Generació automàtica de l'esquema de bases de dades relacional d'una aplicació a partir del seu diagrama de classes representat en XML i creació automàtica d'interfícies de control d'intercanvi d'informació

Autor: Alex Sabata i Pardell
Enginyeria en informàtica

Consultor: Àlex Alfonso Minguillón

18 de juny de 2004

Per a la Sílvia.
Gràcies pel suport que m'has donat
durant tots aquests anys...

Resum del projecte

Avui en dia, les empreses, administracions, etc. disposen d'un gran nombre d'aplicacions especialitzades, que requereixen ser integrades les unes amb les altres. Així mateix, les noves exigències en les relacions B2B i B2C, multipliquen els intercanvis d'informació entre les empreses, administracions, clients, etc. Es fa difícil per tant, mantenir en aquest escenari un model d'intercanvi d'informació que no estandarditzi el format de la informació a intercanviar (paper, e-mail, fitxers plans, etc.), i al seu torn, aportí un valor afegit en la interpretació, tractament i transformació d'aquesta informació, mitjançant l'ús d'eines estandarditzades.

La solució a aquest problema, passa per l'ús de XML com a format d'intercanvi d'informació entre les diferents entitats, i complementar aquest, amb la implementació d'interfícies que aportin un valor afegit en el tractament i processament d'aquesta informació. Així, XML permet definir i presentar de forma simple, qualsevol tipus d'informació complexa, i al seu torn, aporta un conjunt d'eines i mecanismes que garanteixen la validació, interpretació, processament i transformació de forma estandarditzada, cosa que facilita la creació d'interfícies per l'intercanvi d'aquesta informació expressada en XML.

En aquest projecte, i per tal d'analitzar a través d'un exemple (base de dades d'un Botiga que opera On-Line) algunes de les possibilitats que dona XML, s'ha realitzat la construcció d'un sistema que permet la generació de bases de dades relacionals (amb SQL) a partir de diagrames de classes definits en XML. D'aquesta manera, es mostra la flexibilitat que aporta XML per la definició de documents, processament i transformació dels mateixos. Per la consecució d'aquesta tasca, hom s'ha de centrar en la definició del document XML (DTD), i en la definició de la seva transformació (XSL); mentre que la resta (validació dels documents XML i els processos que porten a terme la transformació) estan totalment estandarditzats.

En una segona fase aquest projecte, ha estat dissenyada i implementada una interfície que permet l'intercanvi d'informació entre una base de dades (la Botiga On-Line), i qualsevol altre sistema d'informació exterior, mitjançant l'intercanvi de documents XML vàlids, facilitant operacions d'inserció, reenviament, esborrat i descàrrega. Per que aquesta interfície no sigui un simple traspàs (importació/exportació) d'informació, s'ha dotat la interfície amb un sistema de control de qualitat que garanteix la validés de la informació intercanviada.

Per tal d'aconseguir aquests objectius, s'ha desenvolupat una aplicació modular, amb una arquitectura de nivells, que permet substituir o adaptar qualsevol part de la mateixa, per tal que sigui possible migrar-la a qualsevol altre sistema operatiu o SGBD. Al seu torn, s'han dissenyat els documents XML i els processos de càrrega/descàrrega, de manera que la interfície resulti neutra a la base de dades contra la que treballa, i només sigui necessari adaptar determinades funcions d'avaluació del sistema de qualitat que resten separades del nucli de la interfície, per poder treballar contra una altra base de dades.

Així mateix, s'ha desenvolupat un sistema de control de qualitat, que permet a la interfície avaluar el contingut dels enviaments (documents XML), en base a una

configuració de qualitat administrable. I finalment, s'ha dissenyat una eina Web que permet el tractament dels enviaments que hagin resultat fallits.

Aquesta projecte per tant, afronta el problema de la creació d'interfícies per a l'intercanvi d'informació entre diferents sistemes, unificant el format d'intercanvi d'aquesta informació, amb llenguatges que permeten una representació clara i estandarditzada (mitjançant XML), i al seu torn, construint una interfície fàcilment reutilitzable i flexible, que suporta un sistema de control de qualitat i una eina de tractament d'enviaments fallits, que estalvien molts problemes en el tractament (càrrega/descàrrega) d'aquests enviaments.

Índex

1. Introducció

1.1 Justificació del PFC i context en el qual es desenvolupa: punt de partida i aportació del PFC

Històricament, un signe de distinció i de qualitat entre les diferents empreses, administracions, etc. en el camp de la gestió de la informació, ha estat poder presumir de disposar de grans volums d'informació, emmagatzemada en entorns fortament propietaris i totalment integrats. Així doncs, i per posar algun exemple, un ajuntament podia presumir de disposar de tota la informació sobre el padró d'habitants, gestió tributària, comptabilitat, gestió de recursos humans, etc. com un enorme conjunt de taules relacionades en una única i complexa base de dades, que conformava la gestió integral de la corporació.

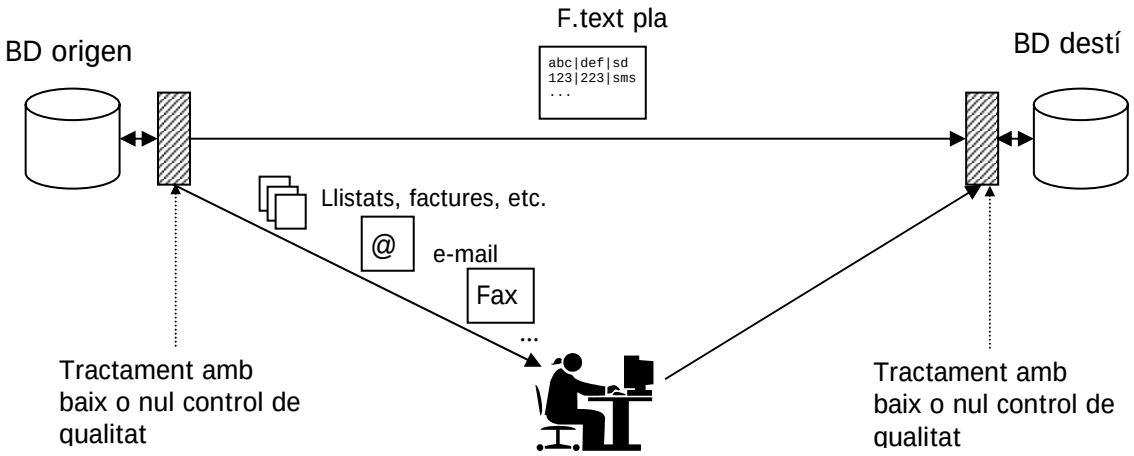
A aquest model, sovint se li atribueixen grans virtuts com per exemple, la nul·la (o mínima) duplicitat de la informació, l'accés automàtic i directe a tota la informació de la corporació, un major control sobre la mateixa, una gran integració de la informació entre diferents departaments, etc. Al llarg del temps però s'ha demostrat que aquest model monolític de gestió de la informació, lluny de ser idíl·lic, no està exempt de problemes. Així doncs en un model integral de gestió de la informació, qualsevol canvi en el model de dades, implica sovint un anàlisi exhaustiu sobre els "efectes secundaris" que produirà el canvi en els conjunt de la base de dades i usos que d'aquesta se'n fa. Així, seguint l'exemple anterior de l'ajuntament, un canvi en el format de l'adreça d'una de les taules relacionades amb la gestió del padró d'habitants, pot implicar canvis en la gestió tributària, etc.; així mateix, aquest tipus de models, provoquen una dependència total en vers l'empresa subministradora del sistema, cosa que moltes vegades fa que es mantinguin en explotació aplicacions poc àgils, pel simple fet que estan integrades en altres que proporcionen un gran rendiment. Continuant amb el mateix exemple, ens podríem trobar que estiguem dotats d'un gran programa de gestió tributària i de padró d'habitants, però que tinguem integrat una deficient gestió comptable, de la qual ens sigui molt difícil prescindir-ne per que les dades siguin mútuament dependents; i en la mateixa línia, podríem citar molts d'altres casos en que queda palès que els sistemes d'informació "monolítics", tot i ser una possible solució, no tenen per que ser-ne la millor, com per exemple, les dificultats de migració a altres sistemes d'altres empreses, dificultats de manteniment, problemes en la incorporació de noves aplicacions de gestió, problemes de rendiment, etc.. És per aquests motius, que sovint trobem empreses que han optat per adquirir aplicacions de gestió no integrats, i després han definit processos d'intercanvi de la informació entre ells.

Sovint també, les diferents empreses, administracions, etc., tinguin o no de sistemes de gestió més o menys integrals, disposen d'altres aplicacions per al processament d'informació, que moltes vegades són altament especialitzades, i per tant, sovint estan poc pensades per donar facilitats a que siguin integrades a altres aplicacions per a cobrir objectius complementaris als propis pels quals van ser concebudes. Alguns exemples d'aquests tipus d'aplicacions són els programes de disseny assistit per ordinador (CAD), programes de simulació de mercats, programes de tractament de text, etc. Continuant amb l'exemple de l'ajuntament, trobarem que el

departament d'arquitectura i serveis tècnics, pot utilitzar el Microstation per fer el disseny i l'edició dels plànols d'un pla parcial de la ciutat, o per fer el disseny d'una oficina; o els integrants o assessors dels grups polítics poden utilitzar eines especialitzades en projecció de vot a partir d'enquestes i altres paràmetres; o els/les administratius/ves poden utilitzar processadors de text que combinin informació continguda en les bases de dades de contribuents per enviar mailings d'informació sobre variacions en el sistema de cobrament d'una taxa, etc.

Finalment, la integració de la informació s'acaba de complicar quan parlem de cobrir les necessitats d'intercanvi d'informació entre les diferents empreses, clients, ciutadans, administracions, etc. els quals poden utilitzar diferents sistemes operatius, SGBD, programes, etc. Efectivament, en aquests casos cadascú disposa dels seus sistemes d'informació de forma exclusiva, i per tant, s'envia i es rep informació en diferents formats contra (o des de) sistemes d'informació completament diferents i totalment desconeguts per la part emissora/receptora. En l'exemple de l'ajuntament, podem trobar que el departament d'estadística ha d'enviar els fitxers mensuals a l'INE (amb un format especificat pel propi INE) amb les modificacions del padró d'habitants; el servei de gestió tributària, envia al banc els fitxers amb la relació dels contribuents amb rebuts domiciliats (segons les especificacions del quadern 19 del consell superior bancari); o el departament de comptabilitat, rep factures en format paper sobre bens o serveis que ha adquirit l'ajuntament, etc.

Amb tot això, les empreses, administracions, etc. no han tingut altre remei que anar buscant possibles solucions per cadascuna de les necessitats d'intercanvi i/o d'integració de la informació que disposen de forma discreta en les diferents aplicacions que es troben disseminades pels diferents departaments; o en les seves relacions amb l'exterior, per poder resoldre els seus vincles amb altres empreses o administracions. Així sovint, les diferents parts que intervenen en l'intercanvi d'informació (ja siguin departaments d'una mateixa empresa, o per les relacions B2B o B2C), han d'arribar a acords per establir quins seran els mecanismes, suports, etc. en que es basarà aquest intercanvi d'informació. Altres vegades, i degut al gran nombre d'actors que intervenen en l'intercanvi d'informació, s'estableix com a norma la utilització de mecanismes, suports, etc. que han estat més o menys estandarditzats per algun ens o organisme. Així, aquests mecanismes o suports poden anar des de l'emissió d'una factura (amb un format més o menys estàndard de factura) per correu postal, o l'intercanvi de llistats en format paper entre diferents departaments d'una mateixa empresa; a l'intercanvi de fitxers plans de text tramesos per algun sistema de telecomunicacions considerat segur, etc.



