l'entorn d'execució Java, el seu ús resulta ser un mecanisme simple i robust per a desenvolupar aplicacions distribuïdes. No obstant, en algunes situacions aquesta avantatge es pot convertir en una limitació, ja que qualsevol sistema basat en RMI obliga a emprar el llenguatge Java de forma exclusiva.

En la figura 10[6] es resumeix gràficament l'arquitectura de la tecnologia RMI. Bàsicament l'estructura es divideix en 4 capes:

[6] Gràfic recollit del llibre: *Aplicaciones distribuidas en Java con tecnología RMI*. Caballé, Santi i Xhafa, Fatos (2008). Delta Publicaciones Universitarias

Capa 1. Nivell d'aplicació. Se correspon amb la implementació real de les aplicacions client i servidor. Es on se realitzen totes les crides d'alt nivell per accedir i exportar objectes remots.

Capa 2. Nivell d'interfícies intermèdies o capa Stub-Skeleton. És on s'envien les peticions del client i retornen els resultats del servidor. La comunicació es realitza mitjançant *sockets*.

Capa 3. Nivell de gestió de referències. Responsable de la gestió de referències de les crides remotes. En aquesta capa s'espera una connexió de tipus *stream*.

Capa 4. Nivell de transport. Estableix les connexions necessàries per establir la comunicació i transportar les dades d'un equip a un altre. El protocol utilitzat és el TCP/IP.

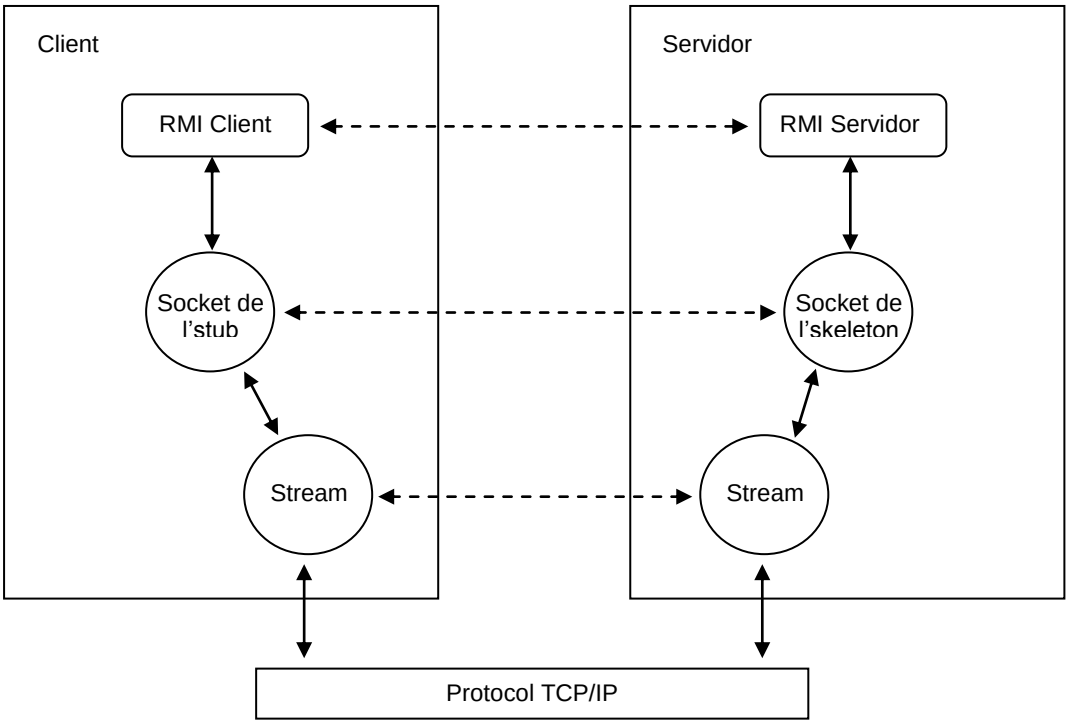


Figura 10. Arquitectura RMI

3.1.4 EL PATRÓ MVC

Un dels objectius que es persegueix en el disseny de l'aplicació és aconseguir un alt grau de modularitat, sobretot, tot el que concerneix a la interfície d'usuari. Aquest aspecte, segons alguns autors, pot arribar a suposar el 60% de les línies de codi del total d'un sistema (Sanz 1996). Amb aquesta xifra, és fàcil deduir quin volum de feina pot requerir una actualització de les pantalles si la seva implementació es troba barrejada amb la resta de codi de l'aplicació.

La forma d'obtenir una clara separació del sistema en varies parts independents però que col·laborin entre si, és mitjançant el patró Model-Vista-Contralador o MVC. Es tracta d'un model que divideix l'aplicació en tres parts, propiciant que els canvis aplicats en una repercuteixen el més mínim en les altres.

Les capes que s'obtenen són les següents:

- **Model**. Se correspon al model del domini de l'aplicació.
- **Vista**. És la interfície gràfica d'usuari de l'aplicació.
- **Controlador**. Gestiona el flux de l'aplicació. Fa d'intermediari entre la vista i el model.

I el funcionament es pot resumir amb les següents passes:

1- **Capa vista**. L'usuari realitza alguna operació per mitjà de la interfície gràfica.
2- **Capa controlador**:
    2.1- Rep l'esdeveniment provocat per l'acció de l'usuari.
    2.2- Gestiona l'esdeveniment i accedeix a la capa model.
    2.3- Dona l'ordre a la vista perquè s'actualitzin les dades.
3- **Capa vista**. S'actualitza amb les dades que arriben des de el controlador.

El patró MVC es pot implementar mitjançant la classe *Observer* i *Observable* de Java o utilitzant alguns Frameworks coneguts com Struts, Spring o Grails.

3.2 DESCRIPCIÓ DELS SUBSISTEMES

La complexitat[7] d'un sistema augmenta a mesura que s'afegeixen noves funcionalitats. Aquest fet, requereix l'aplicació d'alguna estratègia que permeti controlar la situació per evitar dificultats en la gestió i desenvolupament. Un mètode molt útil per aconseguir-ho és fragmentant el sistema global en unitats més petites o subsistemes. Això permet que en lloc de treballar en una gran i única tasca, es faci sobre blocs més reduïts i simples de desenvolupar.

Si es considera que la major font de complexitat d'un sistema prové de les relacions entre objectes, la meta és reduir o manejar de la millor forma aquestes col·laboracions organitzant-les en grups. Així, si observem que certs objectes estan molt relacionats entre sí però no amb altres, és millor assignar-los a un subsistema o paquet comú.

Seguint aquesta recomanació, s'ha previst que el programari WFM com a sistema global, estigui composat dels següents subsistemes:

- **Subsistema manteniment**. Permet realitzar la gestió de les dades principals de l'aplicació, como són: gestió de rols, gestió d'usuaris, gestió de paràmetres de l'aplicació, gestió de magatzems, gestió de models, gestió dels tipus de materials, gestió de materials, gestió de clients, gestió de proveïdors i gestió d'articles.

- **Subsistema connexió**. La funcionalitat contemplada en aquest TFC és la de controlar l'accés dels usuaris a l'aplicació segons sigui el seu perfil o rol, mostrant els menús i opcions que se'ls hagi assignat. També permet el canvi de contrasenya d'un usuari.

- **Subsistema flux de treball**. És l'encarregat d'efectuar els processos diaris del negoci com les comandes, compres a proveïdors o la gestió de l'estoc. Aquests són els fluxos contemplats en el projecte actual, però en les extensions que s'aniran realitzant s'afegiran altres operacions com la facturació.

[7] Extret del llibre: A. WEITZENFELD. *Ingeniería de software orientada a objetos con UML, Java e Internet.* Paraninfo Thomson Learning (2005).

- **Subsistema informes.** Tot i que es deixa pendent per una pròxima ampliació de l'aplicació, aquest subsistema oferirà una sèrie de llistats i estadístiques. Algunes de les que es tenen en ment són: una relació dels articles més venuts, un llistat del consum dels clients, un llistat de l'estoc dels articles, un llistat de factures pendents de cobro, etc.

3.2.1 DIAGRAMA DELS SUBSISTEMES

En la figura 11 es descriu gràficament els subsistemes del programari que seran tractats en el TFC i les relacions existents entre ells. Tot i a aquesta representació, s'ha de destacar que els subsistemes no existeixen durant l'execució de l'aplicació, simplement són abstraccions o entitats conceptuals de com està organitzat el sistema.

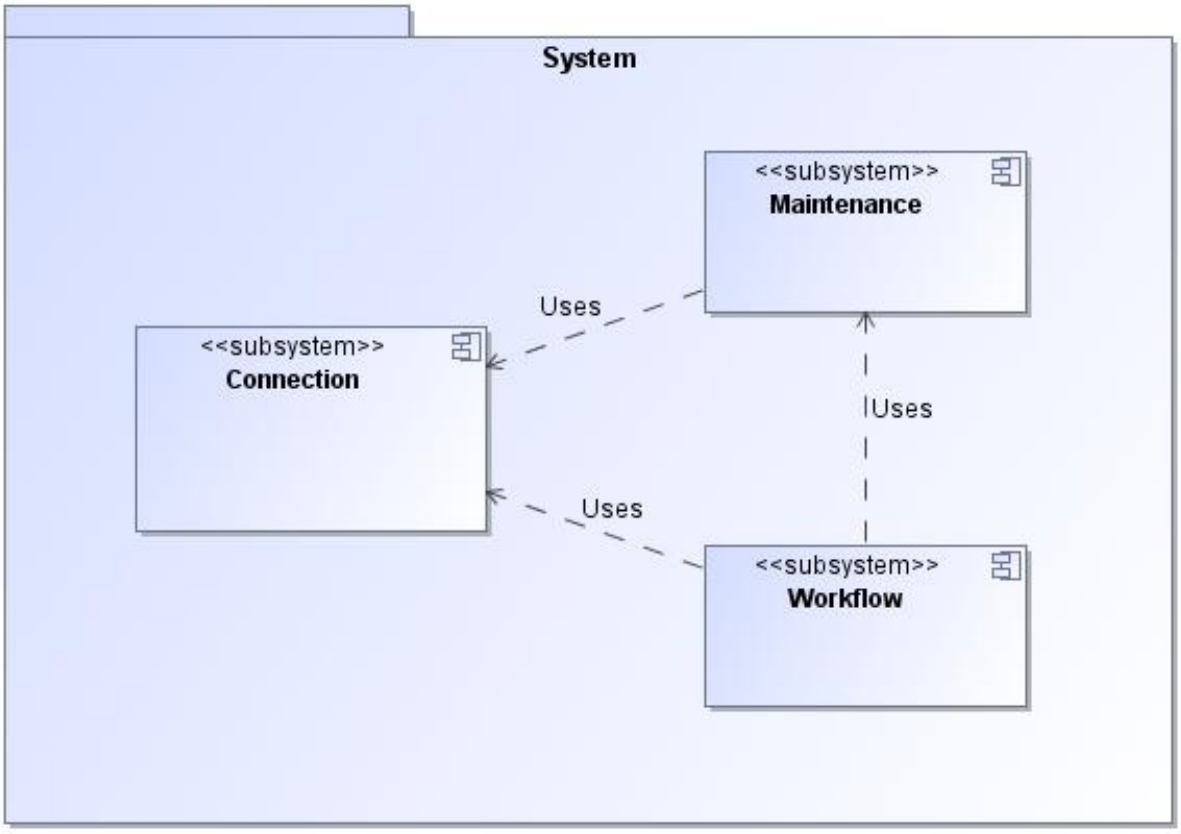


Figura 11. Diagrama de subsistemes

Del diagrama es poden treure les següents conclusions:

- Tant el subsistema de manteniment (Maintenance) com el de flux de treball (Workflow) tenen una relació d'ús amb el de connexió (Connection), perquè les funcionalitats dels dos primers tan sols estan disponibles si l'usuari s'identifica.