En molts d'aquests casos, la utilització de mecanismes i suports més o menys estandarditzats entre les parts, no evita haver de processar diverses vegades una mateixa informació, massa vegades de forma manual, i en la major part dels casos, sense haver-hi establerts controls de qualitat sobre la informació processada. Tot plegat implica unes despeses molt elevades en operaris, genera un índex d'errors en el processament de la informació molt gran i provoca unes pèrdues de temps i de diners extraordinàries.

Des d'un temps ençà, aquesta situació s'ha vist agreujada pel creixen ús de les tecnologies de la informació i la comunicació. En els darrers anys, les empreses, administracions, etc. han obert nous horitzons en el tractament i l'accés a la informació, i han creat nous paradigmes en les relacions B2B i B2C. Així per exemple, cada cop és més habitual veure com les diferents empreses treuen els seus productes a la venda fora dels canals de distribució habituals, obrint el seu mercat sobre plataformes com la d'internet; o a veure com administracions com la de l'agència tributaria és capaç d'estar al corrent de tots els nostres moviments bancaris, inversions, propietats, etc. provenint totes de fonts d'informació diferents; i així podríem enumerar multitud de casos en els que es fa evident la necessitat de disposar d'una gran integració de la informació, tant internament per compartir dades entre els propis departaments d'una mateixa empresa, com externament per compartir dades amb altres empreses, institucions, administracions o clients. Així doncs, i continuant amb aquests exemples, en el cas de l'empresa que ha decidit vendre els seus productes o serveis per internet necessitarà que el departament de vendes, facturació, comptabilitat, etc., disposin d'informació constantment i totalment actualitzada sobre les promocions que el departament de marketing hagi disposat impulsar, sobre tot degut al caràcter on-line que ha agafat la seva gestió. Així mateix, l'administració de l'agència tributària, requereix establir vincles per l'intercanvi d'informació, amb multitud d'empreses, institucions, administracions, etc., per tal de poder processar de forma àgil tota la informació relativa a una persona física o jurídica, i així poder-la contrastar amb les seves declaracions de la renda i patrimoni (per posar algun cas). Essent aquests dos exemples, dues gotes dins d'un mar d'iniciatives promogudes des del sector privat, públic, etc., les quals tenen com a denominador comú, l'exigència d'una informació permanentment actualitzada, integrada i accessible des de qualsevol mitjà.

En aquest nou escenari, es fa cada cop més difícil per aquestes empreses o administracions continuar mantenint els mètodes i suports clàssics utilitzats fins al moment per a l'intercanvi d'informació, als quals d'alguna manera o d'una altra s'havien anat acostumant. L'extraordinària explosió d'informació a intercanviar que suposen els sistemes de gestió i tractament de la informació en línia (on-line), no dona marge al processament diferit de la informació, i menys si s'escau, al processament d'aquesta informació amb errors.

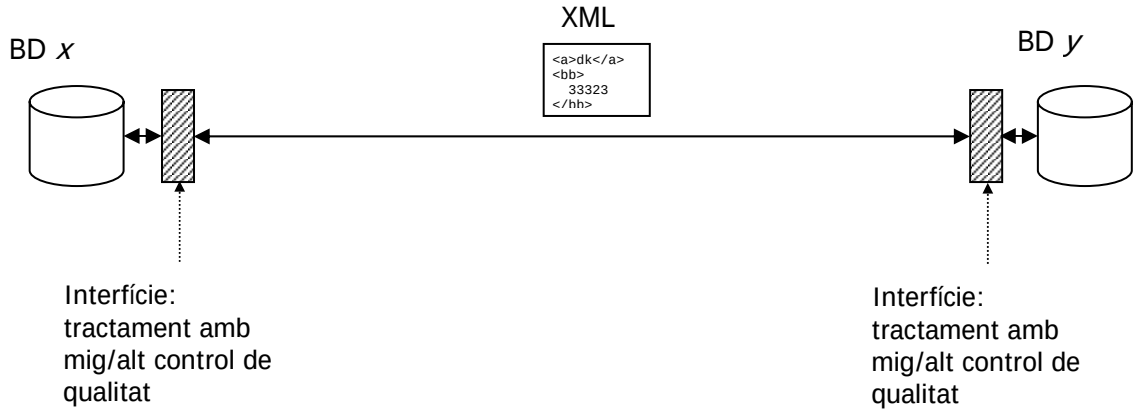
Per aquests motius, es fa palesa la necessitat d'utilitzar nous mecanismes, llenguatges, formats, protocols (en el sentit ampli de la paraula), etc. que permetin intercanvis d'informació de forma que resultin genèricament estandarditzats, i al seu torn, que permetin un processament, validació i control de la informació continguda en els intercanvis, també de forma genèrica i estàndard.

En aquests cas la solució passa per en la utilització de documents XML (eXtensible Markup Language) per a l'intercanvi d'informació, i de forma complementària, per la creació d'interfícies capaces d'interpretar aquests documents, processar-los i validar-

los. Així mateix, cal dotar al tractament de la informació d'un cert control de qualitat abans del seu processament definitiu.

XML és un estàndard industrial per representar dades de forma independent del sistema. Dades que estan descrites en "esquemes" de llenguatge XML, o en definicions de tipus de documents (DTD), els quals descriuen l'estructura d'un conjunt de documents XML i que poden utilitzar-se per limitar-ne els seus continguts. Aquests esquemes o DTD's són els que doten a XML de la seva portabilitat. Al seu torn, existeixen estàndards que giren sobre l'òrbita de XML que permeten obtenir un control sobre la informació continguda en aquests documents, i sobre la representació de les dades, i així permeten convertir els documents XML en altres documents XML o en altres formats.

D'una forma molt resumida, si un departament d'una empresa vol intercanviar informació amb un altre departament de la mateixa empresa o d'una altra, només ha de conèixer l'esquema o el DTD dels documents que aquest admet; i a partir d'aquests, generar documents XML vàlids i enviar-los-hi pel mitjà que hagin establert. Tot seguit, i gràcies al control sobre la informació que el processament de documents XML proveeix, es podrà processar aquests documents XML mitjançant una interfície que podrà garantir la qualitat de les dades enviades, i en definitiva de la transacció.



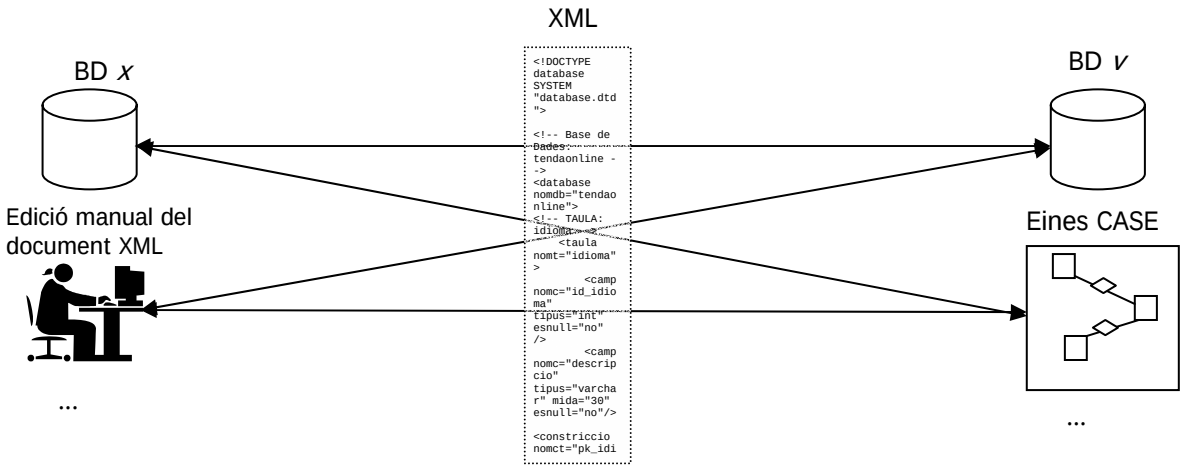
1.2 Objectius generals del PFC

De forma general, l'objectiu en la realització d'aquest projecte és ampliar els coneixements en l'àrea de les bases de dades, introduint-se en el món XML.

En aquest projecte, es pretén analitzar les possibilitats que dona l'estàndard XML per l'intercanvi i integració de dades entre diferents sistemes d'informació, observar les facilitats que presta el llenguatge XML per la representació de la informació, demostrar algunes de les possibilitats que proveeix XML pel processament i transformació de la informació, i finalment, analitzar, dissenyar i construir un sistema d'intercanvi d'informació entre dos sistemes, que es basi en la utilització de documents XML i permeti un control sobre la informació intercanviada.

Per tal de poder demostrar a través d'un exemple concret, la bondat de l'ús de l'estàndard XML en l'entorn en que es situa el projecte, s'estableix com un dels objectius d'aquest, la construcció d'un sistema que permeti la generació d'una base de dades relacional a partir d'un diagrama de classes definit en XML.

En síntesi, es tracta de definir documents XML vàlids que continguin la representació d'una base de dades relacional, i a partir d'aquests, definir una transformació per aquests documents, que acabi generant com a sortida un fitxer amb les sentències SQL que indiquin l'estructura de la base de dades en aquest llenguatge. D'aquesta manera, en els processos d'anàlisi, disseny i definició dels documents XML i de la transformació, es mostra i es justifiquen les atribucions sobre la facilitat que presta l'estàndard XML en la representació de la informació, i al seu torn es demostra amb aquest exemple concret, algunes de les possibilitats que proveeix XML pel processament i transformació d'aquesta.



Un cop presentades les avantatges de treballar amb XML per la representació, processament i transformació de la informació, és necessari fer un pas més per tal d'acabar d'ubicar l'ús XML en l'entorn en que es situa el projecte, proposant una solució per realitzar la validació, processament, control de qualitat i incorporació de la informació continguda en els documents XML, en els sistemes d'informació origen/destí de l'intercanvi d'informació. D'aquesta manera, s'estableix també com a objectiu d'aquest projecte, l'anàlisi, disseny i construcció d'un sistema d'intercanvi d'informació entre la base de dades generada en l'anterior fase del projecte (en aquest cas, una base de dades d'una botiga que ven els seus productes per internet), i qualsevol altre sistema d'informació exterior.

Es tracta per tant, de crear processos automàtics de càrrega i descàrrega de la informació, tenint en compte la definició de la informació a traspassar (basada en documents XML), i actuant segons es tracti d'una inserció d'un nou registre, reenviament o esborrat (en el cas de càrrega d'informació), o una consulta (en el cas d'una descàrrega).

Aquest procés de càrrega i descàrrega d'informació, s'ha de dotar d'un cert control de qualitat, de manera que no sigui un simple traspàs. Així s'identificaran les possibles errades de format i de contingut de les dades contingudes en els documents XML, i es tipificaran segons la seva gravetat. En cap cas no es rebutjarà cap enviament (ja sigui d'inserció, reenviament, esborrament o descarrega/consulta) sigui quin sigui el nivell d'error detectat.

Finalment, i per tal de dotar al sistema d'una major autonomia, es realitzarà el disseny d'una aplicació Web que permeti despatxar per part dels emissors, els enviaments que no han estat incorporats a la base de dades degut a l'existència d'errors detectats durant el control de qualitat. Aquesta aplicació Web, facilitarà la identificació dels enviaments erronis que ha realitzat un emissor concret, i permetrà resoldre aquests errors per tal de poder realitzar el tractament de la informació de forma definitiva a la base de dades.

Concretament, els objectius coberts en la realització d'aquest projecte, són:

- Estudiar de les capacitats de generació d'esquemes de bases de dades a partir de diagrames basats en XML.
- Definir la transformació entre els elements del diagrama i una base de dades relacionals.
- Construcció d'una aplicació capaç d'aplicar la transformació anterior a partir d'un model expressat en XML.
- Realitzar un estudi sobre els diferents sistemes i plantejaments existents en l'actualitat per la creació d'interfícies.
- Construcció d'un sistema d'intercanvi d'informació (interfície) que permeti la càrrega, descàrrega i control de qualitat, entre la base de dades creada anteriorment i qualsevol altre sistema exterior.
- Disseny d'una aplicació Web pel tractament dels enviaments fallits , que permeti la identificació i tractament d'aquests per part del seu agent emissor.

En tot cas, en el disseny i la construcció de les diferents aplicacions, s'ha primat que aquestes siguin portables i independents dels sistemes sobre les quals s'executen; i al seu torn, procurar que aquestes siguin independents de la bases de dades en la que s'ha creat (BD de la Botiga On-Line), i finalment siguin el mínim dependents possible del SGBD contra el que actua.

1.3 Planificació del projecte

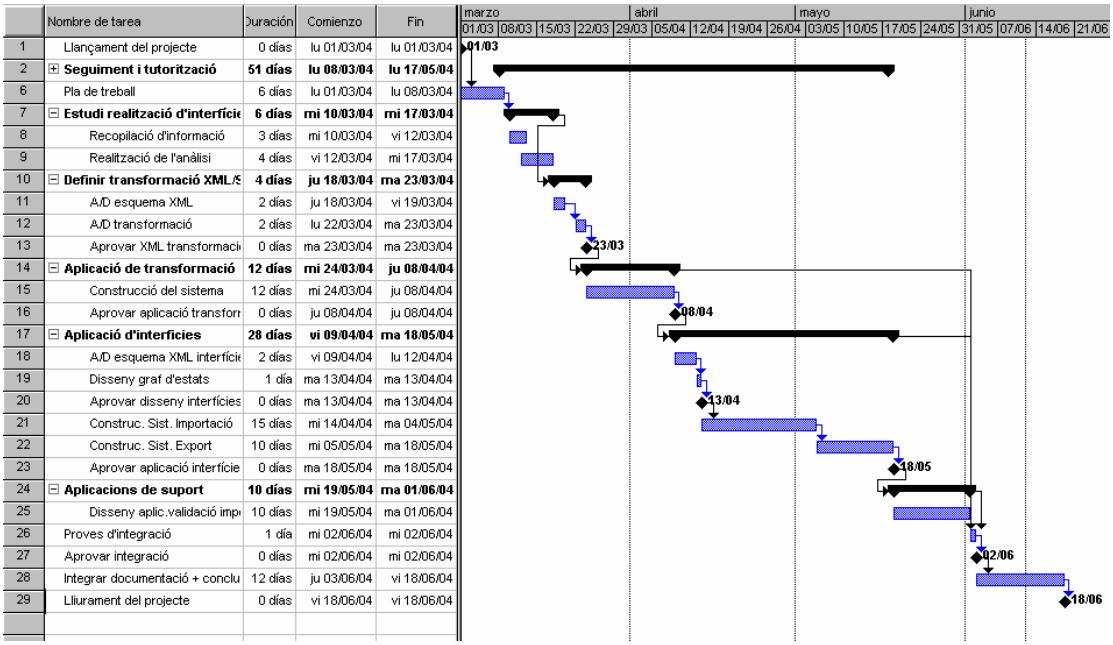
1.3.1 Relació d'activitats

Per tal de definir la planificació del projecte, s'ha considerat la següent descomposició d'activitats:

C.Activitat	Nom de l'activitat de nivell 1	Nom de l'activitat de nivell 2
01	Llançament del projecte	
02	Seguiment i tutorització	
03	Pla de treball	

04	Estudi sobre la realització d'interfícies	
04.01		Recopilació d'informació
04.02		Anàlisi de les alternatives trobades i justificació de l'ús d'XML.
05	Definir els elements per a la transformació dels fitxers XML en fitxer SQL.	
05.01		Analitzar i dissenyar el DTD del document XML.
05.02		Analitzar i dissenyar la trasformació dels elements del diagrama, en llenguatge XML.
06	Aplicació de transformació	
06.01		Construcció del sistema de trasformació
07	Aplicació d'interfícies	
07.01		Anàlisi i disseny de l'esquema XML que definirà les interfícies.
07.02		Disseny del graf d'estats en que es pot trobar un registre pel seu tractament
07.03		Construcció del sistema d'importació
07.04		Proves unitàries
07.05		Construcció del sistema d'exportació
07.06		Proves unitàries
08	Aplicacions de suport	
08.01		Disseny d'una aplicació Web per la indentificació i correcció dels enviaments fallits.
09	Proves d'integració	
10	Integració de la documentació i exposició de conclusions	
11	Lliurament del projecte	

1.3.2 Calendari de treball



	Nombre de tarea	Duración	Comienzo	Fin
1	Llançament del projecte	0 días	lu 01/03/04	lu 01/03/04
2	**Seguiment i tutorització**	**51 días**	**lu 08/03/04**	**lu 17/05/04**
6	Pla de treball	6 días	lu 01/03/04	lu 08/03/04
7	**Estudi realització d'interfície**	**6 días**	**mi 10/03/04**	**mi 17/03/04**
8	Recopilació d'informació	3 días	mi 10/03/04	vi 12/03/04
9	Realització de l'anàlisi	4 días	vi 12/03/04	mi 17/03/04
10	**Definir transformació XML/S**	**4 días**	**ju 18/03/04**	**ma 23/03/04**
11	A/D esquema XML	2 días	ju 18/03/04	vi 19/03/04
12	A/D transformació	2 días	lu 22/03/04	ma 23/03/04
13	Aprovar XML transformaci	0 días	ma 23/03/04	ma 23/03/04
14	**Aplicació de transformació**	**12 días**	**mi 24/03/04**	**ju 08/04/04**
15	Construcció del sistema	12 días	mi 24/03/04	ju 08/04/04
16	Aprovar aplicació transforr	0 días	ju 08/04/04	ju 08/04/04
17	**Aplicació d'interfícies**	**28 días**	**vi 09/04/04**	**ma 18/05/04**
18	A/D esquema XML interfície	2 días	vi 09/04/04	lu 12/04/04
19	Disseny graf d'estats	1 día	ma 13/04/04	ma 13/04/04
20	Aprovar disseny interfícies	0 días	ma 13/04/04	ma 13/04/04
21	Construc. Sist. Importació	15 días	mi 14/04/04	ma 04/05/04
22	Construc. Sist. Export	10 días	mi 05/05/04	ma 18/05/04
23	Aprovar aplicació interfície	0 días	ma 18/05/04	ma 18/05/04
24	**Aplicacions de suport**	**10 días**	**mi 19/05/04**	**ma 01/06/04**
25	Disseny aplic.validació imp	10 días	mi 19/05/04	ma 01/06/04
26	Proves d'integració	1 día	mi 02/06/04	mi 02/06/04
27	Aprovar integració	0 días	mi 02/06/04	mi 02/06/04
28	Integrar documentació + conclu	12 días	ju 03/06/04	vi 18/06/04
29	Lliurament del projecte	0 días	vi 18/06/04	vi 18/06/04

1.4 Productes obtinguts

Aquest projecte s'ha desenvolupat en dos parts:

- En una primera part, s'ha realitzat la construcció d'un sistema que pretén la definició de documents XML vàlids, que contenen la representació d'una base de dades relacional, i a partir d'aquests, definir una transformació per aquests documents, que acabi generant com a sortida un fitxer amb les sentències SQL que indiquin l'estructura de la base de dades en aquests llenguatge.
- Mentre que en la segona, s'ha desenvolupat un sistema d'intercanvi d'informació (interfície) entre la base de dades generada en l'anterior fase de treball i qualsevol sistema d'informació exterior. Tractant que aquest sistema d'intercanvi d'informació, sigui prou "intel·ligent" com per identificar les possibles errades de format i de contingut que es pugui produir durant el processament de la informació a través de la interfície, i al seu torn, dotar al sistema d'eines pel tractament d'aquests enviaments fallits.

De cadascuna de les quals, s'obté:

- De la primera part, s'obté el disseny i construcció d'una definició de tipus de document (DTD), a partir de la qual, es poden crear els documents XML vàlids, que continguin la representació (taules, constriccions, índex, vistes) d'una base de dades relacional.
  Així mateix, s'obté el disseny i la construcció d'un full d'estil XSL per a la transformació dels documents XML vàlids, que contenen la representació d'una base de dades relacional, segons els DTD dissenyat en aquests mateix apartat, i permet generar com a resultat de la transformació un fitxer amb el

contingut de l'estructura de la base de dades (taules, constriccions, índex i vistes) en llenguatge SQL.
Així mateix, s'ha realitzat la construcció de l'estructura d'una base de dades per a una botiga de venda de productes On-Line, mitjançant la creació d'un document XML vàlid segons el DTD creat en aquest apartat. De la transformació d'aquest document XML, se n'obté l'estructura de la base de dades en llenguatge SQL, que s'utilitzarà en la següent part d'aquest projecte.
Finalment, s'ha realitzat una petita aplicació Java que permet la transformació dels documents XML definits, en els seus productes resultants segons la transformació.
En resum, s'obté:

- Un fitxer DTD que servirà de base per la creació dels documents XML, que contindran la representació d'estructures de bases de dades.
- Un full d'estil XSL que facilitarà la transformació dels documents XML vàlids (segons l'anterior DTD), per obtenir estructures de bases de dades en llenguatge SQL.
- Un document XML que conté la representació de la base de dades d'una Botiga On-Line.
- Un fitxer SQL que conté l'estructura de la base de dades de la Botiga On-Line, producte de la transformació, en llenguatge SQL.
- Un petit programa Java que permet la transformació dels documents XML a fitxers SQL, en base a un document XSL.

- Mentre que de la segona part, s'obté el disseny i la construcció d'una interfície que fa possible els processos de càrrega i descàrrega per a l'intercanvi d'informació, entre la base de dades de la Botiga On-Line, i qualsevol altre sistema d'informació exterior (arquitectura, diagrama casos d'ús, diagrama d'estats, diagrama de transicions, format i DTD per a crear fitxers que continguin els documents amb la informació a enviar, pseudocodi, etc.). I es fa de forma que aquesta sigui flexible i reutilitzable per a qualsevol altra base de dades.
  Al seu torn, es dissenya i construeix també el subsistema de control de qualitat, (taules, funcions d'avaluació de qualitat, etc.).
  I finalment, s'obté el disseny d'una eina que ha de permetre el tractament dels enviaments fallits (arquitectura, pseudocodi de les funcions, disseny de les pantalles, etc).
  Així doncs, a part del disseny de tot el sistema, s'obté:
  - Una classe amb el nucli de la interfície.
  - Una classe per la presentació del menú de l'usuari.
  - Una classe que facilita la lectura, processament (amb control de qualitat inclòs) i tractament contra el SGBD dels enviaments.
  - Un fitxer DTD amb l'estructura dels documents XML per a la creació i validació dels enviaments.
  - Una estructura de taules que dona suport als processos de control de qualitat.
  - Un seguit de funcions PLSQL per l'avaluació de la informació a processar.

1.5 Organització del projecte

El projecte s'ha estructurat en dos parts:

- Una primera part (tractada integrament en el capítol "Transformació XML a SQL"), on es mostren algunes de les facilitats que presta el llenguatge XML per la representació, processament i transformació de la informació, a través d'un exemple concret, basat en la construcció d'un sistema que pretén la definició de documents XML vàlids, que contenen la representació d'una base de dades relacional, i a partir d'aquests, definir una transformació, que acabi generant com a sortida un fitxer amb les sentències SQL que indiquin l'estructura de la base de dades en aquests llenguatge.
  Així doncs, en aquesta primera part del projecte, es cobreixen els objectius següents, establerts anteriorment:
  - Estudiar de les capacitats de generació d'esquemes de bases de dades a partir de diagrames basats en XML.
  - Definir la transformació entre els elements del diagrama i una base de dades relacionals.
  - Construcció d'una aplicació capaç d'aplicar la transformació anterior a partir d'un model expressat en XML.