- El subsistema flux de treball (Workflow) té una relació d'ús amb el de manteniment (Maintenance) degut a que les diferents activitats diàries registrades en el primer (comandes, compres) utilitzen components del segon (articles, proveïdors, clients).

3.3 MODEL DE DOMINI

Un model de domini[8] captura els tipus més importants d'objectes i esdeveniments que existeixen en l'entorn en el que treballa un sistema. La forma més normal de fer-ho és mitjançant un diagrama format per classes, les quals representa algun concepte de la realitat que es vol modelar.

3.3.1 DIAGRAMA DE CLASSES D'ENTITAT

En la figura 12 es pot observar la representació de l'estructura estàtica de les entitats que l'aplicació a de manejar. Tot i que es pot elaborar un diagrama més detallat incloent els atributs i mètodes de cada entitat, es deixa aquesta tasca per un altre apartat. En aquest cas, l'objectiu és mostrar d'una forma clara i simplificada les classes que s'han detectat i les relacions que hi ha entre elles.

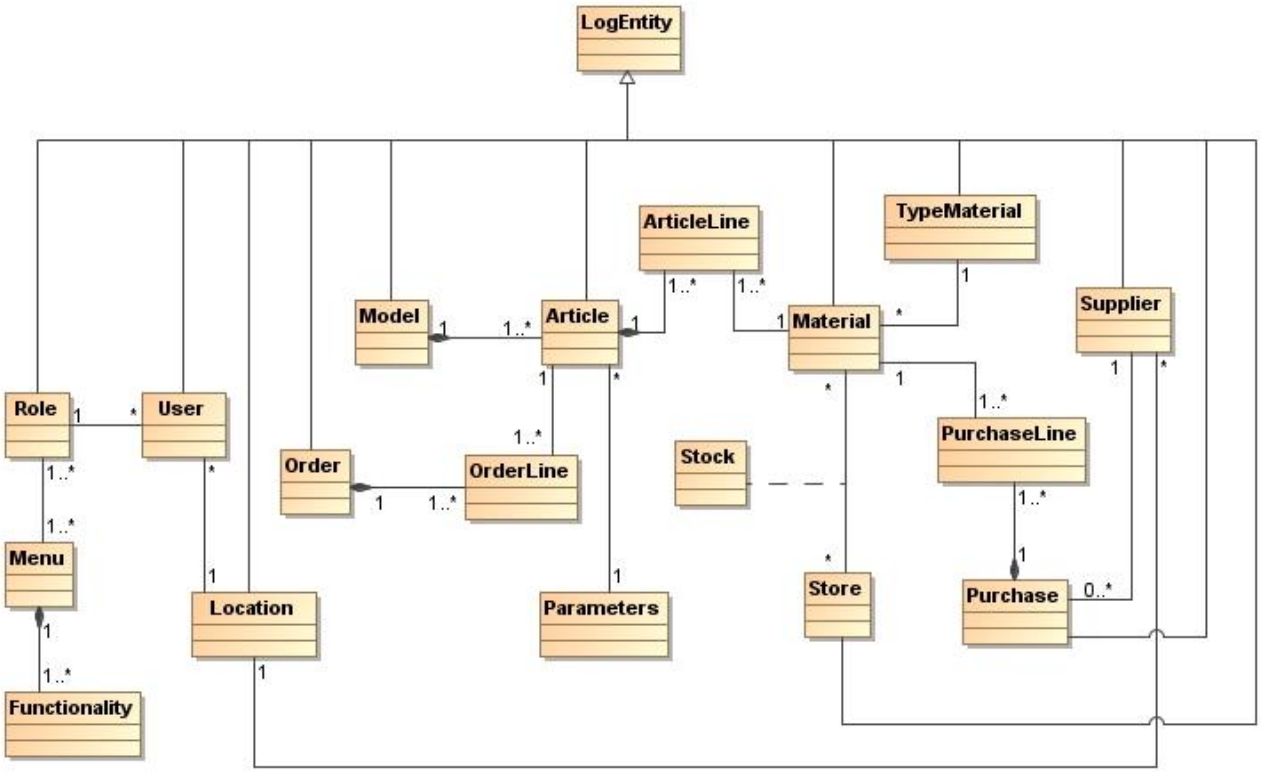


Figura 12. Diagrama de classes d'entitat

[8] I. JACOBSON, G. BOOCK, J.RUMBAUGH. *El proceso unificado de desarrollo de software*. Pearson Educación, S.A. (2000)

3.3.2 GLOSARI DEL DIAGRAMA DE CLASSES D'ENTITAT

- **Article.** Representa un producte final el qual es ven als clients i està compost de varis materials. Es realitza un escandall (càlcul dels costos) per assignar el preu a l'article.
- **ArticleLine.** Representa una línia amb un material que correspon a un article.
- **Functionality**. Descriu una funcionalitat inclosa dins un menú.
- **LogEntity**. Segons el requeriment RF_14, les entitats que tinguin l'opció d'eliminació seguiran emmagatzemades en la base de dades per evitar problemes d'inconsistència. Aquesta classe permet dur el registre d'una entitat, com per exemple, si està activa o no, les dates de quan s'efectuen el canvis i l'usuari que els realitza. Les classes que requereixen aquest control hereten de LogEntity.
- **Material.** Entitat que representa un material concret utilitzat per fabricar un article.
- **Menu**. Classe que descriu un menú amb les funcionalitats que s'assignen a un usuari.
- **Model**. Entitat que representa un model o família composta d'un o varis articles.
- **Order.** Representa una comanda realitzada per un client.
- **OrderLine**. Representa una línia d'una comanda.
- **Parameters**. Classe que defineix els paràmetres generals de l'aplicació.
- **Location.** Descriu una província, una població i el seu codi postal.
- **Purchase**. Representa una compra entre l'empresa i un proveïdor.
- **PurchaseLine**. Entitat que representa un línia d'una compra.
- **Role**. Descriu un tipus d'usuari (administrador, client, etc.).
- **Stock**. Permet controlar l'estoc dels diferents materials.
- **Store.** Entitat que representa un magatzem on es guarden els materials.
- **Supplier**. Proveïdor que subministra material a l'empresa mitjançant les compres.
- **TypeMaterial**. Descriu un tipus de material per indicar la mesura amb que es quantifica el material (unitats, m^2 i m^3).
- **User.** Entitat que descriu un usuari que ha d'accedir al sistema, assignant-li un nom d'usuari i contrasenya per identificar-se, a més de les dades personals. La classe contindrà un atribut per indicar si es tracta d'un administrador, un auxiliar administratiu o un client.