- En la segona part del projecte (tractada en els capítols "Alternatives en la creació d'interfícies", "Interfícies per l'intercanvi d'informació amb la BD de la Botiga On-Line" i "Tractament d'enviaments fallits"), es porta a terme un anàlisi de les possibilitats que dona XML per l'intercanvi i integració de dades entre diferents sistemes d'informació, i es compara aquest amb altres sistemes d'intercanvi d'informació clàssics.
  Així mateix, també es realitza l'anàlisi, disseny i implementació d'una interfície destinada a la càrrega i descàrrega d'informació entre dos sistemes, basat en l'intercanvi de documents XML, i que permet un control de qualitat sobre la informació intercanviada.
  Finalment, es mostra el disseny d'una eina Web destinada a la identificació i tractament els enviaments fallits produïts per un emissor concret.
  Així doncs, en aquesta segona part del projecte, es cobreixen els objectius següents, establerts anteriorment:
  - Realització d'un estudi sobre els diferents sistemes i plantejaments existents en l'actualitat per la creació d'interfícies.
  - Construcció d'un sistema d'intercanvi d'informació (interfície) que permeti la càrrega, descàrrega i control de qualitat, entre la base de dades creada anteriorment i qualsevol altre sistema exterior.
  - Disseny d'una aplicació Web pel tractament dels enviaments fallits, que permeti la identificació i tractament d'aquests per part del seu agent emissor.

Queda per al final les conclusions del projecte, les futures línies de treball. Finalment, en els annexes s'ha disposat tot el codi font desenvolupat, tant pel que fa al codi font de la interfície (desenvolupat en Java), com les funcions per al control de qualitat (desenvolupades en PLSQL).

2. Transformació XML a SQL

XML (eXtensible Markup Language) és un estàndard industrial per representar dades de forma independent del sistema. Al igual que HTML, XML tanca les dades en etiquetes, però hi ha importants diferències entre ambdós llenguatges de marques. Primerament, les etiquetes XML tenen relació amb el significat de la informació que contenen, mentre que les etiquetes HTML especifiquen com mostrar aquesta informació. D'aquesta manera, donat que XML especifica el contingut i l'estructura de les dades que contenen, és possible realitzar cerques, arxivar dades, etc. sobre aquestes.

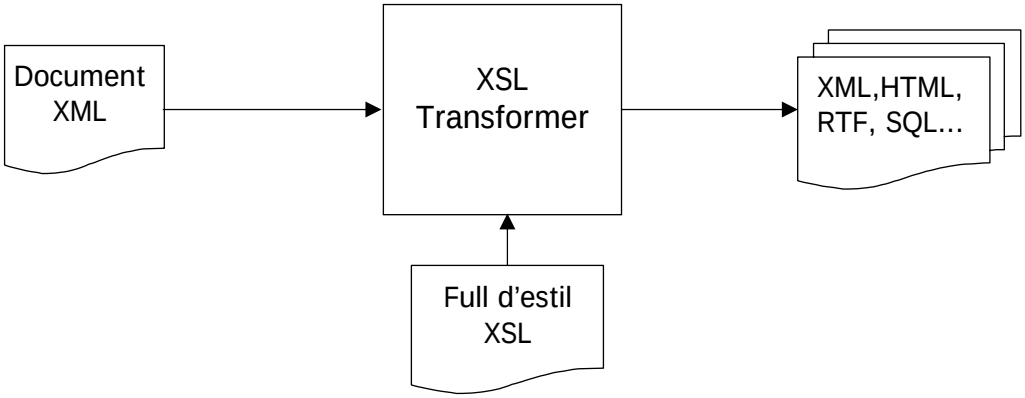
La segona diferència important entre XML i HTML és que les etiquetes XML són extensibles, de manera que ens permeten escriure les nostres pròpies etiquetes XML per tal de descriure el contingut que es desitja representar. D'aquesta manera doncs, amb XML podem crear les etiquetes que necessitem per un tipus de document en particular.

S'acostuma a definir les etiquetes dels documents XML, a partir d'un esquema de llenguatge XML. Un esquema descriu l'estructura d'un conjunt de documents XML i pot utilitzar-se per limitar-ne els continguts. Probablement, el llenguatge per definir esquemes més utilitzat és el DTD (Document Type Definition).

Un DTD (o un esquema en general) garanteix a XML la seva portabilitat, donat que si una aplicació rep un document en format XML i té el seu DTD associat, podrà processar el document d'acord amb les regles especifiques del DTD. Així doncs, donat un DTD es pot conèixer l'estructura del contingut de qualsevol document XML basat en aquell DTD, i al seu torn, es pot avaluar si el document XML és o no vàlid, en funció de si conté elements que no estan inclosos en el DTD, o si segueixen, o no, l'estructura adequada.

Una altra característica sovint preuada de XML, és el fet que està codificat en format de text, de manera que el poden llegir tant els programaris d'edició de text, com els humans.

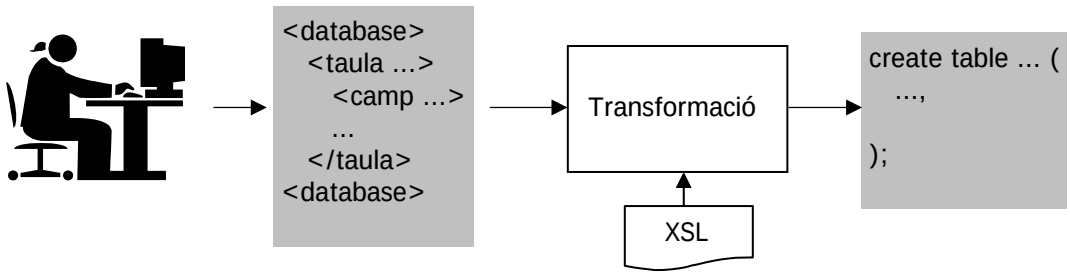
XSLT (XSL Transformations) per la seva banda, fou definit pel grup de treball XSL de la W3C, i descriu un llenguatge per transformar documents XML en altres documents XML o en altres formats. Per realitzar la transformació, normalment necessitem subministrar un full d'estil, que està escrit en XSL (XML Stylesheet Language). El full d'estil XSL especifica com es mostraran les dades XML, mentre que XSLT utilitza les instruccions de formateig del full d'estil per realitzar la transformació.



Atenent a tot això, podem afirmar que XML configura un entorn ideal per la consecució d'un sistema fàcil d'escriure per l'usuari, amb una estructura concreta i amb una representació (transformació) independent, com la que es pretén cobrir en aquesta primera part del projecte.

2.1 Objectius

El què s'exposa en aquesta primera part del projecte, és una implementació per a la construcció d'un sistema que permet la generació de l'estructura d'una base de dades relacional a partir d'una especificació XML. Per tant, l'aplicació a desenvolupar té com a entrada un fitxer XML i ha de generar com a sortida un fitxer amb sentències SQL de creació de la corresponent base de dades relacional, tenint en compte les restriccions que puguin aparèixer com a part del diagrama de la base de dades.



El fet de fer servir XML com a format d'entrada, permet que l'aplicació pugui utilitzar-se per estendre la funcionalitat de qualsevol eina present en el mercat (quasi totes les eines CASE actuals, entre altres aplicacions, permeten generar models en XMI o XML).

Així doncs, els objectius a cobrir són:

- Definir l'estructura del document XML que contindran la representació d'estructures de bases de dades.
- Definir la transformació entre els elements del diagrama i una base de dades relacional.
- Construcció d'una aplicació capaç d'aplicar la transformació anterior a partir d'un model expressat en XML.

En les següents seccions, es mostra amb detall l'especificació de l'estructura que hauran de tenir els documents XML, juntament amb la definició de la transformació que s'aplicarà als documents XML vàlids per tal generar estructures de bases de dades. A continuació, es mostra un exemple de document XML "vàlid" segons el DTD definit, i el resultat de la seva transformació. Finalment, es mostra dues maneres diferents d'aplicar la transformació dels fitxers XML ben formats, en l'estructura d'una base de dades, una d'elles mitjançant l'execució del programa Xml2Sql creat en Java.

2.2 Definició del tipus de document (DTD)

La creació d'una "definició de tipus de document" (DTD), és en essència la creació del nostre propi llenguatge de marques per a una aplicació especifica. Essent així, s'ha creat un DTD que especifica l'esquema que han de complir els fitxers XML per tal de definir estructures de bases de dades. Aquests fitxers XML "vàlids" (que s'ajusten al DTD), entre altres coses, podran ser transformables a estructures SQL mitjançant transformacions XSL.

Així doncs, en primer lloc es mostra una definició de les estructures SQL suportades per aquesta aplicació. Posteriorment, s'indica com s'ha realitzat l'encaix per definir aquestes estructures SQL en un DTD.

2.2.1 Sintaxi SQL suportada

Per a la construcció de l'aplicació de generació d'estructures de bases de dades a partir de documents XML, s'ha pres com a referència la sintaxis amb llenguatge SQL que es mostra a continuació:

Creació de taules:

```
CREATE TABLE <nomt> (
   {camp 1},
   [{camp 2}],
   [{...}],
   [{camp n}],
   [{constriccio 1}],
   [{constriccio 2}],
   [{...}],
   [{constriccio n}]
  );
```

On `{camp i}` és:

```
<nomc> <tipus> [(<mida>)] [DEFAULT <perdefecte>]
       [CHECK (<nomc><condicio>)] [<esnull>]
```

i cadascun dels elements denota:

- `<nomc>`: nom del camp.
- `<tipus>`: tipus de dades del camp, que en aquest cas poden ser: `int, varchar, integer, char, decimal, date`
- `<mida>`: mida del tipus de dades, en cas que s'hagi d'especificar.
- `<perdefecte>`: valor que prendrà el camp en cas de no indicar-ne un altre.
- `<condicio>`: condició que han de complir els valors del camp.
- `<esnull>`: indica si el camp admet valors nuls o no.

`<nomc>` i `<tipus>` són obligatoris mentre que la resta són opcionals.

i on {constricció} és:

```
CONSTRAINT <nomct> <tipus> (<columna 1>[,<columna 2>,...,<columna n>]
         [REFERENCES <nomt>(<campsr 1>[,<campsr 2>, ..., <campsr n>])
```

Essent:

- <nomct>: nom de la constricció. Nom per tant, que tindrà el índex que d'aquesta es generi.
- <tipus>: tipus de constricció, que en aquest cas pot ser: PRIMARY KEY, FOREIGN KEY, UNIQUE
- <columna>: nom dels camps inclosos en la constricció.
- <nomt>: nom de la taula referenciada, en cas que sigui oportú.
- <campsr>: nom dels camps referenciats inclosos en la taula referenciada.

Creació d'índex:

També es permet la definició d'Índex amb la següent estructura:

```
CREATE [UNIQUE] INDEX <nomi> ON <nomt> (<nomci 1> [<ordre>][,<nomci 2>
      <ordre>,] ...[, <nomci n> <ordre>])
```

Essent:

- <nomi>: nom de l'índex.
- <nomt>: nom de la taula sobre la que es crea l'índex.
- <nomci>: nom del/s camp/s indexats.
- <ordre>: tipus d'ordenació dels diferents camps dins de l'índex, que en aquest cas pot ser: ASC, DESC

Creació de vistes:

Finalment, es defineix la següent sintaxi per la creació de vistes:

```
CREATE VIEW <nomv> [(<columnav 1>, ..., <columnav n>)]
   AS
   <Sentència select>
```

La sentència SELECT, s'ha considerat de forma genèrica amb la següent sintaxi:

```
   SELECT [<tipus>] <columnas 1>[, ... <columnas n>]
   FROM <taulaf 1>[,..., <taulaf n>]
   [WHERE <condiciow>
   [GROUP BY <agrupat 1> ... <agrupat n>]
   [HAVING <condicioh>]
   [ORDER BY <ordenatper 1>, ..., <ordenatper n>]
```

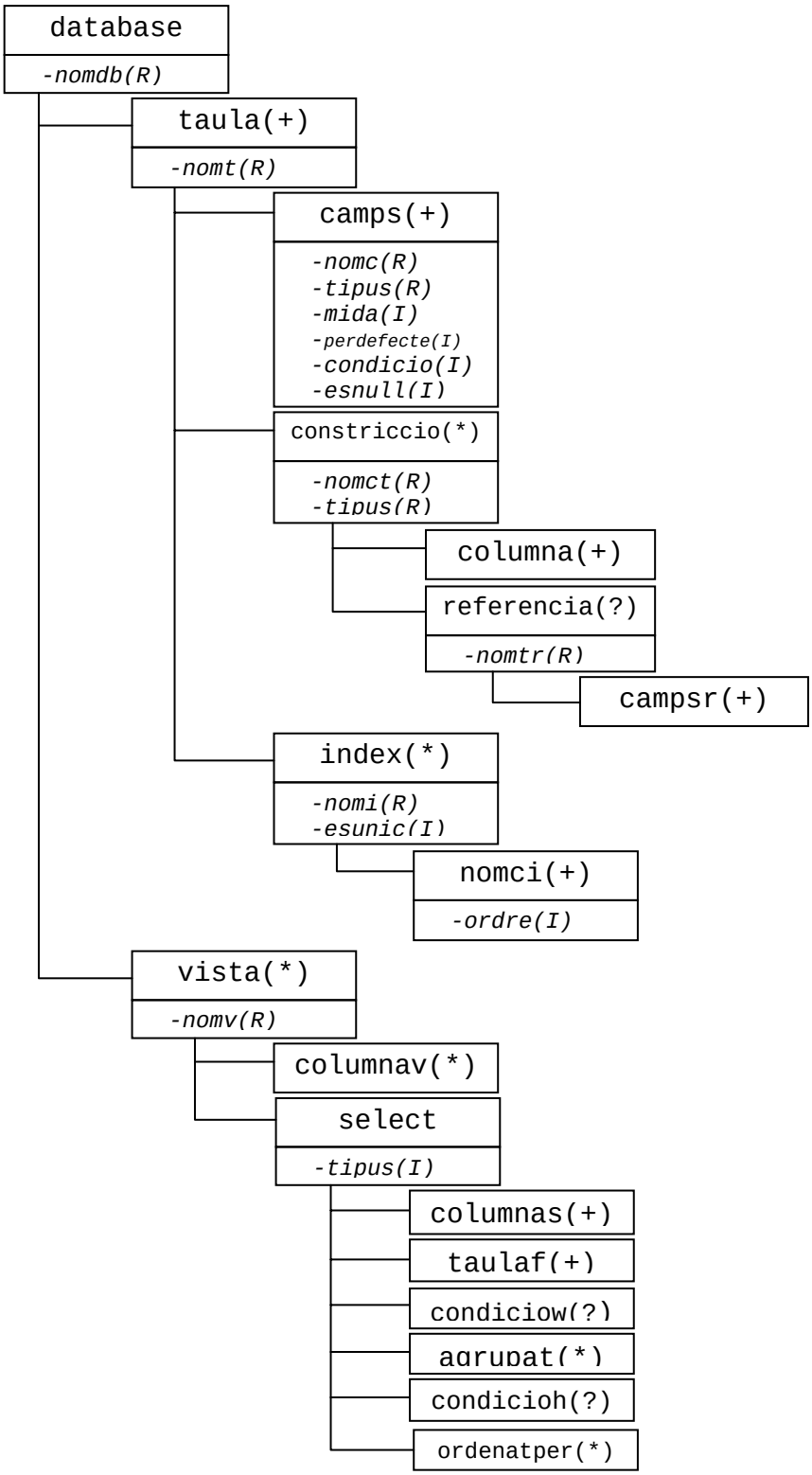
Essent:

- <tipus>: tipus de SELECT, que en aquest cas pot ser: ALL, DISTINCT, UNIQUE
- <columnas>: nom de les columnes de la SELECT, que per aquest cas han de coincidir amb el nom de camps de les taules.
- <taulaf>: taules sobre les quals es realitza la consulta.
- <condiciow>: conjunt de sentències SQL que defineixen les condicions de la consulta.
- <agrupat>: pot ser una o diverses columnes que figuren obligatòriament en la llista de selecció de la clàusula SELECT.
- <condicioh> es refereix a una o diverses condicions enllaçades amb els operadors lògics AND i/o OR. Una condició pot comparar una funció de grup a una altra funció de grup o a una constant.
- <ordenatper>: pot ser una o diverses columnes que figuren obligatòriament en la llista de selecció de la clàusula SELECT.

2.2.2 Definició del DTD

Els documents XML han de seguir una estructura estrictament jeràrquica pel que fa referència a les etiquetes que defineixen els seus elements. És el DTD l'encarregat de definir l'estructura, els tipus d'elements, atributs i entitats permeses, i pot expressar algunes limitacions per a la seva combinació.

Essent així, l'estructura que defineix el DTD, i que per tant, hauran de complir els documents XML "vàlids" que continguin la definició de l'estructura de la base de dades, serà de la següent manera:



On `database`, `taula`, `camps`, `constriccio`, `columna`, `referencia`, `campsr`, `index`, `nomci`, `vista`, `columnav`, `select`, `columnas`, `taulaf`, `condiciow`, `agrupat`, `condicioh` i `ordenatper`, són els elements, i els objectes assenyalats amb cursiva són els *atributs*.

Així mateix, en l'estructura jeràrquica anterior, també es destaquen les modificacions del número d'ocurrències en que poden aparèixer els diferents elements, entre les quals trobem:

- ?: Un cop o cap
- +: Mínim un cop
- *: Qualsevol número de vegades o cap
- (res): Exactament un cop

I finalment, es mostra l'obligatorietat o no, de l'existència d'un atribut en un element:

- *R*: Obligatori
- *I*: Opcional

El DTD per tant, que s'ha dissenyat per definir l'estructura dels documents XML que continguin esquemes de bases de dades vàlids, és de la següent manera:

```
<!ELEMENT database (taula+,vista*)>
<!ATTLIST database nomdb ID #REQUIRED>

<!ELEMENT taula (camp+,constriccio*,index*)>
<!ATTLIST taula nomt ID #REQUIRED>

<!ELEMENT camp EMPTY>
<!ATTLIST camp nomc NMTOKEN  #REQUIRED>
<!ATTLIST camp tipus (int|varchar|integer|char|decimal|date) #REQUIRED>
<!ATTLIST camp mida CDATA #IMPLIED>
<!ATTLIST camp perdefecte CDATA #IMPLIED>
<!ATTLIST camp condicio CDATA #IMPLIED>
<!ATTLIST camp esnull ( si | no ) #IMPLIED>

<!ELEMENT constriccio (columna+, referencia?)>
<!ATTLIST constriccio nomct ID #REQUIRED>
<!ATTLIST constriccio tipus ( PK | FK | UNIQUE ) #REQUIRED>

<!ELEMENT columna (#PCDATA)>
<!ELEMENT referencia (campsr+)>
<!ATTLIST referencia nomtr IDREF #REQUIRED>
<!ELEMENT campsr (#PCDATA)>

<!ELEMENT index (nomci+)>
<!ATTLIST index nomi ID #REQUIRED>
<!ATTLIST index esunic ( si | no ) #IMPLIED>
<!ELEMENT nomci (#PCDATA)>
<!ATTLIST nomci ordre ( ASC | DESC ) #IMPLIED>

<!ELEMENT vista (columnav*,select)>
<!ATTLIST vista nomv ID #REQUIRED>
<!ELEMENT columnav (#PCDATA)>

<!ELEMENT select (columnas+,taulaf+,condiciow?,agrupat*,
                                              condicioh?,ordenatper*)>
<!ATTLIST select tipus ( ALL | DISTINCT | UNIQUE) #IMPLIED>
<!ELEMENT columnas (#PCDATA)>
<!ELEMENT taulaf (#PCDATA)>
<!ELEMENT condiciow (#PCDATA)>
<!ELEMENT agrupat (#PCDATA)>
<!ELEMENT condicioh (#PCDATA)>
<!ELEMENT ordenatper (#PCDATA)>
```

Com es pot comprovar, i tal com s'ha dit anteriorment, aquest DTD defineix l'estructura jeràrquica mostrada anteriorment, indicant els elements, atributs, modificacions de número d'ocurrències dels elements, i l'obligatorietat o no, de l'existència d'un determinat atribut. A més d'aquestes definicions i restriccions, destaquen també les següents:

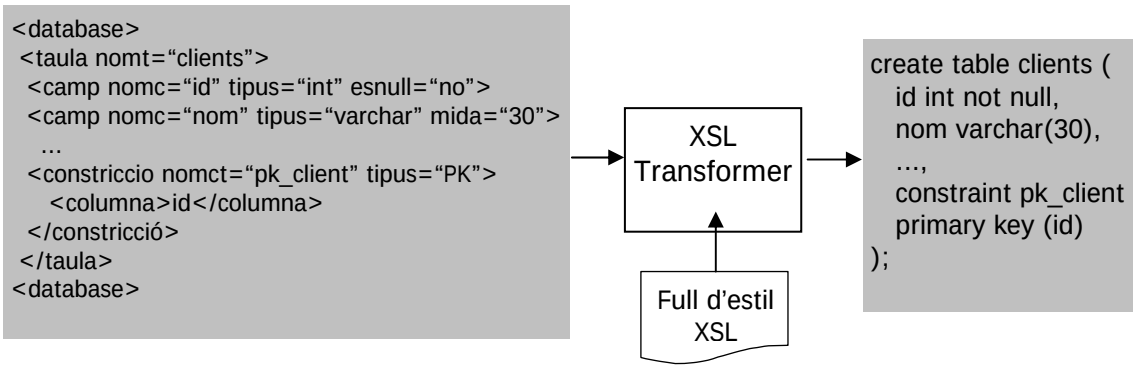
- Identificadors (`ID, IDREF`): El tipus `ID` permet que un atribut determinat tingui un valor únic, que podrà ser referenciat per un atribut d'un altre element que sigui del tipus `IDREF`.
    - En el cas del DTD, ha estat utilitzat el tipus `ID` per a identificar `nomdb` (nom de la base de dades), `nomt` (nom de les taules), `nomct` (nom de les constriccions), `nomi` (nom del índex) i `nomv` (nom de les vistes). D'aquesta manera s'aconsegueix que aquests siguin únics, i que per tant, un document XML ven format, no pugui tenir dos atributs dels indicats en aquesta llista, amb el mateix nom.
    - Així mateix, s'ha utilitzat el tipus `IDREF` per a identificar `nomtr` (nom de la taula referenciada en les constriccions de les taules, quan aquestes referencien alguna altra taula).
- Restricció dels valors que poden agafar alguns dels atributs: Els casos en que s'han restringit els valors que poden agafar els seus corresponents atributs, són els següents:
    - Atribut `tipus` de l'element `camp`, el qual pot agafar els següents valors: `int, varchar, integer, char, decimal, date`
    - Atribut `esnull` de l'element `camp`, el qual pot agafar els següents valors: `si, no`
    - Atribut `tipus` de l'element `constriccio`, el qual pot agafar els valors: `PK, FK, UNIQUE`
    - Atribut `esunic` de l'element `index`, amb els possibles valors: `si, no`
    - Atribut `ordre` de l'element `nomci`, amb els possibles valors: `ASC, DESC`
    - I finalment l'atribut `tipus` de l'element `select`, que pot agafar els valors: `ALL, DISTINCT, UNIQUE`

2.3 Transformació a SQL

Tots els documents XML que compleixin les especificacions del DTD anterior, seran documents "vàlids", i per tant, serviran per indicar l'estructura d'una base de dades. Qualsevol persona o eina pot interpretar el DTD i definir així documents XML "vàlids".