3.3.3 DETALL DE LES CLASSES D'ENTITAT

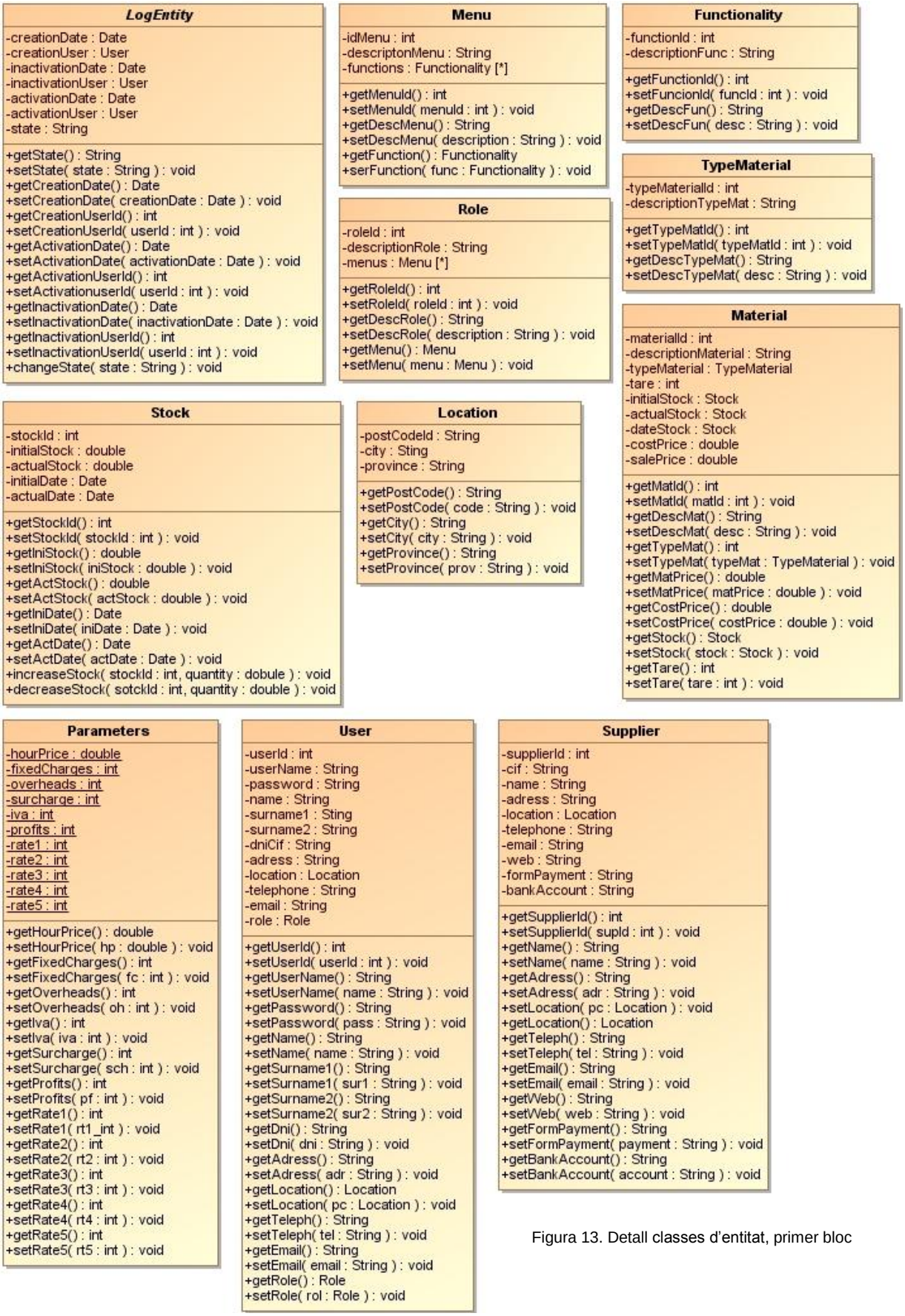


Figura 13. Detall classes d'entitat, primer bloc

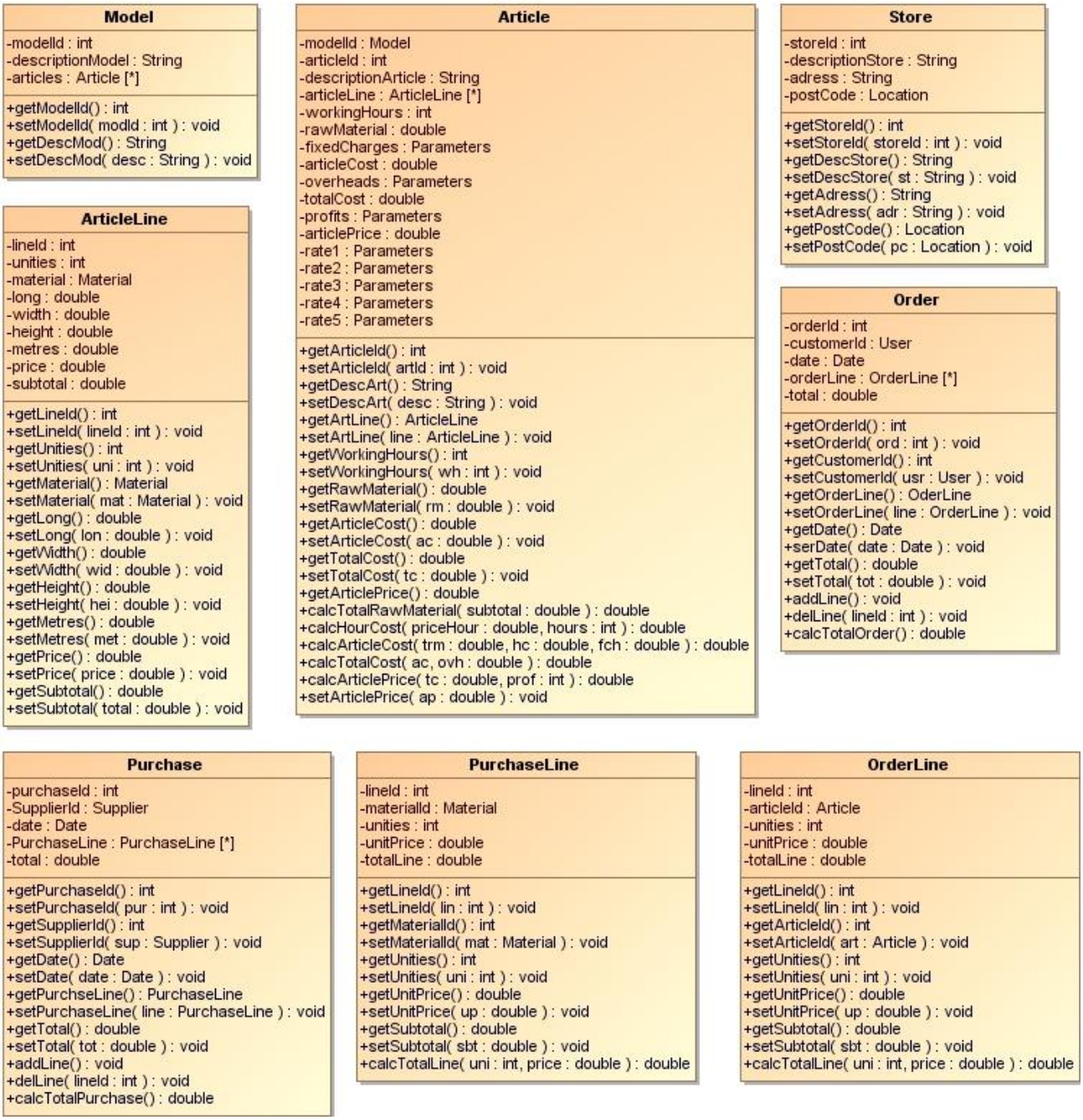


Figura 14. Detall classes d'entitat, segon bloc

3.4 DIAGRAMA DE CLASSES GESTORES

Una classe de control o gestora es pot dir que fa la funció de pont entre les dades i l'aplicació, les quals permeten executar diferents operacions per a tractar les dades de les taules com per exemple, inserir-les, consultar-les o eliminar-les. Des del punt de vista del patró MVC, aquestes classes de control, així com les vistes i les excepcions que veurem més endavant, no s'han inclòs en l'apartat del model de domini perquè són estructures d'implementació i no un intent de representar conceptes directament presents en el domini del problema.

Generalment, a cada entitat de negoci que ha de ser persistent se li assigna una classe de control. En la figura 15 es pot apreciar la jerarquia de les classes gestores que s'han detectat. Cadascuna d'elles hereta de *DiskManager* que està composta dels atributs i mètodes que permetran realitzar les operacions necessàries amb la base de dades, com per exemple, establir la connexió o preparar una sentència i poder-la executar.

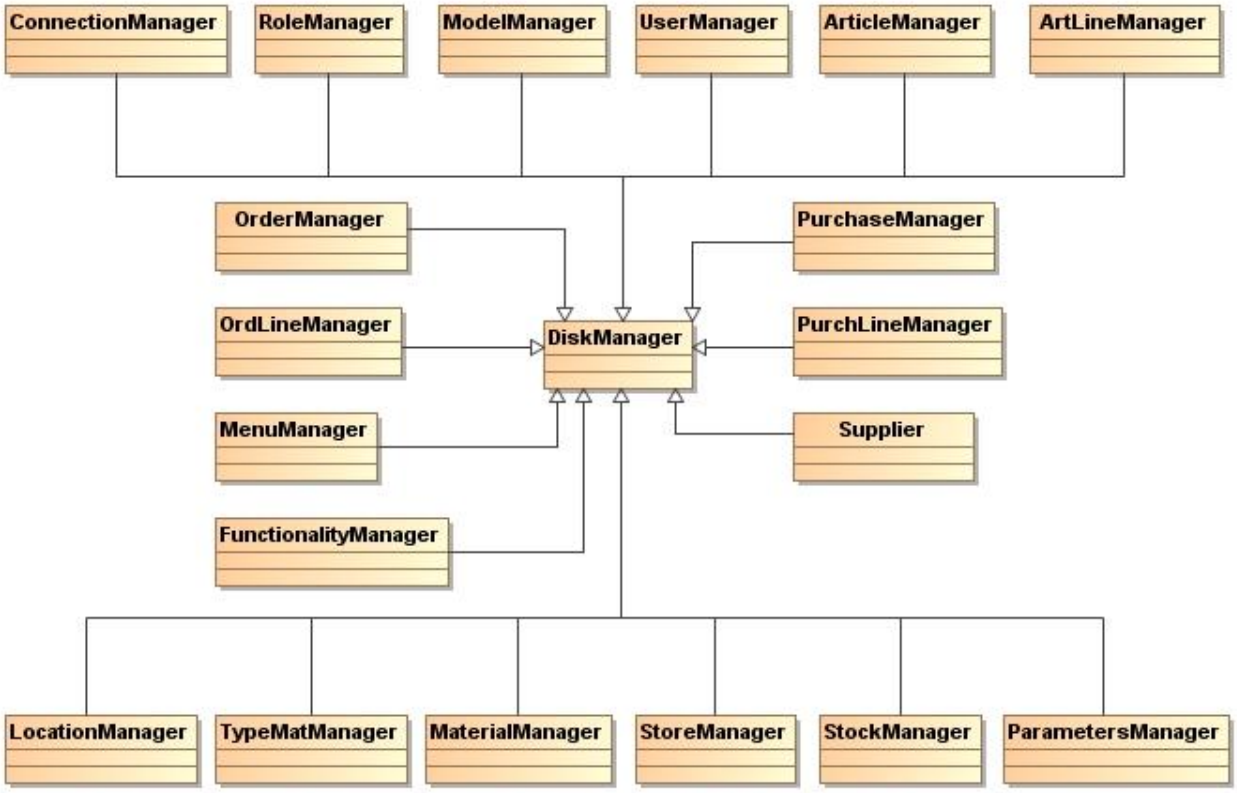


Figura 15. Diagrama de classes gestores

Aquest enfocament permet deslligar les classes de negoci amb l'esquema de la base de dades, fent que l'aplicació guanyi amb portabilitat. De no fer-ho així, al realitzar algun canvi en l'estructura de la base de dades, seria necessari reescriure part del codi de les classes de negoci.