En aquests sentit, una eina que podria presentar-se com a molt útil, seria aquella que interpretés un fitxer XML "vàlid" segons el DTD indicat anteriorment, i fos capaç de transformar-lo a un document SQL (de fet, aquest és l'objectiu d'aquesta primera part del projecte). Per tal de portar a terme la transformació dels documents XML "vàlids" en documents SQL, s'ha utilitzat l'estàndard XSLT.

Com ja s'ha indicat anteriorment, XSLT és un llenguatge utilitzat per a transformar documents XML en altres formats, com per exemple: XML, HTML, DHTML, PDF, RTF, etc., o com és el nostre cas, SQL.



Per realitzar la transformació de XML a SQL, s'ha creat el full d'estil que es mostra a continuació, el qual està escrit en XSL (XML Stylesheet Language):

```
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="2.0"
>

<xsl:strip-space elements="*" />

<xsl:output method="text" />

<xsl:template match="/">
   <xsl:apply-templates />
</xsl:template>

<xsl:template match="database">
  <xsl:text>/*</xsl:text>
  <xsl:value-of select="concat('Estructura de la Base de Dades ',
                                                          @nomdb)" />
  <xsl:text>*/ </xsl:text>
<xsl:text>

</xsl:text>
  <xsl:apply-templates select="taula" />
  <xsl:apply-templates select="vista" />
</xsl:template>

<xsl:template match="taula">
    <xsl:value-of select="concat('CREATE TABLE ', @nomt, '(')" />
<xsl:text>
   </xsl:text>
   <xsl:apply-templates select="camp" />
   <xsl:apply-templates select="constriccio" />
<xsl:text>
);

</xsl:text>
    <xsl:apply-templates select="index" />
</xsl:template>

<xsl:template match="camp">
   <xsl:value-of select="@nomc" />
   <xsl:value-of select="concat(' ',@tipus)" />
   <xsl:if test="@mida">
      <xsl:value-of select="concat('(',@mida,')')" />
   </xsl:if>
```

```
   <xsl:if test="@perdefecte">
      <xsl:text> DEFAULT '</xsl:text>
      <xsl:value-of select="@perdefecte" />
      <xsl:text>'</xsl:text>
   </xsl:if>
   <xsl:choose>
      <xsl:when test="@esnull='no'">
          <xsl:text> not null</xsl:text>
      </xsl:when>
   </xsl:choose>
   <xsl:if test="@condicio">
      <xsl:value-of select="concat(' CHECK (',@nomc,' ',@condicio,
                                                          ')')" />
   </xsl:if>
   <xsl:choose>
     <xsl:when test="not(count(../camp)=position())" >
<xsl:text>,
   </xsl:text>
     </xsl:when>
   </xsl:choose>
</xsl:template>

<xsl:template match="constriccio">
<xsl:text>,
   </xsl:text>
   <xsl:choose>
      <xsl:when test="@tipus='PK'">
          <xsl:value-of select="concat('CONSTRAINT ', @nomct, ' ', '
                                                  PRIMARY KEY (')" />
      </xsl:when>
      <xsl:when test="@tipus='FK'">
         <xsl:value-of select="concat('CONSTRAINT ', @nomct, ' ', '
                                                  FOREIGN KEY (')" />
      </xsl:when>
      <xsl:when test="@tipus='UNIQUE'">
         <xsl:value-of select="concat('CONSTRAINT ', @nomct, ' ', '
                                                       UNIQUE (')" />
      </xsl:when>
   </xsl:choose>
   <xsl:for-each select="columna">
      <xsl:value-of select="." />
      <xsl:choose>
      <xsl:when test="not(count(../columna)=position())" >
          <xsl:text>, </xsl:text>
      </xsl:when>
      </xsl:choose>
   </xsl:for-each>
   <xsl:text>)</xsl:text>
   <xsl:if test="referencia">
      <xsl:value-of select="concat(' REFERENCES ', referencia/@nomtr,
                                                          '(')"/>
      <xsl:for-each select="referencia/campsr">
         <xsl:value-of select="." />
         <xsl:choose>
         <xsl:when test="not(count(../campsr)=position())" >
             <xsl:text>, </xsl:text>
         </xsl:when>
         </xsl:choose>
      </xsl:for-each>
      <xsl:text>)</xsl:text>
   </xsl:if>
```

```
</xsl:template>

<xsl:template match="index">
   <xsl:text>CREATE</xsl:text>
   <xsl:choose>
      <xsl:when test="@esunic='si'">
         <xsl:text> UNIQUE</xsl:text>
      </xsl:when>
   </xsl:choose>
   <xsl:value-of select="concat(' INDEX ', @nomi,' ON ',../@nomt)" />
   <xsl:text> (</xsl:text>
   <xsl:for-each select="nomci">
       <xsl:value-of select="concat(. ,' ', @ordre)" />
       <xsl:choose>
          <xsl:when test="not(count(../nomci)=position())" >
             <xsl:text>, </xsl:text>
          </xsl:when>
       </xsl:choose>
   </xsl:for-each>
   <xsl:text>);
   </xsl:text>
</xsl:template>

<xsl:template match="vista">
   <xsl:value-of select="concat('CREATE VIEW ', @nomv)" />
   <xsl:if test="columnav">
      <xsl:text> (</xsl:text>
      <xsl:for-each select="columnav">
         <xsl:value-of select="." />
         <xsl:choose>
            <xsl:when test="not(count(../columnav)=position())" >
               <xsl:text>, </xsl:text>
            </xsl:when>
         </xsl:choose>
      </xsl:for-each>
      <xsl:text>)</xsl:text>
   </xsl:if>
   <xsl:text> AS
   </xsl:text>
   <xsl:apply-templates select="select" />
</xsl:template>

<xsl:template match="select">
   <xsl:value-of select="concat('SELECT ', @tipus, ' ')" />
   <xsl:for-each select="columnas">
      <xsl:value-of select="." />
      <xsl:choose>
         <xsl:when test="not(count(../columnas)=position())" >
            <xsl:text>, </xsl:text>
         </xsl:when>
      </xsl:choose>
   </xsl:for-each>
   <xsl:text>
   FROM </xsl:text>
   <xsl:for-each select="taulaf">
      <xsl:value-of select="." />
      <xsl:choose>
         <xsl:when test="not(count(../taulaf)=position())" >
            <xsl:text>, </xsl:text>
         </xsl:when>
      </xsl:choose>
```

```
   </xsl:for-each>
   <xsl:text>
   </xsl:text>
   <xsl:if test="condiciow">
      <xsl:value-of select="concat('WHERE ', condiciow)" />
   </xsl:if>
   <xsl:text>
   </xsl:text>
   <xsl:if test="agrupat">
      <xsl:text>GROUP BY </xsl:text>
      <xsl:for-each select="agrupat">
         <xsl:value-of select="." />
         <xsl:choose>
            <xsl:when test="not(count(../agrupat)=position())" >
               <xsl:text>, </xsl:text>
            </xsl:when>
         </xsl:choose>
      </xsl:for-each>
      <xsl:text>
    </xsl:text>
   </xsl:if>
   <xsl:if test="condicioh">
      <xsl:value-of select="concat('HAVING ', condicioh)" />
   </xsl:if>
   <xsl:text>
   </xsl:text>
   <xsl:if test="ordenatper">
      <xsl:text>ORDER BY </xsl:text>
      <xsl:for-each select="ordenatper">
         <xsl:value-of select="." />
         <xsl:choose>
            <xsl:when test="not(count(../ordenatper)=position())" >
               <xsl:text>, </xsl:text>
            </xsl:when>
         </xsl:choose>
      </xsl:for-each>
   </xsl:if>
   <xsl:text>;
   </xsl:text>
</xsl:template>
</xsl:stylesheet>
```

Com es pot observar en el full d'estil XSL que aquí es mostra, s'han definit un seguit de plantilles (regles) que seran aplicades sobre els elements del document font i mitjançant aquestes es configurarà l'arbre resultant, que en aquest cas tindrà l'estructura d'una base de dades en SQL.

XSLT utilitza extensions XPath per consultar elements des de l'arbre font, o per avaluar fragments del document a ser inserits dins de l'arbre resultat.

En aquest cas, s'han definit les plantilles `database`, `taula`, `camp`, `constriccio`, `index`, `vista` i `select`. Quan s'aplica una d'aquestes plantilles, s'insereix en l'arbre resultant un node de text que depenent del cas tindrà una forma o una altra. Quan una plantilla es deixa d'aplicar, l'arbre resultant s'imprimeix utilitzant el mètode de sortida de text.

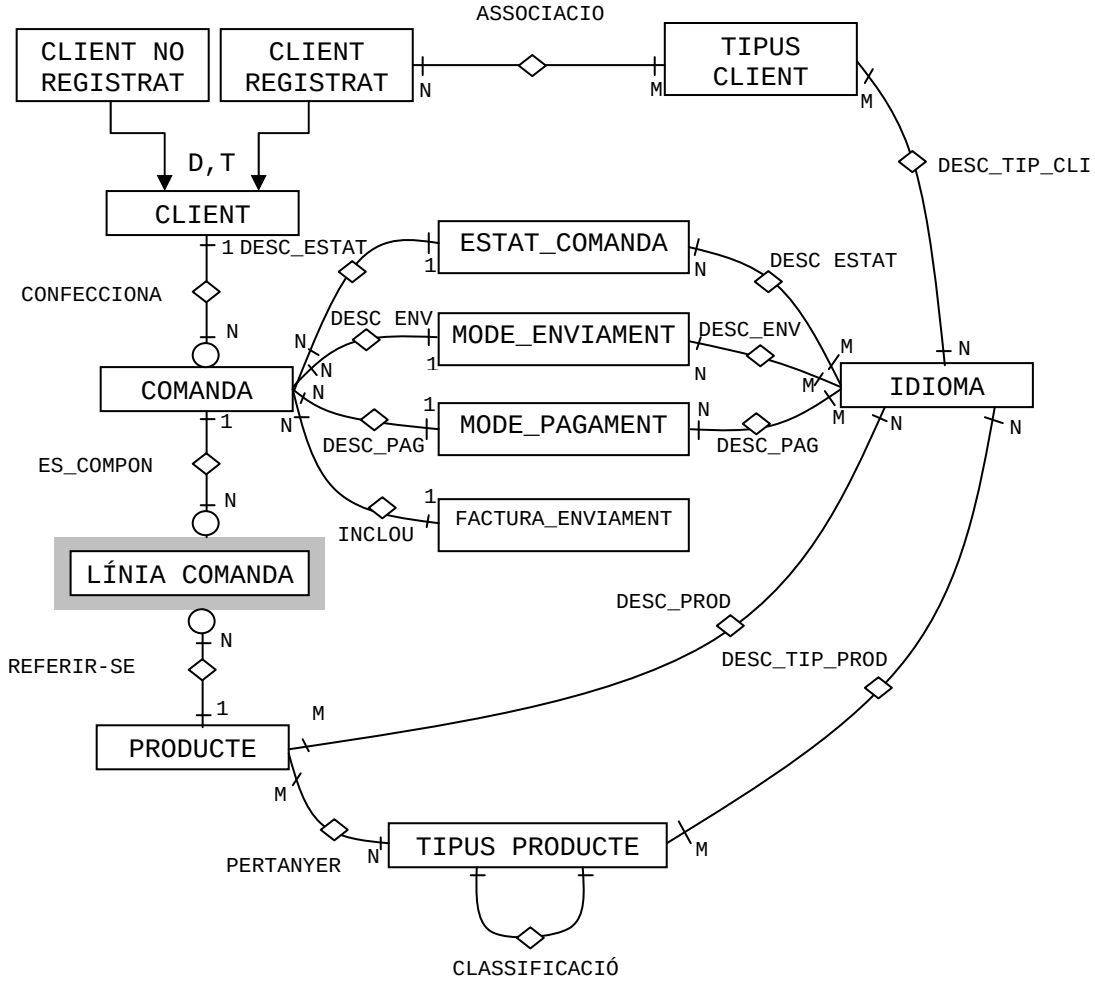
2.4 Exemple de fitxer XML "vàlid" (BD d'una botiga On-Line)

Com a exemple de fitxer XML "vàlid" segons el DTD anteriorment especificat, i per tal de poder mostrar el resultat de la seva transformació, s'ha optat per utilitzar el model conceptual i lògic d'un exemple de base de dades que configura una part d'una botiga que opera on-line a internet[1].