3.4.1 DETALL DE LES CLASSES GESTORES

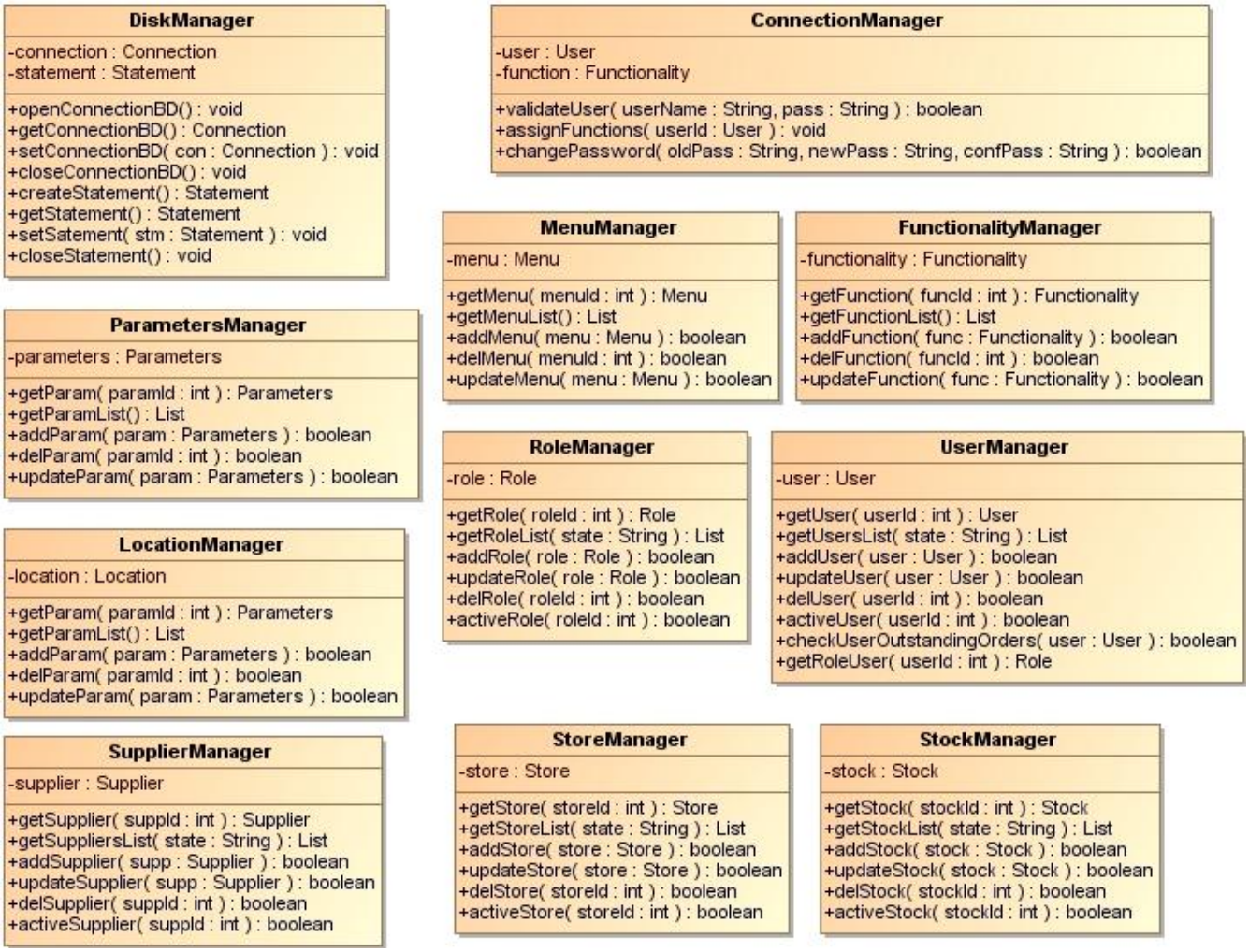


Figura 16. Detall classes gestores, primer bloc

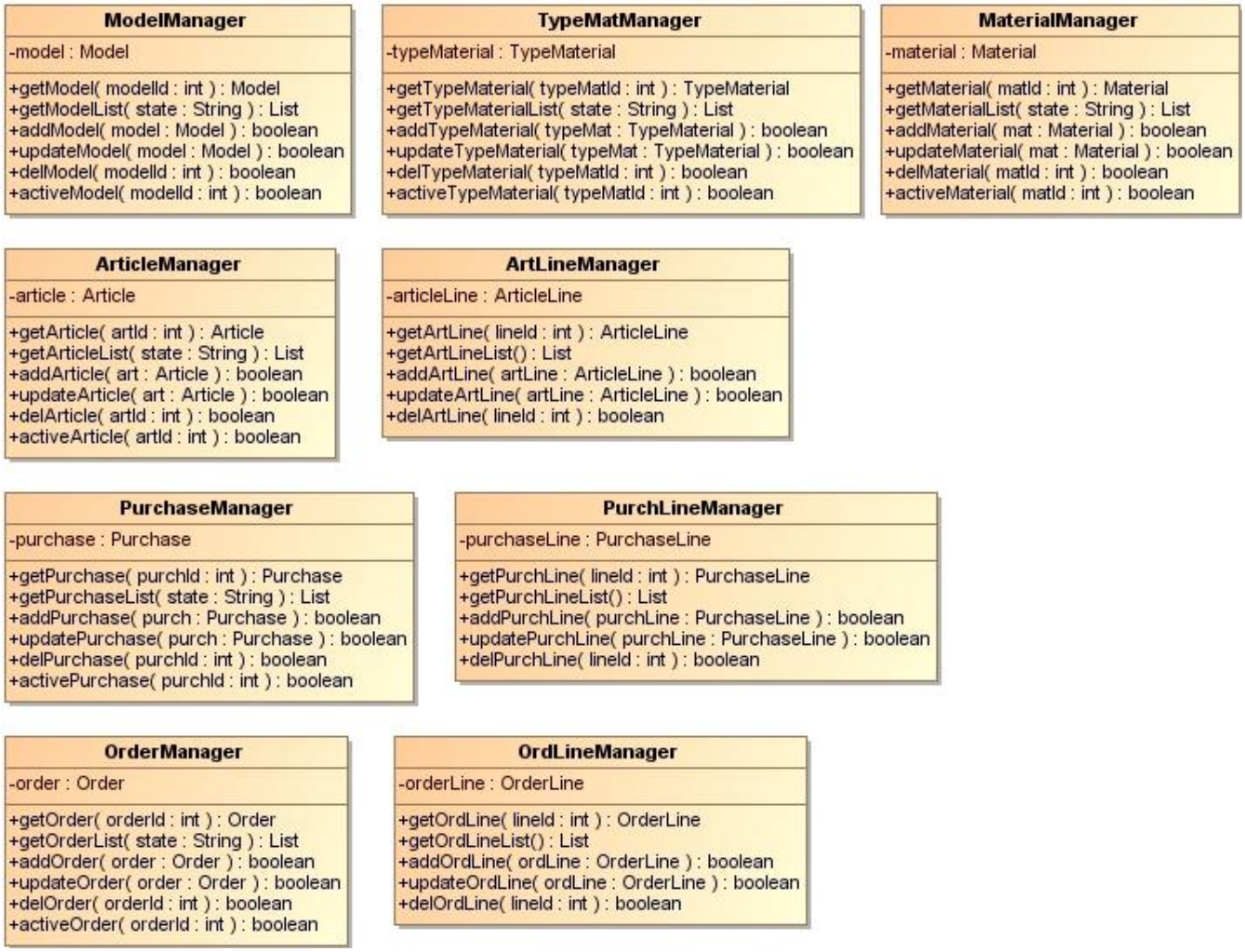


Figura 17. Detall classes gestores, segon bloc

Tant en els apartats anteriors com en els següents, s'ha intentat descriure al màxim el nombre de classes, atributs i mètodes necessaris per cobrir les funcionalitats que aquest projecte té com objectiu. No obstant, pot ocorre que en fases posteriors es detectin algunes mancances o dissenys incorrectes, fet que requeriria aplicar un nou anàlisi i disseny dels temes a tractar.

Per tant, seguint la metodologia iterativa queda oberta la possibilitat de tenir que afegir, eliminar o modificar alguna de les classes, així com els seus atributs i mètodes.

3.5 DIAGRAMA DE PANTALLES

Les interfícies gràfiques són el medi que té l'usuari per interactuar amb el sistema. Per a cada cas d'ús identificat en la fase d'anàlisi li correspon una pantalla que li permet realitzar la seva funció.

En la figura 18 es pot observar la jerarquia d'interfícies d'usuari principals de l'aplicació, amb les que es duran a terme la majoria de les d'operacions.

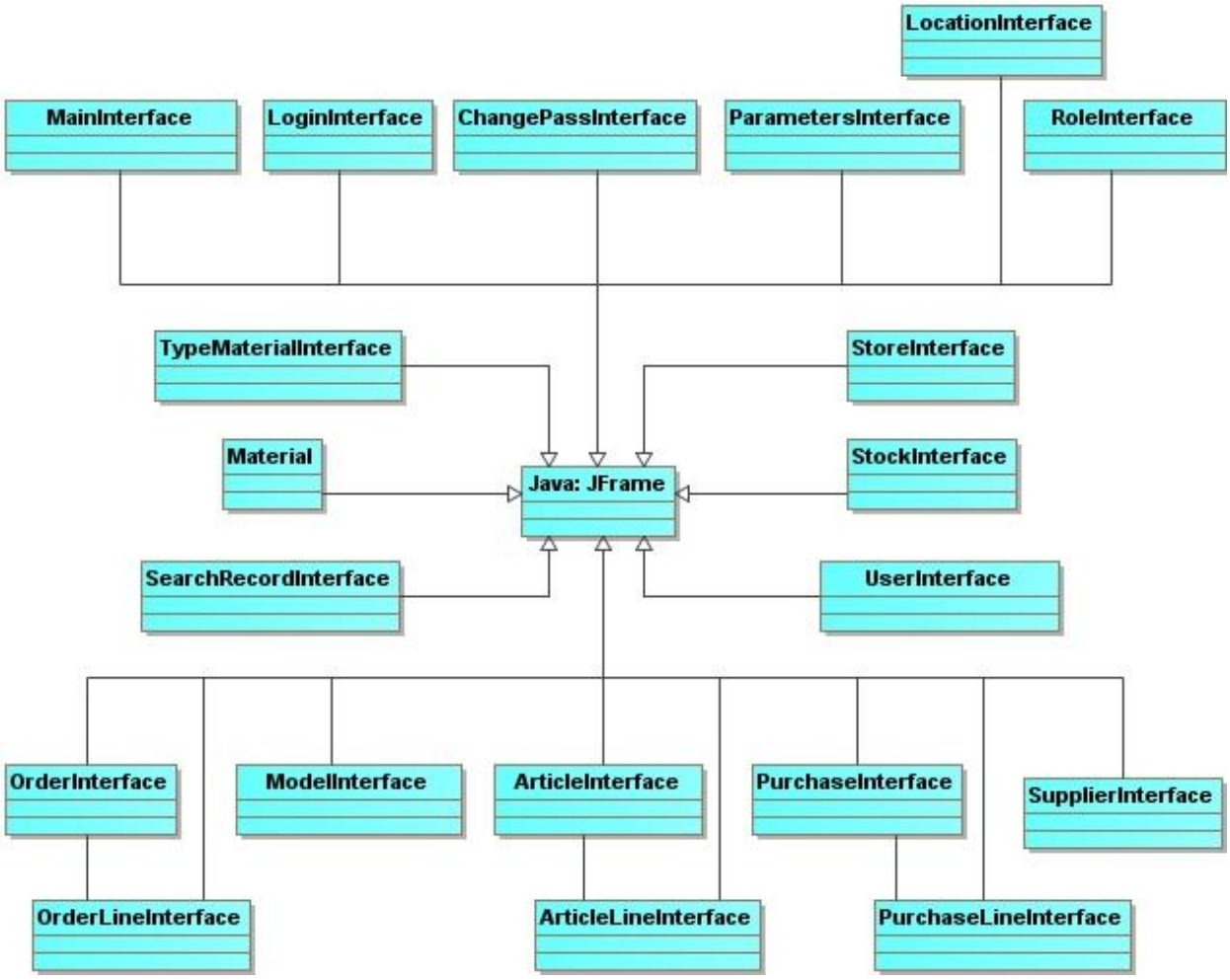


Figura 18. Diagrama de pantalles d'usuari

3.6 DIAGRAMA D'EXCEPCIONS

Si durant l'execució d'algun mètode de l'aplicació succeeix alguna anomalia que no es pot resoldre, el més segur és que el programa finalitzi automàticament i de forma inesperada amb la possibilitant de perdre les dades que no s'hagin guardat. Per evitar aquesta situació el més recomanable és capturar i manejar les excepcions.

Amb aquest propòsit s'han dissenyat les classes de la figura 19 perquè capturin les situacions que podrien originar alguna excepció, com per exemple, errors en entrada o sortida de dades, errors en càlculs (divisió per zero), conversions de tipus o excedir els límits d'una llista.

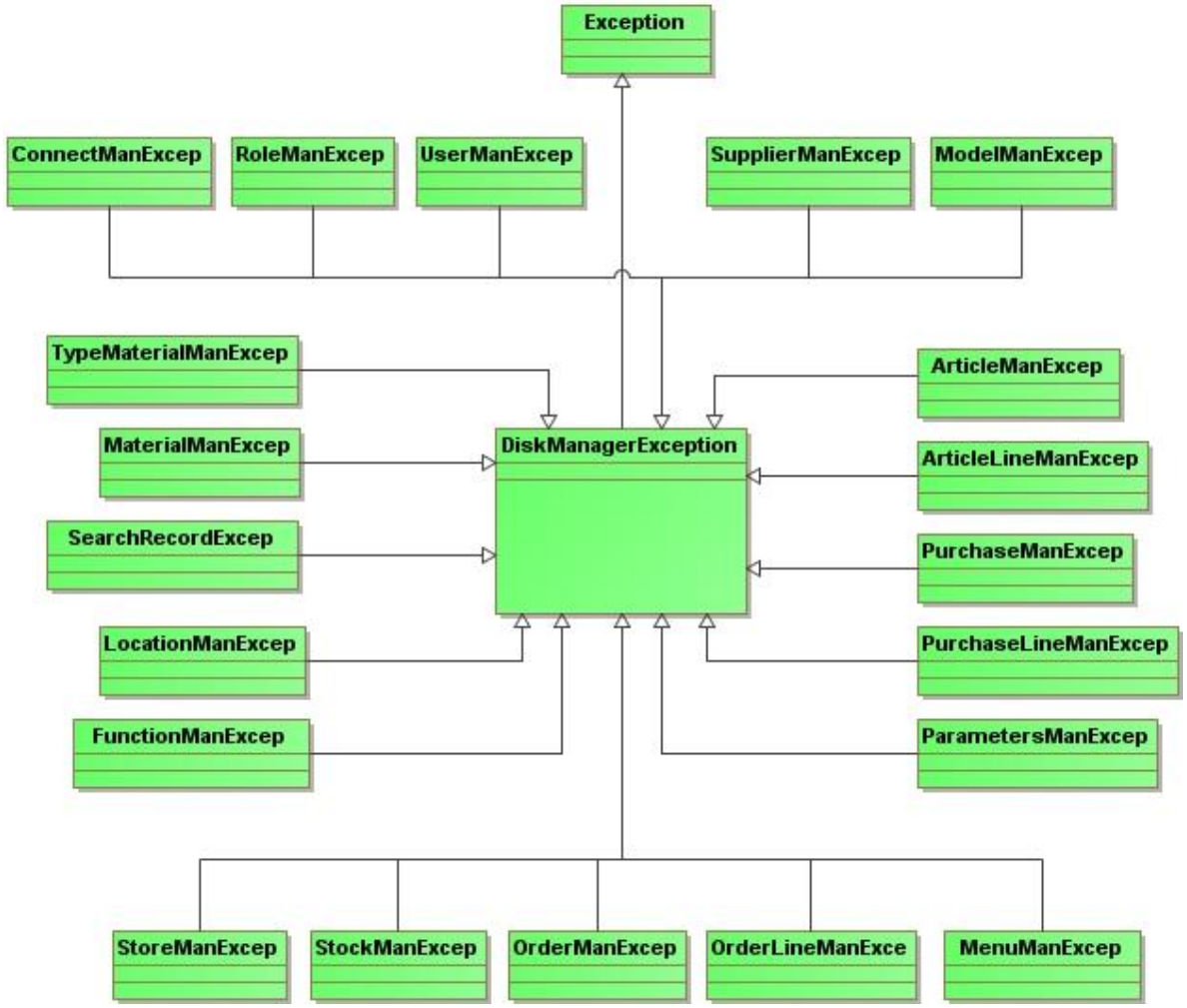


Figura 19. Diagrama d'excepcions

3.7 INTERACCIÓ ENTRE JERARQUIES DE CLASSES

Per tal de que el lector es faci una idea més concreta de com estan relacionades les diferents jerarquies que s'han descrit en els apartats anteriors, en la figura 20 es pot observar un exemple per al cas d'alta d'un usuari.

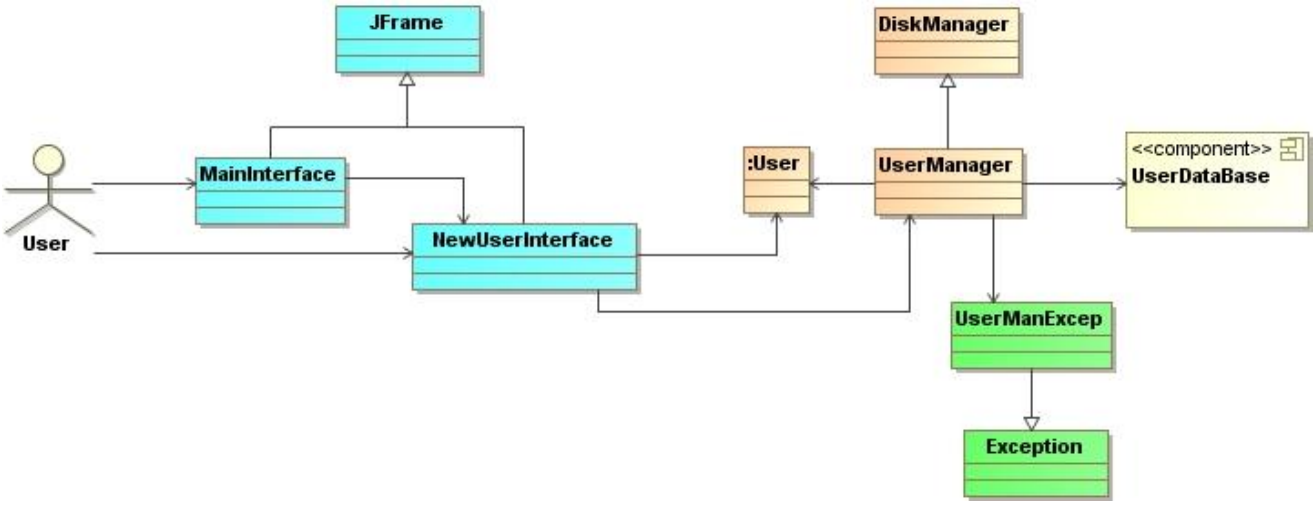


Figura 20. Interacció entre jerarquies de classes

3.8 DIAGRAMES DE SEQÜÈNCIA

Els diagrames de seqüència permeten representar l'intercanvi de missatges entre instàncies d'objectes d'un mateix cas d'ús, descrivint els aspectes dinàmics d'un sistema. Aquest model està compost per uns rectangles amb el nom de les instàncies que intervenen, els quals estan proveïts d'una franja vertical que indica la seva línia de vida durant el procés. Els missatges apareixen en forma de fletxes horitzontals en direcció de d'alt a baix per indicar l'ordre. La circulació, normalment va d'esquerra a dreta i els resultats o respostes de dreta a esquerra.

Una vegada més, per no excedir el volum de pàgines establert en el TFC, a continuació s'exposen dues mostres de diagrama de seqüència per descriuen alguns dels casos d'ús principals. Dirigim al lector a l'annex 2 si desitja veure algun exemple més, tenint en compte que tampoc s'inclouen tots els casos possibles. No obstant, de cara a la implementació si que s'han tingut en compte en el disseny i evidentment es desenvoluparan.

3.8.1 DIAGRAMA DE SEQÜÈNCIA IDENTIFICACIÓ D'UN USUARI

L'usuari, des de la pantalla principal, accedeix al menú *Connection* i a l'opció *Start session*, activant així la pantalla de *login* on s'ha d'introduir el nom d'usuari i la contrasenya. El sistema comprova que les dades són correctes en la base de dades. Si és el cas, es dona accés a l'aplicació activant-ne els menús i opcions assignats a l'usuari. Es disposa de 3 intents per identificar-se, de no fer-ho correctament s'envia un missatge a l'usuari i es tanca la interfície sense tenir accés al sistema.

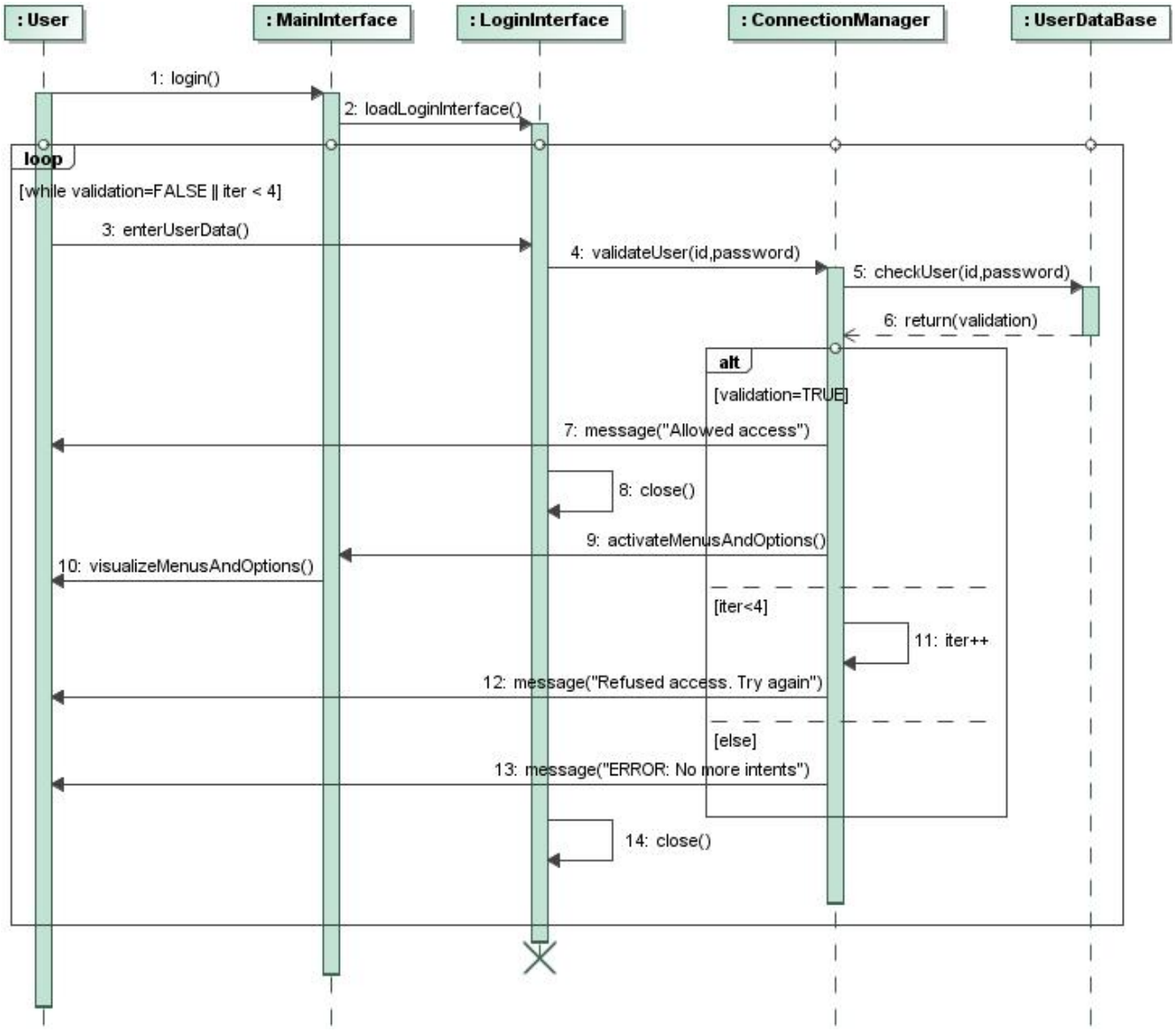


Figura 21. Diagrama de seqüència identificació d'un usuari

3.8.2 DIAGRAMA DE SEQÜÈNCIA CONSULTAR UN ROL

En aquest diagrama, a part de descriure el procés de consulta també s'inclou el de cerca degut a que primer hi ha que buscar el registre a tractar. La interfície de consulta disposa del botó *Search* que activa el cas d'ús *Search record* mostrant la pantalla corresponent. En ella, hi ha que especificar si es vol un rol en estat ACTIVE o INACTIVE (eliminat) a més de l'ID o la descripció de l'objecte a localitzar.

Recordem que des de la pantalla de consulta es pot cercar un objecte en estat INACTIVE (eliminat) per activar-ho de nou si així es desitja.