Al seu torn, aquesta base de dades serà la que s'utilitzarà en la segona part d'aquest projecte, on es crearà una interfície per a l'intercanvi d'informació entre aquesta base de dades, i qualsevol altre sistema d'informació exterior. És per aquest motiu, que s'ha decidit fer una explicació formal de l'exemple.

2.4.1 Model conceptual

Per a modelitzar les dades que intervenen en l'escenari de les aplicacions B2C d'una botiga que opera on-line a internet, i de manera general, es considera que els clients, per mitjà de la web, compren productes i que el mecanisme habitual per a encomanar els productes és mitjançant la confecció d'una comanda. El model ER que s'ha considerat per modelitzar aquest escenari, és el següent:



[1] El model conceptual i lògic d'aquest exemple s'ha extret dels apunts de l'assignatura de comerç electrònic de la UOC.

El diagrama ER és la representació d'un model conceptual de dades genèric per l'escenari en què ens trobem, i del qual, les entitats principals identificades són: CLIENT, PRODUCTE, COMANDA i IDIOMA.

- CLIENT: Els clients no registrats compren sense que l'aplicació els reconegui, per tant, malgrat que comprin diverses vegades, es consideren com si fossin clients diferents. Al seu torn, els clients registrats són aquells dels quals la base de dades conté la informació següent: dades personals, dades sobre el seu historial de compres i dades addicionals per a indentificar-lo en cas que oblidi el seu codi secret.
- PRODUCTE: L'entitat producte representa els diferents productes que formen part d'un catàleg de productes i, per tant, que l'empresa ven. Aquesta entitat emmagatzemarà els atributs que descriuen les característiques de cada producte amb independència de l'idioma.
- COMANDA: La comanda és precisament el nexe entre CLIENTS i PRODUCTE, ja que un cop el client ha seleccionat els productes que desitja, els demana formulant la comanda. De la comanda, n'hem d'emmagatzemar moltes dades, per això es presenta com a una entitat en lloc de fer-ho com a interrelació entre CLIENTS i PRODUCTE.
- IDIOMA: A l'entitat idioma s'emmagatzemarà el codi d'idioma amb la seva descripció en un idioma de referència.

2.4.2 Disseny lògic

Per a obtenir el disseny lògic de la base de dades operativa, partirem del model conceptual genèric presentat, i el transformarem perquè s'adapti a la tecnologia que s'hagi d'emprar.

En el nostre cas farem servir les bases de dades relacionals, per tant, obtindrem un conjunt de relacions amb els seus atributs, claus primàries i foranes.

Les relacions obtingudes a partir de la transformació relatives a clients són les següents:

```
CLIENT(id_client)

CLIENT_REGISTRAT(id_client, nif, nom, cognoms, email,
     contrasenya, pregunta, resposta, data_naixement,
     estat_civil, data_primera_compra, data_darrera_compra,
     import_acumulat_compres, nombre_compres, baixa_logica)
on {email} és clau alternativa
on {id_client} referencia CLIENT

CLIENT_NO_REGISTRAT(id_client)
on {id_client} referencia CLIENT

TIPUS_CLIENT(id_tipus_client, dte)
```

```
DESC_TIP_CLI(id_tipus_client, id_idioma,
      descripcio_tipus_client)
on {id_tipus_client} referencia TIPUS_CLIENT i {id_idioma} referencia
IDIOMA

ASSOCIACIO(id_client, id_tipus_client)
on {id_client} referencia CLIENT i {id_tipus_client} referencia
TIPUS_CLIENT
```

Com es pot observar, la relació CLIENT_NO_REGISTRAT no aporta més informació addicional de la que aporta la relació CLIENT i, per tant, a nivell lògic en podem prescindir.

Les relacions obtingudes a partir de la transformació relatives a productes són les següents:

```
PRODUCTE(id_producte, preu_actual, es_oferta, preu_oferta,
      estoc_inicial, estoc_actual, estoc_modificacio)

TIPUS_PRODUCTE(id_tipus_producte, dte)

DESC_TIP_PROD(id_tipus_producte, id_idioma, descripció)
on {id_tipus_prod} referencia TIPUS_PRODUCTE i {id_idioma} referencia
IDIOMA.

CLASSIFICACIO(id_classif, id_tipus_generic,
      id_tipus_especific)
on {id_tipus_generic} referencia TIPUS_PRODUCTE i
{id_tipus_especific} referencia TIPUS_PRODUCTE

DESC_PROD(id_producte, id_idioma, descripcio_curta,
      descripcio_llarga)
on {id_producte} referencia PRODUCTE i {id_idioma} referencia IDIOMA.

PERTANYER(id_producte, id_tipus_producte)
on {id_producte} referencia PRODUCTE i {id_tipus_producte} referencia
TIPUS_PRODUCTE
```

Les relacions obtingudes a partir de la transformació relativa a comanda són les següents:

```
FACTURA_ENVIAMENT(id_fact_env, nif, nom, cognoms, adreca,
      poblacio, codi_postal, pais, telefon, fax, email, nomenv,
      cognomsenv, adrecaenv, poblacioenv, codi_postalenv,
      paisenv)

ESTAT_COMANDA(id_estat)

FORMA_ENVIAMENT(id_enviament)
```

```
FORMA_PAGAMENT(id_pagament)

COMANDA(id_comanda, id_client, id_fact_env, total_comanda,
     data_comanda, hora_inici_compra, hora_fi_compra,
     adre_ip_compra, id_estat, num_transaccio,
     data_transaccio, id_resultat_transaccio, id_pagament,
     id_enviament, data_lliurament, hora_lliurament)
on {id_estat} referencia ESTAT_COMANDA, {id_enviament} referencia
FORMA_ENVIAMENT, {id_pagament} referencia FORMA_PAGAMENT.

LINIA_COMANDA(id_linia, id_comanda, id_producte, descripcio,
     unitats, preu_unitari_but, dte, preu_net)
on {id_producte} referencia PRODUCTE, {id_comanda} referencia COMANDA.

DESC_ENV(id_enviament, id_idioma, descripcio)
on {id_idioma} referencia IDIOMA

DESC_ESTAT(id_estat, id_idioma, descripcio)
on {id_idioma} referencia IDIOMA

DESC_PAG(id_pagament, id_idioma, descripcio)
on {id_idioma} referencia IDIOMA
```

Finalment, la relació obtinguda a partir de la transformació relativa a idioma és la següent:

```
IDIOMA(id_idioma, descripció)
```