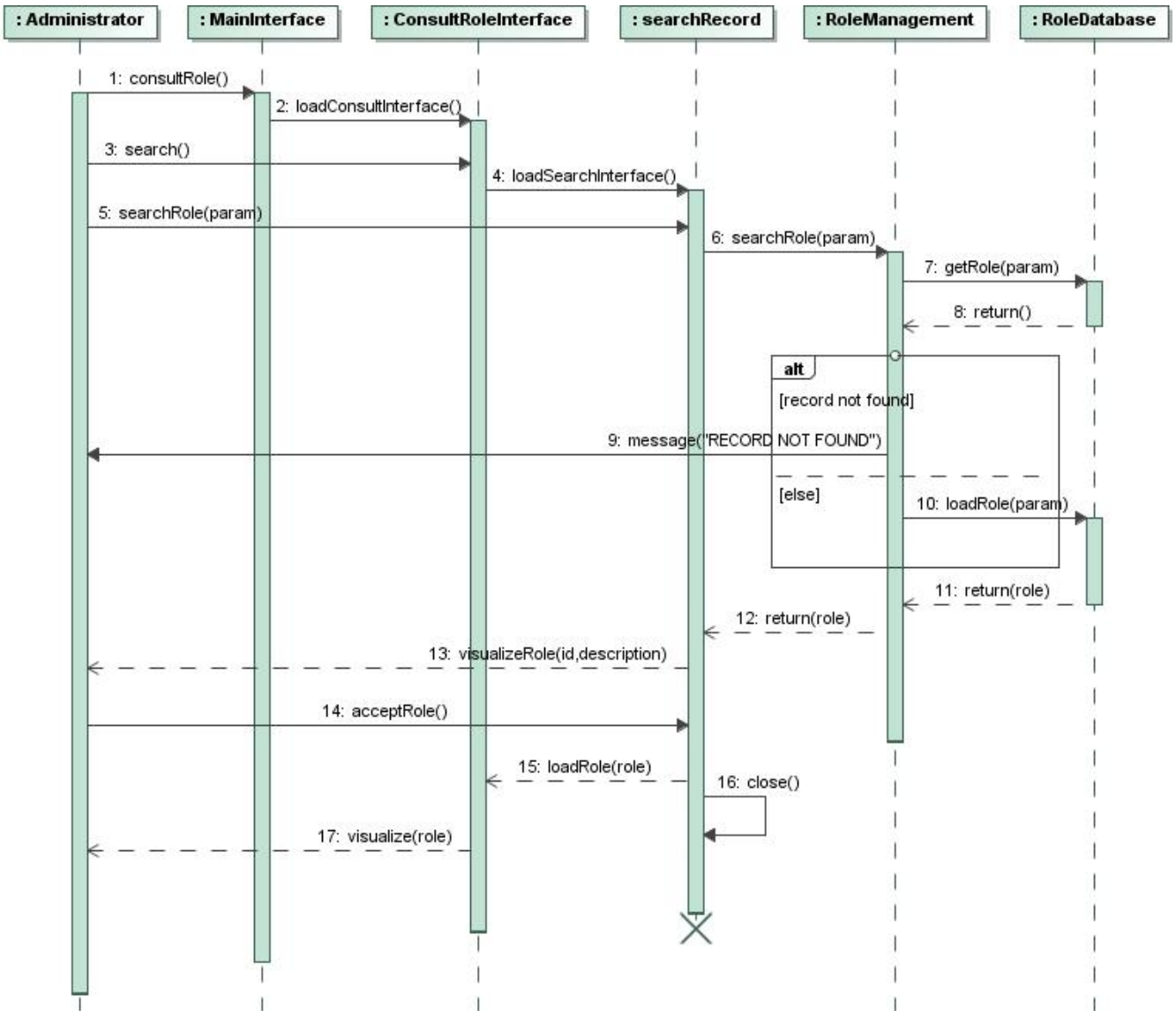


Figura 22. Diagrama de seqüència consultar un rol

3.9 DISSENY DE LA PERSISTÈNCIA

Un aspecte molt important en el desenvolupament de programari és decidir de quina forma s'han d'emmagatzemar les dades que perduraran més enllà de l'execució de l'aplicació. Tot i que hi ha diferents possibilitats, la més comú, sobretot en aplicacions mínimament complexes, és fer servir bases de dades.

Existeixen diverses tipologies de bases de dades però una de les que han tingut més repercussió és el model relacional, el qual, utilitza una estructura composta de taules o relacions que agrupen les dades en files (contenen les dades específiques) i columnes (identifiquen el tipus d'informació). Aquesta solució és la que s'aplicarà en aquest projecte.

3.9.1 DISSENY CONCEPTUAL: EL DIAGRAMA D'ENTITAT-RELACIÓ

El disseny conceptual[9] és una etapa inclosa en el disseny d'una base de dades, que obté com a resultat una estructura de la informació a tractar i que és independent de la tecnologia a utilitzar.

La representació gràfica del disseny conceptual és fa mitjançant un diagrama d'entitat-relació o *entity relationship* (ER), que s'utilitza per modelar els objectes rellevants d'un sistema d'informació. Els elements bàsics que el conformen són les entitats, els atributs i les interrelacions. Com ja passava amb el diagrama de classes, el nivell de detall que es pot aplicar a l'hora d'elaborar el model ER pot variar segons siguin les necessitats. En aquest cas no s'han inclòs els atributs per obtenir un gràfic el més simple i entenedor possible.

En la figura 23 es pot observar el resultat del diagrama ER obtingut.

[9] SISTAC PLANAS, J. *Bases de dades I*. Material de l'Universitat Oberta de Catalunya. Eureca Media, SL (2005)

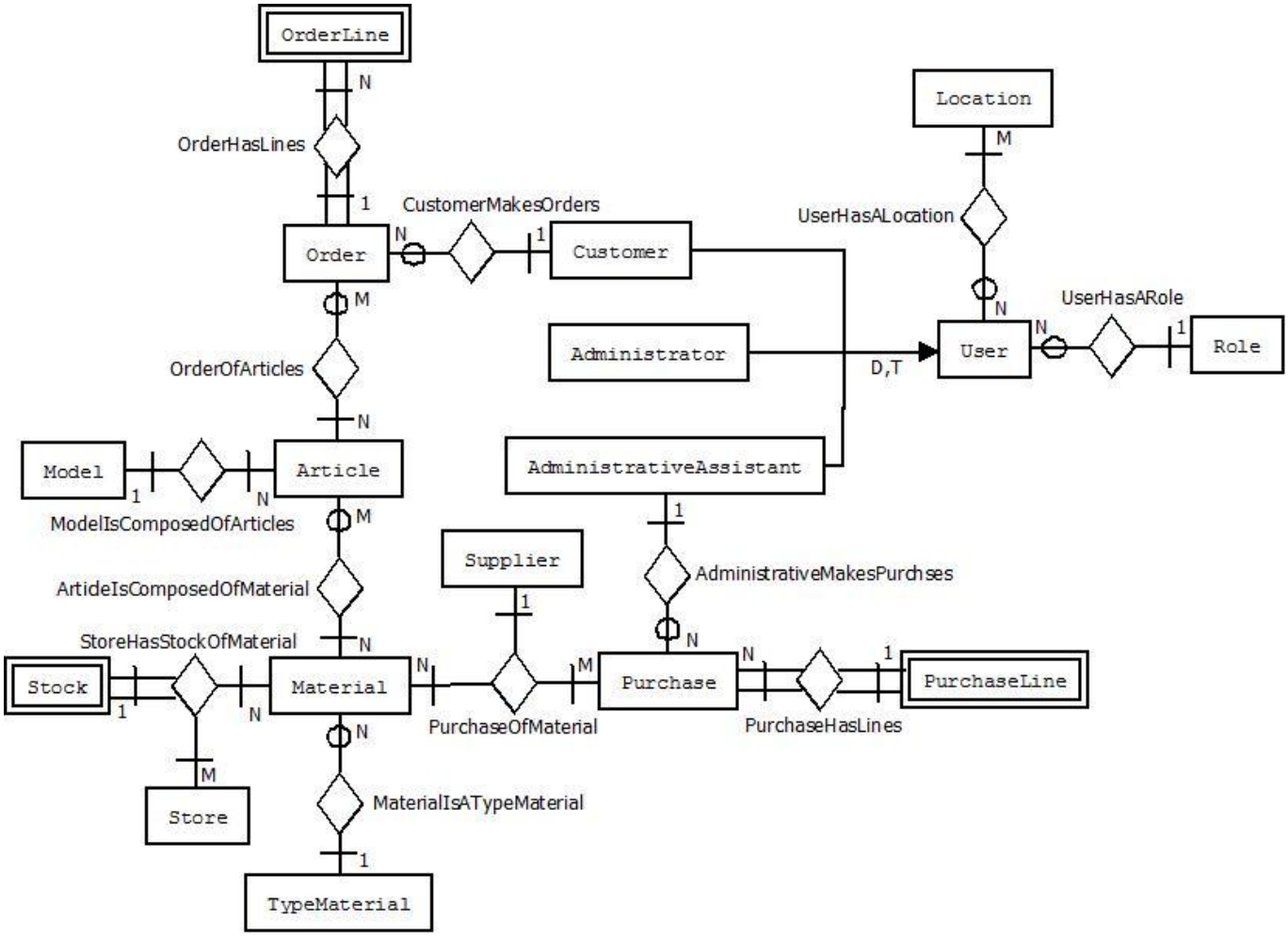


Figura 23. Diagrama entitat-relació

3.10 DISSENY DE LES INTERFÍCIES GRÀFIQUES

En aquesta secció entrem en el disseny de les interfícies gràfiques, medi fonamental pel qual l'usuari pot interactuar amb el sistema. Al haver-hi un gran nombre de pantalles, tal com hem fet en anteriors casos, s'exposen tan sols algunes mostres. Encara que en l'annex 3 es poden veure varis exemples més, la il·lustració completa de les pantalles es deixa com a tasca a desenvolupar quan es confeccioni el manual d'usuari en una fase posterior al TFC. L'objectiu actual es dissenyar alguns prototips perquè el client tingui una visió més real de l'aspecte que tindrà l'aplicació. En cap cas es poden considerar definitives ja que poden sofrir canvis en posteriors revisions del projecte o per indicacions del propi client.

3.10.1 PANTALLA GESTIÓ D'ARTICLES

Des d'aquesta interfície es pot gestionar l'apartat dels articles mitjançant les opcions d'alta, consulta/modificació, eliminació i activació. A la figura 24 es pot veure com inicialment els camps que componen l'article estan desactivats a l'espera de que l'usuari esculli una de les operacions esmentades.

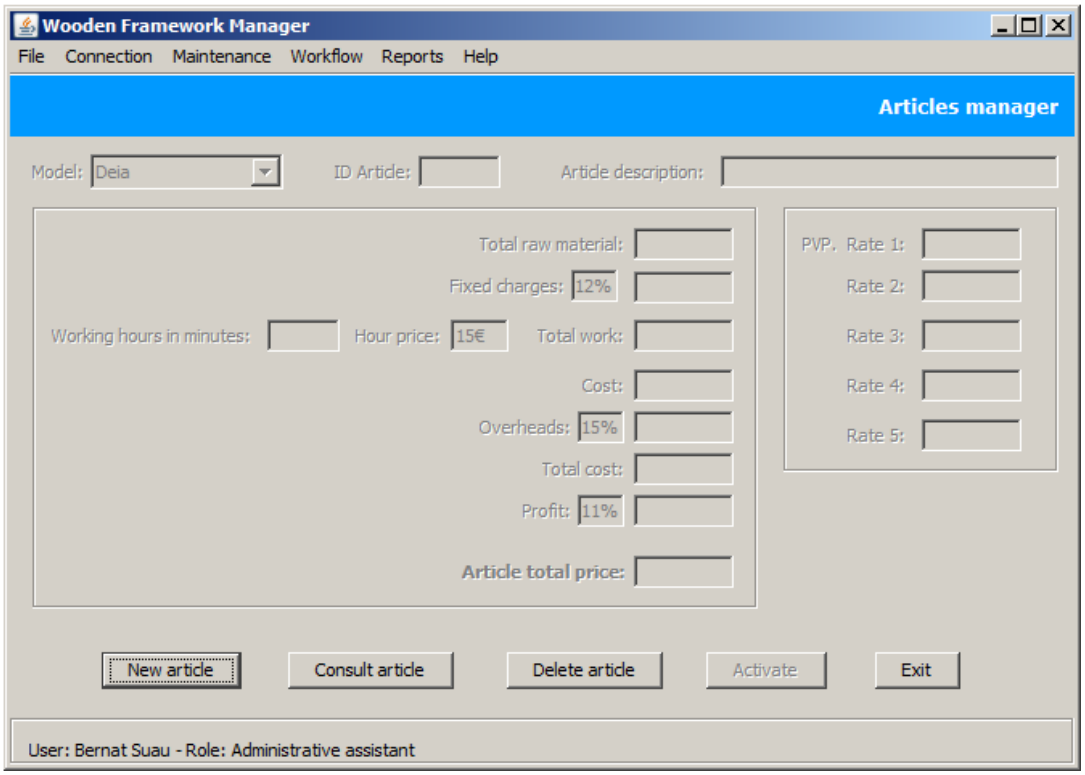


Figura 24. Pantalla gestió d'articles

3.10.2 PANTALLES ALTA D'ARTICLE

Des de la interfície de gestió d'articles, al prémer el botó *New article*, apareix una nova pantalla per introduir les línies que indiquen els materials que componen l'article, iniciant així l'operació d'escandall. El procés requereix en primera instància escollir el model al qual pertany l'article i la seva descripció. Pel que fa a l'ID no falta especificar-ho doncs es tracta d'un camp automàtic.

Seguidament s'introduirà en cada fila un material i les unitats utilitzades. Per els casos de tipus de material mesurat en metres cúbics, a més, farà falta indicar la llargada, l'amplada i l'altura. En la següent figura es pot veure un exemple.

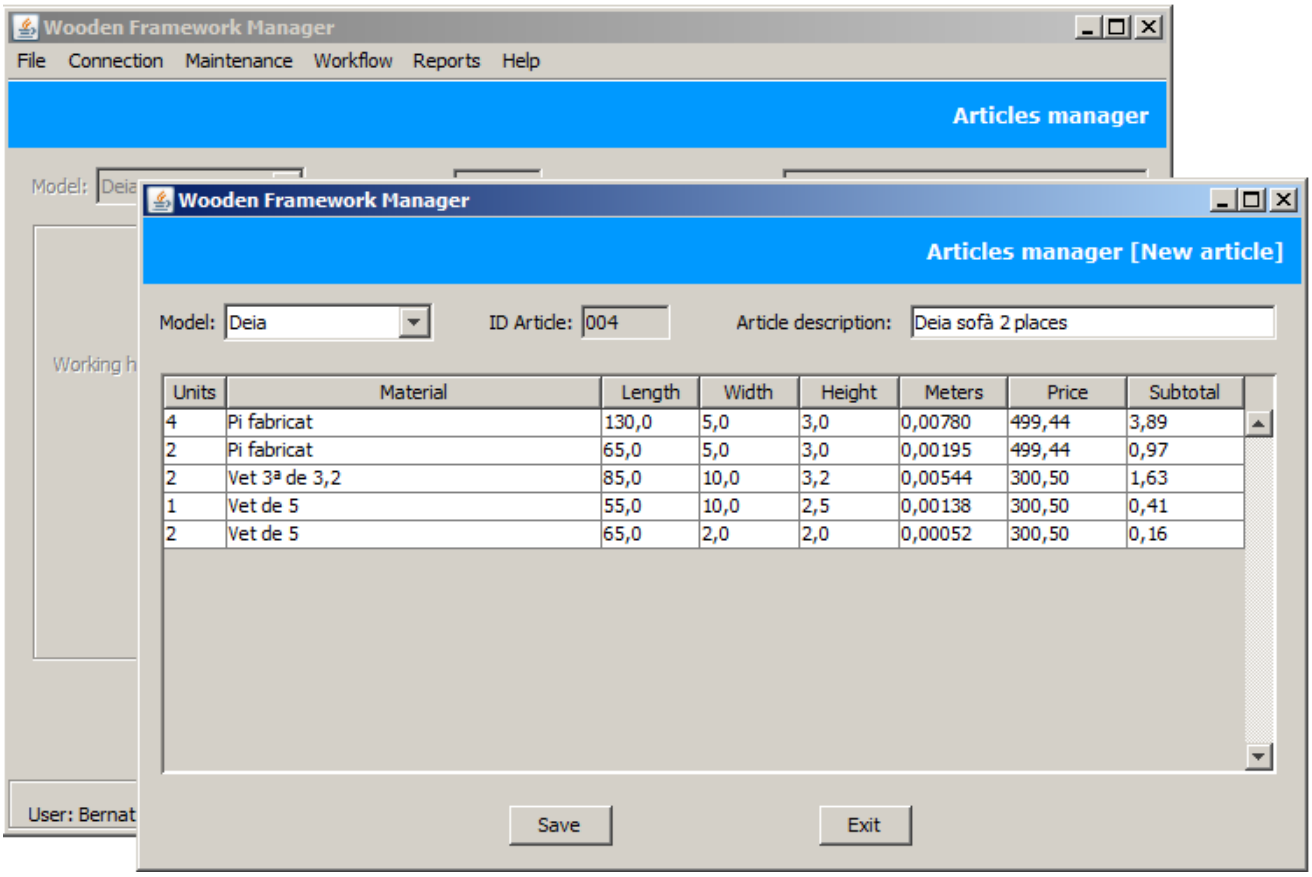


Figura 25. Pantalla introducció de línies d'article

Una vegada indicats tots els materials que componen l'article, hi ha que prémer el botó *Save* perquè el sistema guardi les dades.

Situats de nou en la pantalla d'alta, veurem que s'han activat i actualitzats els camps amb les dades introduïdes anteriorment. Només cal emplenar l'apartat d'hores treballades –el valor ha de ser en minuts- per realitzar la resta de càlculs i acabar amb el procés d'alta de l'article. En la figura 26 es pot apreciar el resultat final de l'escandall d'un article.

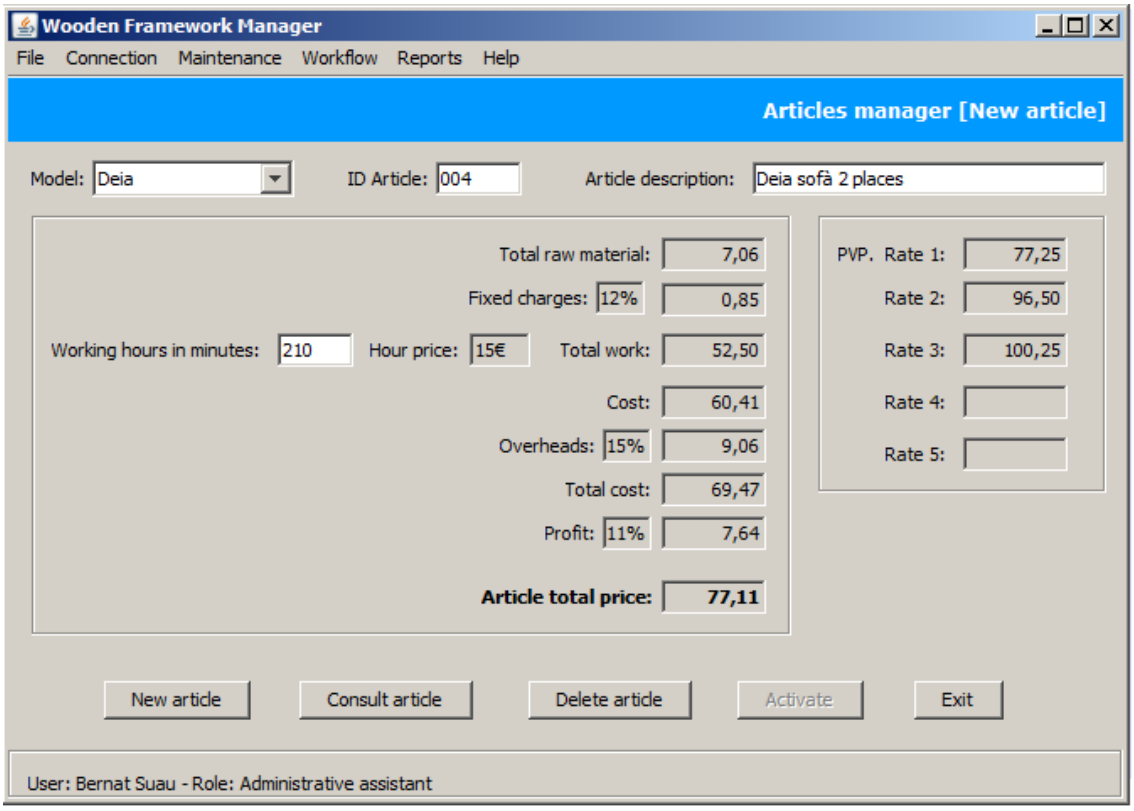


Figura 26. Pantalla article escandallat

3.11 USABILITAT

Fins no fa molt de temps l'enginyeria del programari es centrava en aspectes purament interiors del sistema, com el rendiment o la fiabilitat. A l'hora de desenvolupar una aplicació bàsicament només se seguien els criteris dels analistes i programadors. En l'actualitat, el fet que els productes informàtics vagin dirigits a un públic cada vegada més ampli i menys expert, ha fet canviar aquesta tendència cap a mètodes que tinguin en compte les necessitats i opinions de l'usuari final. D'aquesta forma en el desenvolupament de programari sorgeix el concepte de disseny centrat en l'usuari i l'usabilitat.

L'Organització Internacional per a l'Estandardització[10] (ISO) defineix la usabilitat de la següent forma:

- **ISO/IEC 9126.** *"La usabilitat es refereix a la capacitat d'un programari de ser compres, aprés, usat i ser atractiu per l'usuari, en condicions específiques d'ús".*

- **ISO/IEC 9241.** *"Usabilitat és l'eficàcia, eficiència i satisfacció amb la que un producte permet aconseguir objectius específics a usuaris específics en un context d'ús específic".*

La primera norma fa referència als atributs interns i externs del producte, els quals contribueixen a la seva funcionalitat i eficiència. La segona es centra en el concepte de qualitat d'ús, es a dir, a com l'usuari realitza tasques específiques en escenaris específics de forma efectiva. Així, la usabilitat depèn no tan sols del producte sinó també de l'usuari.

A partir d'aquests conceptes, diferents experts com Jakob Nielsen, Preece, Bruce Tognazinni, Simpson, Ben Shneiderman o Mandel han definit un conjunt de principis bàsics que tota aplicació -ja sigui d'escriptori o en entorn web- hauria complir per assegurar la qualitat d'usable. Si bé, cadascun d'ells té la seva pròpia visió sobre els principis d'usabilitat, es cert que coincideixen en molts aspectes.

[10] http://www.iso.org/iso/home.html

Seguint les recomanacions d'aquests autors, en aquest projecte s'han aplicat una sèrie de criteris amb l'objectiu d'aconseguir un grau de qualitat d'usabilitat acceptable.

3.11.1 CRITERIS D'USABILITAT DE L'APLICACIÓ

- **Visibilitat de l'estat del sistema**. L'aplicació, en tot moment mantindrà informat a l'usuari de l'estat del sistema. Un exemple de com s'ha adoptat aquest criteri és el canvi en l'aparença dels menús quan l'usuari s'identifica, que passen d'un to gris apagat (desactivat) a un negre més ressaltat (activat). També s'han previst quadres de diàleg que mostren missatges informatius després de realitzar operacions crítiques.

- **Control i llibertat de l'usuari**. La interfície serà dissenyada de tal forma que el control de la interacció amb el sistema la tingui l'usuari, de manera que interactuï directament amb els objectes de la pantalla tal com ho faria en el mon real. Un exemple d'aquest criteri són els botons que apareixen en diferents seccions.

- **Facilitat d'aprenentatge**. El sistema seguirà les línies de disseny actual que adopten la majoria d'aplicacions d'escriptori de manera que li resulti familiar a l'usuari i el temps d'aprenentatge es redueixi.

- **Disseny senzill i clar**. Les interfícies adoptaran una estètica senzilla incloent només la informació necessària i rellevant, evitant elements que puguin distreure a l'usuari.

- **Consistència i estàndards**. Un sistema és consistent si tots els mecanismes emprats són sempre usats de la mateixa forma i no es troben paraules, situacions o accions diferents que signifiquin el mateix. El fet d'utilitzar els estàndards de disseny o els coneixements d'altres aplicacions que l'usuari hagi adquirit afavorirà la consistència del sistema. Així, les accions que pugui realitzar l'usuari s'identificaran amb un text o icona de forma única i no s'utilitzaran en operacions diferents, ni canviar-les de lloc o aspecte.

- **Ajuda i documentació**. El millor sistema és el que no necessita cap tipus de documentació, tot i així l'usuari disposarà del menú *Help* per accedir a un manual en el qual s'explicarà el funcionament de l'aplicació.

- **Flexibilitat i eficiència d'ús**. Dins la mesura de lo possible, el sistema s'hauria de dissenyar de forma que el puguin manejar diferents usuaris, amb la possibilitat d'adaptar la interfície a la seva conveniència. Un exemple de com s'inclourà aquest apartat és usant la norma d'internacionalització *i18N*, la qual permetrà que l'aplicació pugui adoptar diferents idiomes. De moment es té previst l'anglès, el català i castellà però se podrien incorporar altres llenguatges fàcilment gràcies a la forma en que s'implementarà. La tècnica consisteix en separar mitjançant un fitxer, els textos que apareixen en la interfície d'usuari del codi de l'aplicació. D'aquesta forma es pot modificar el fitxer per incloure un nou idioma sense tenir que retocar el codi.

- **Recuperabilitat**. Grau de facilitat en que el sistema permet corregir una acció en la qual s'ha comés un error. Per resoldre aquest punt es dissenyaran quadres de diàleg amb avisos per demanar la confirmació d'una operació crítica, perquè l'usuari pugui cancel·lar una acció premuda per equivocació. Per una altre part, el sistema disposarà d'una opció per realitzar *backups* (còpies de seguretat) de les dades, així en cas de pèrdua d'informació es podrà fer un *recovery* (recuperació de les dades).

- **Temps de resposta**. Es prestarà atenció en implementar l'aplicació de forma que el temps que el sistema necessita per mostrar a l'usuari els seus canvis d'estat o els resultats d'operacions sigui mínim.

- **Prevenció d'errors**. El millor tractament dels errors es anticipant-se amb un bon disseny per minimitzar que ocorrin. Alguns mètodes que s'utilitzaran en l'aplicació és la inclusió de missatges informatius en realitzar operacions no vàlides, ressaltar camps amb colors si es deixa en blanc algun camp obligatori o establir formats en els camps per evitar tipus incorrectes. A més, abans de lliurar qualsevol versió de l'aplicació es realitzarà una fase de *testing* composta de proves unitàries i d'integració.

ASPECTES FINALS DEL TFC

4 ANÀLISI DE COSTOS DEL PROJECTE

A continuació, es detalla un anàlisi dels costos generats en el desenvolupament del projecte que hem tractat al llarg d'aqueta memòria. L'objectiu es obtenir una quantitat aproximada, ja que s'ha de tenir en compte que existeixen fases que encara no s'han executat i que per tant, es tindran que calcular de forma empírica i en funció de l'experiència d'altres projectes.

Per realitzar els càlculs s'han tingut en compte els següents punts:

- Nombre de persones: 2
- Càrrec que ocupen: 1 analista i un programador
- Hores realitzades: 790h repartides entre el personal
- Cost per hora. 16€ l'analista i 14€ el programador.

El cost/hora del personal s'ha extret de la web d'*Infojobs*, la qual compte amb l'eina *Infojobs Trends*[11], que mostra l'evolució salarial dels diferents sectors al llarg del temps. En ella hem trobat que un analista, entenem que amb varis anys d'experiència, té un sou brut d'uns 30.000€ anuals i en canvi un programador ronda els 27.000€. Aquestes dades són per una jornada laboral de 8 hores/dia, de dilluns a divendres.

Així tenim que:

- Un analista que fa 160 hores al mes cobra 2500 € bruts. 2500/160=15,63 €/hora.
- Pel que fa al programador 2250/160=14,06 €/hora.

Amb aquestes dades es poden calcular els costos de les tres primeres fases que s'han descrit en el document -pla de treball, anàlisi i disseny- perquè sabem les hores dedicades. En canvi per la implementació s'utilitzarà el mètode *COCOMO* per ser un dels més coneguts, pel qual s'obté l'esforç i el cost en funció de la grandària del programa estimat en *KLOC* (línies de codi en milers).

[11] http://salarios.infojobs.net/index.cfm

En base a un projecte anterior amb una estructura similar a l'actual, hem fet una previsió de 1200 línies de codi per les funcionalitats tractades en el TFC.

4.1 CÀLCUL DELS COSTOS

Aplicació del model COCOMO bàsic-orgànic (projectes amb <50000 línies)

Esforç (E)= $a \cdot KLOC^b$ → $E=2,4 \cdot 1,2^{1,05}=2,9$ persones/mes

Duració (T)= $c \cdot E^d$ → $T=2,5 \cdot 2,9^{0,38}=3,75$ mesos

Persones recomanades (N)=E/D → N=2,9/3,75=0,77 persones per acabar en el termini T

Així tenim que amb un programador es tardarà uns 3 mesos a jornada completa per fer la codificació de les funcions descrites, el que suposa una dedicació d'unes 480 hores. Per reduir aquest temps tindríem que augmentar el personal.

Costos aproximats del programari Wooden Framework Management

Pla de treball – 1 analista 30h x 16€	480 €
Fase d'anàlisi – 1 analista 75h x 16€	1200 €
Fase de disseny – 1 analista 75h x 16€	1200 €
Fase d'implementació - 1 programador 480h x 14€	6720 €
Fase de proves i instal·lació – 1 programador 80h x 14€	1120 €
Documentació – 50 h x 9€	450 €
Cost total	**11170 €**

Mitjançant aquestes dades es podria realitzar una previsió del nombre de còpies que s'haurien de comercialitzar per amortitzar la inversió dins un termini concret. Posteriorment i en base al resultat, també es calcularia el preu de venda que hauria de tenir l'aplicació.

Per una altre part, en cas que el client no disposi de la infraestructura de maquinari necessària, es tindria que fer un estudi de necessitats i del cost de la instal·lació dels equips adequada a cada situació. Finalment, aquest import s'hauria d'afegir al preu de venda de l'aplicació.

5 EXTENSIONS DEL PROGRAMARI

Com s'ha comentat varies vegades, aquest treball només ha tractat una part del total de les funcionalitats que componen el projecte global. En el moment que el client ens ho indiqui, s'iniciarà de nou tot el procés que hem vist al llarg d'aquesta memòria per a desenvolupar noves funcions que han quedat pendents. En concret, per les pròximes iteracions es tenen previst:

- La gestió de facturació (generació d'albarans, factures)
- La generació d'informes i estadístiques (facturació per client, articles més venuts, etc.).
- I finalment, es té previst incorporar un sistema de notificacions per correu electrònic, per informar de l'estat de les seves comandes.

6 CONCLUSIONS

Al principi d'aquest document es varen descriure una sèrie d'objectius que ara, una vegada finalitzat el projecte, puc dir que s'han vist complerts en la seva totalitat. Des de el meu punt de vista el TFC m'ha ajudat a consolidar i entendre millor molts conceptes que hem vist durant la carrera, alguns d'ells de forma superficial. Per exemple, he pogut aprofundir en els diferents tipus de metodologies existents en l'enginyeria del programari, en la usabilitat, o també, en la gestió i planificació de projectes.

Sobre aquest últim punt, he pogut veure la importància que té realitzar una bona planificació i seguiment de les tasques, establint objectius raonables i terminis assequibles. D'aquesta forma s'aconsegueix mantenir la motivació durant tot el projecte perquè veus que vas superant els diferents propòsits.

Ara tinc molt més clar la rellevància que té la recollida de requisits i entendre correctament el domini del problema en la producció de programari. Si aquesta tasca no s'executa de forma correcta, per exemple, no detectar alguns requisits o descriure'ls incorrectament, suposarà com a mínim l'aparició de retards en els lliuraments, podent acabar en funcionalitats mal implementades si no es detecta l'error a temps i per tant, fer que el projecte sigui un fracàs.

Finalment, a hores d'ara, hi ha una empresa que es dedica al desenvolupament de programari que està avaluant aquest projecte, per valorar si pot ser una bona base com a futura aplicació pels seus clients. En cas d'estar interessats pot suposar entrar dins el seu equip i continuar amb la tasca iniciada. Crec que aquest fet és un motiu suficient per concloure que el TFC m'ha resultat una experiència molt positiva.

7 GLOSSARI

Actor. Conjunt de papers d'una entitat exterior en relació amb el sistema considerat.

Arquitectura client/servidor. Sistema distribuït en que una part (servidor) gestiona i proveeix un recurs que una altre (client) pot sol·licitar en un moment donat.

BD (Base de Dades). Conjunt estructurat de dades que representa entitats i interrelacions.

Cas d'ús. Documentació d'una interacció entre el programari i un o més actors.

CASE (Computer-Aided Software Engineering). Programari de suport al desenvolupament, manteniment i documentació informatitzats de programari.

Classe. Forma de representar un conjunt d'objectes que tenen els mateixos atributs i operacions.

Entitat. Objecte del món real que podem distingir de la resta d'objectes i del qual ens interessen algunes propietats.

Escandall. Mètode per calcular els costos de producció d'un article.

Framework. Conjunt de classes que constitueix una aplicació genèrica i incompleta que cal complementar amb classes d'usuari.

GUI (Graphic User Interface). Interfície d'usuari, allò que veuen els usuaris del funcionament del programari.

IDE (Integrated Development Environment). Entorn de programació que permet codificar, compilar i depurar una aplicació.

ISO (International Organization for Standardization). Organisme encarregat de promoure el desenvolupament de normes internacionals de fabricació, comerç i comunicació.

Java. Llenguatge de programació que es pot emprar en qualsevol tipus de plataforma.

MVC (Model-View-Controller). Patró d'arquitectura de programari que separa les dades de l'aplicació, la interfície d'usuari i la lògica de negoci.

OO. Orientat a Objectes. Paradigma de programació.

Patró. Idea de disseny provada extensament i documentada.

RMI (Remote Method Invocation). Part de Java que suporta la invocació d'operacions entre objectes distribuïts.

Socket. Concepte abstracta que permet la comunicació entre un programa del client i un del servidor.

UML (Unified Modeling Language). Model estàndard per a la construcció de programari OO.

Usabilitat. En informàtica, concepte que mesura la facilitat en que les persones poden emprar una aplicació.

WFM (Wooden Framework Management). Nom de l'aplicació a desenvolupar en el TFC.

8 BIBLIOGRAFIA

CAMPDERRICH FALQUERAS, BENET. (2004). *Enginyeria del programari*. Barcelona: Universitat Oberta de Catalunya

PRESSMAN, ROGER S. (2002). *Ingeniería del software: Un enfoque práctico*. McGraw-Hill

CABALLE, Santi i XHAFA, Fatos. (2008). *Aplicaciones distribuidas en Java con tecnología RMI*. Delta Publicaciones Universitarias

A. WEITZENFELD. (2005) *Ingeniería de software orientada a objetos con UML, Java e Internet.* Paraninfo Thomson Learning

I. JACOBSON, G. BOOCK, J.RUMBAUGH. (2000) *El proceso unificado de desarrollo de software*. Pearson Educación, S.A.

CRAIG, Larman. (2003). *UML y patrones. Una introducción al anàlisis y diseño orientado a objetos y al proceso unificado*. Pearson educación, S.A.

NORMAN, D.A. i DRAPER, S.W. (1986). *User centered system design: New perspectives on human-computer interaction*. Hillsdale, NJ: Erlbaum Associates

SISTAC PLANAS, J. (2005) *Bases de dades I*. Material de l'Universitat Oberta de Catalunya. Eureca Media, S.L.

9 ALTRES FONTS DE CONSULTA

http://es.wikipedia.org/wiki/Software#Modelo_cascada
http://www.iso.org/iso/home.html
http://www.useit.com/ (Jakob Nielsen's Website)
http://salarios.infojobs.net/index.cfm