2.4.3 Estructura de la BD de la “botiga on-line” en XML

El contingut del fitxer XML “vàlid” segons el DTD definit al principi, que descriu la base de dades per a la botiga on-line especificada en els apartats anteriors, és el següent:

```
<!DOCTYPE database SYSTEM "database.dtd">

<!-- Base de Dades: tendaonline -->
<database nomdb="tendaonline">

<!-- TAULA: idioma -->
   <taula nomt="idioma">
      <camp nomc="id_idioma" tipus="int" esnull="no" />
      <camp nomc="descripcio" tipus="varchar" mida="30" esnull="no"/>
      <constriccio nomct="pk_idioma" tipus="PK">
         <columna>id_idioma</columna>
       </constriccio>
   </taula>
```

```
<!-- TAULA: client -->
   <taula nomt="client">
      <camp nomc="id_client" tipus="int" esnull="no" />
      <constriccio nomct="pk_client" tipus="PK">
         <columna>id_client</columna>
      </constriccio>
   </taula>

<!-- TAULA: tipus_client -->
   <taula nomt="tipus_client">
      <camp nomc="id_tipus_client" tipus="int" esnull="no" />
      <camp nomc="dte" tipus="decimal" mida="4,2" esnull="no"
                                                  perdefecte="0" />
      <constriccio nomct="pk_tipus_client" tipus="PK">
         <columna>id_tipus_client</columna>
      </constriccio>
   </taula>

<!-- TAULA: producte -->
   <taula nomt="producte">
      <camp nomc="id_producte" tipus="int" esnull="no" />
      <camp nomc="preu_actual" tipus="decimal" mida="10,2"
                                    esnull="no" condicio=">0" />
      <camp nomc="es_oferta" tipus="varchar" mida="1" esnull="no"
                          condicio="IN ('N','S')" perdefecte="N" />
      <camp nomc="preu_oferta" tipus="decimal" mida="10,2"
                                    esnull="si" condicio=">0" />
      <camp nomc="estoc_inicial" tipus="decimal" mida="12,2"
                                   esnull="no" perdefecte="0" />
      <camp nomc="estoc_actual" tipus="decimal" mida="12,2"
                                   esnull="no" perdefecte="0" />
      <camp nomc="estoc_notificacio" tipus="decimal" mida="12,2"
                                   esnull="no" perdefecte="0" />
      <constriccio nomct="pk_producte" tipus="PK">
         <columna>id_producte</columna>
      </constriccio>
   </taula>

<!-- TAULA: tipus_producte -->
   <taula nomt="tipus_producte">
      <camp nomc="id_tipus_producte" tipus="int" esnull="no" />
      <camp nomc="dte" tipus="decimal" mida="4,2" esnull="no"
                                                  perdefecte="0" />
      <constriccio nomct="pk_tipus_producte" tipus="PK">
         <columna>id_tipus_producte</columna>
      </constriccio>
   </taula>

<!-- TAULA: forma_enviament -->
   <taula nomt="forma_enviament">
      <camp nomc="id_enviament" tipus="int" esnull="no" />
      <constriccio nomct="pk_forma_enviament" tipus="PK">
         <columna>id_enviament</columna>
      </constriccio>
   </taula>

<!-- TAULA: forma_pagament -->
   <taula nomt="forma_pagament">
      <camp nomc="id_pagament" tipus="int" esnull="no" />
      <constriccio nomct="pk_forma_pagament" tipus="PK">
         <columna>id_pagament</columna>
```

```
      </constriccio>
   </taula>

<!-- TAULA: associacio -->
   <taula nomt="associacio">
      <camp nomc="id_client" tipus="int" esnull="no" />
      <camp nomc="id_tipus_client" tipus="int" esnull="no"/>
      <constriccio nomct="pk_associacio" tipus="PK">
         <columna>id_client</columna>
         <columna>id_tipus_client</columna>
      </constriccio>
      <constriccio nomct="fk_associacio_idclient" tipus="FK">
         <columna>id_client</columna>
          <referencia nomtr="client">
             <campsr>id_client</campsr>
          </referencia>
      </constriccio>
      <constriccio nomct="fk_associacio_tipusclient" tipus="FK">
         <columna>id_tipus_client</columna>
          <referencia nomtr="tipus_client">
             <campsr>id_tipus_client</campsr>
          </referencia>
      </constriccio>
   </taula>

<!-- TAULA: classificacio -->
   <taula nomt="classificacio">
      <camp nomc="id_classif" tipus="int" esnull="no" />
      <camp nomc="id_tipus_generic" tipus="int" esnull="no"/>
      <camp nomc="id_tipus_especific" tipus="int" esnull="no"/>
      <constriccio nomct="pk_classificacio" tipus="PK">
         <columna>id_classif</columna>
      </constriccio>
      <constriccio nomct="fk_classificacio_tgeneric" tipus="FK">
         <columna>id_tipus_generic</columna>
          <referencia nomtr="tipus_producte">
             <campsr>id_tipus_producte</campsr>
          </referencia>
      </constriccio>
      <constriccio nomct="fk_classificacio_tespec" tipus="FK">
         <columna>id_tipus_especific</columna>
          <referencia nomtr="tipus_producte">
            <campsr>id_tipus_producte</campsr>
          </referencia>
      </constriccio>
      <index nomi="idx_tipus_class" esunic="si">
         <nomci ordre="ASC">id_tipus_generic</nomci>
         <nomci ordre="DESC">id_tipus_especific</nomci>
      </index>
   </taula>

<!-- TAULA: client_registrat -->
   <taula nomt="client_registrat">
     <camp nomc="id_client" tipus="int" esnull="no" />
     <camp nomc="nif" tipus="varchar" mida="9" esnull="no"/>
     <camp nomc="nom" tipus="varchar" mida="15" esnull="no"/>
     <camp nomc="cognoms" tipus="varchar" mida="30" esnull="no"/>
     <camp nomc="email" tipus="varchar" mida="30" esnull="no"/>
     <camp nomc="contrasenya" tipus="varchar" mida="10" esnull="no"/>
     <camp nomc="pregunta" tipus="varchar" mida="100"/>
     <camp nomc="resposta" tipus="varchar" mida="50"/>
```

```
     <camp nomc="data_naixement" tipus="date" esnull="no"/>
     <camp nomc="estat_civil" tipus="varchar" mida="1" esnull="no"
                           condicio="in ('C','S')" perdefecte="S" />
     <camp nomc="data_primera_compra" tipus="date" />
     <camp nomc="data_darrera_compra" tipus="date" />
     <camp nomc="import_acumulat_compres" tipus="decimal" mida="12,2"
                                       esnull="no" perdefecte="0" />
     <camp nomc="nombre_compres" tipus="int" esnull="no"
                                                perdefecte="0" />
     <camp nomc="baixa_logica" tipus="varchar" mida="1" esnull="no"
                            condicio="in ('N','S')" perdefecte="N"/>
     <constriccio nomct="pk_client_registrat" tipus="PK">
        <columna>id_client</columna>
     </constriccio>
     <constriccio nomct="u_client_registrat" tipus="UNIQUE">
        <columna>email</columna>
     </constriccio>
     <constriccio nomct="fk_client_registrat" tipus="FK">
        <columna>id_client</columna>
        <referencia nomtr="client">
          <campsr>id_client</campsr>
        </referencia>
     </constriccio>
     <index nomi="idx_nif_client_r" esunic="si">
        <nomci ordre="ASC">nif</nomci>
     </index>
   </taula>

<!-- TAULA: estat_comanda -->
   <taula nomt="estat_comanda">
     <camp nomc="id_estat" tipus="int" esnull="no" />
     <constriccio nomct="pk_estat_comanda" tipus="PK">
       <columna>id_estat</columna>
     </constriccio>
   </taula>

<!-- TAULA: comanda -->
   <taula nomt="comanda">
     <camp nomc="id_comanda" tipus="varchar" mida="10" esnull="no" />
     <camp nomc="id_client" tipus="int" esnull="no"/>
     <camp nomc="id_fact_env" tipus="varchar" mida="10" esnull="no"/>
     <camp nomc="total_comanda" tipus="decimal" mida="11,2"
                                        esnull="no" condicio=">0" />
     <camp nomc="data_comanda" tipus="date" esnull="no"/>
     <camp nomc="hora_inici_comanda" tipus="date" esnull="no"/>
     <camp nomc="hora_fi_comanda" tipus="date" esnull="no"/>
     <camp nomc="adre_ip_comanda" tipus="varchar" mida="16"
                                                   esnull="si"/>
     <camp nomc="id_estat" tipus="int" esnull="no"/>
     <camp nomc="num_transaccio" tipus="int" esnull="no"/>
     <camp nomc="data_transaccio" tipus="date" esnull="no"/>
     <camp nomc="id_resultat_transaccio" tipus="int" esnull="no"/>
     <camp nomc="id_pagament" tipus="int" esnull="no"/>
     <camp nomc="id_enviament" tipus="int" esnull="no"/>
     <camp nomc="data_lliurament" tipus="date" esnull="no"/>
     <camp nomc="hora_lliurament" tipus="date" />
     <constriccio nomct="pk_comanda" tipus="PK">
        <columna>id_comanda</columna>
     </constriccio>
     <constriccio nomct="fk_comanda_estat" tipus="FK">
        <columna>id_estat</columna>
```

```
        <referencia nomtr="estat_comanda">
           <campsr>id_estat</campsr>
        </referencia>
     </constriccio>
     <constriccio nomct="fk_comanda_enviament" tipus="FK">
        <columna>id_enviament</columna>
        <referencia nomtr="forma_enviament">
           <campsr>id_enviament</campsr>
        </referencia>
     </constriccio>
     <constriccio nomct="fk_comanda_pagament" tipus="FK">
        <columna>id_pagament</columna>
        <referencia nomtr="forma_pagament">
          <campsr>id_pagament</campsr>
        </referencia>
     </constriccio>
   </taula>

<!-- TAULA: desc_tip_client -->
   <taula nomt="desc_tip_client">
     <camp nomc="id_tipus_client" tipus="int" esnull="no" />
     <camp nomc="id_idioma" tipus="int" esnull="no"/>
     <camp nomc="descripcio_tipus_client" tipus="varchar" mida="50"
                                                    esnull="no" />
     <constriccio nomct="pk_desc_tip_client" tipus="PK">
       <columna>id_tipus_client</columna>
       <columna>id_idioma</columna>
     </constriccio>
     <constriccio nomct="fk_desc_tip_client_id" tipus="FK">
       <columna>id_tipus_client</columna>
       <referencia nomtr="tipus_client">
         <campsr>id_tipus_client</campsr>
       </referencia>
     </constriccio>
     <constriccio nomct="fk_desc_tip_client_idioma" tipus="FK">
       <columna>id_idioma</columna>
       <referencia nomtr="idioma">
         <campsr>id_idioma</campsr>
       </referencia>
     </constriccio>
   </taula>

<!-- TAULA: desc_env -->
   <taula nomt="desc_env">
     <camp nomc="id_enviament" tipus="int" esnull="no" />
     <camp nomc="id_idioma" tipus="int" esnull="no"/>
     <camp nomc="descripcio" tipus="varchar" mida="50" esnull="no"/>
     <constriccio nomct="pk_desc_env" tipus="PK">
       <columna>id_enviament</columna>
       <columna>id_idioma</columna>
     </constriccio>
     <constriccio nomct="fk_desc_env_idioma" tipus="FK">
       <columna>id_idioma</columna>
       <referencia nomtr="idioma">
         <campsr>id_idioma</campsr>
       </referencia>
     </constriccio>
   </taula>

<!-- TAULA: desc_estat -->
   <taula nomt="desc_estat">
```

```
    <camp nomc="id_estat" tipus="int" esnull="no" />
    <camp nomc="id_idioma" tipus="int" esnull="no"/>
    <camp nomc="descripcio" tipus="varchar" mida="50" esnull="no"/>
    <constriccio nomct="pk_desc_estat" tipus="PK">
      <columna>id_estat</columna>
      <columna>id_idioma</columna>
    </constriccio>
    <constriccio nomct="fk_desc_estat_idioma" tipus="FK">
      <columna>id_idioma</columna>
      <referencia nomtr="idioma">
        <campsr>id_idioma</campsr>
      </referencia>
    </constriccio>
  </taula>

<!-- TAULA: desc_pag -->
  <taula nomt="desc_pag">
    <camp nomc="id_pagament" tipus="int" esnull="no" />
    <camp nomc="id_idioma" tipus="int" esnull="no"/>
    <camp nomc="descripcio" tipus="varchar" mida="50" esnull="no"/>
    <constriccio nomct="pk_desc_pag" tipus="PK">
       <columna>id_pagament</columna>
       <columna>id_idioma</columna>
    </constriccio>
    <constriccio nomct="fk_desc_pag_idioma" tipus="FK">
       <columna>id_idioma</columna>
       <referencia nomtr="idioma">
         <campsr>id_idioma</campsr>
       </referencia>
    </constriccio>
  </taula>

<!-- TAULA: desc_prod -->
  <taula nomt="desc_prod">
    <camp nomc="id_producte" tipus="int" esnull="no" />
    <camp nomc="id_idioma" tipus="int" esnull="no"/>
    <camp nomc="descripcio_curta" tipus="varchar" mida="30"
                                                    esnull="no" />
    <camp nomc="descripcio_llarga" tipus="varchar" mida="100" />
    <constriccio nomct="pk_desc_prod" tipus="PK">
      <columna>id_producte</columna>
      <columna>id_idioma</columna>
    </constriccio>
    <constriccio nomct="fk_desc_prod_id" tipus="FK">
      <columna>id_producte</columna>
      <referencia nomtr="producte">
        <campsr>id_producte</campsr>
      </referencia>
    </constriccio>
    <constriccio nomct="fk_des_prod_idioma" tipus="FK">
      <columna>id_idioma</columna>
      <referencia nomtr="idioma">
        <campsr>id_idioma</campsr>
      </referencia>
    </constriccio>
  </taula>

<!-- TAULA: desc_tip_prod -->
  <taula nomt="desc_tip_prod">
    <camp nomc="id_tipus_producte" tipus="int" esnull="no" />
    <camp nomc="id_idioma" tipus="int" esnull="no"/>
```

```
     <camp nomc="descripcio" tipus="varchar" mida="50" esnull="no"/>
     <constriccio nomct="pk_desc_tip_prod" tipus="PK">
        <columna>id_tipus_producte</columna>
        <columna>id_idioma</columna>
     </constriccio>
     <constriccio nomct="fk_desc_tip_prod_tipusprod" tipus="FK">
        <columna>id_tipus_producte</columna>
        <referencia nomtr="tipus_producte">
          <campsr>id_tipus_producte</campsr>
        </referencia>
     </constriccio>
     <constriccio nomct="fk_desc_tip_prod_idioma" tipus="FK">
        <columna>id_idioma</columna>
        <referencia nomtr="idioma">
          <campsr>id_idioma</campsr>
        </referencia>
     </constriccio>
   </taula>

<!-- TAULA: factura_enviament -->
   <taula nomt="factura_enviament">
     <camp nomc="id_fact_env" tipus="varchar" mida="10" esnull="no"/>
     <camp nomc="nif" tipus="varchar" mida="9" esnull="no"/>
     <camp nomc="nom" tipus="varchar" mida="15" esnull="no"/>
     <camp nomc="cognoms" tipus="varchar" mida="30" esnull="no"/>
     <camp nomc="adreca" tipus="varchar" mida="45" esnull="no"/>
     <camp nomc="poblacio" tipus="varchar" mida="20" esnull="no"/>
     <camp nomc="codi_posta" tipus="varchar" mida="5" esnull="no"/>
     <camp nomc="pais" tipus="varchar" mida="15" esnull="no"/>
     <camp nomc="telefon" tipus="varchar" mida="9" />
     <camp nomc="fax" tipus="varchar" mida="9" />
     <camp nomc="email" tipus="varchar" mida="30" esnull="no" />
     <camp nomc="nomenv" tipus="varchar" mida="15" esnull="si"/>
     <camp nomc="cognomenv" tipus="varchar" mida="30" esnull="si"/>
     <camp nomc="adrecaenv" tipus="varchar" mida="45" esnull="si"/>
     <camp nomc="poblacioenv" tipus="varchar" mida="20" esnull="si"/>
     <camp nomc="codi_postalenv" tipus="varchar" mida="5"
                                                      esnull="si"/>
     <camp nomc="paisenv" tipus="varchar" mida="15" esnull="si"/>
     <constriccio nomct="pk_factura_enviament" tipus="PK">
        <columna>id_fact_env</columna>
     </constriccio>
   </taula>

<!-- TAULA: linia_comanda -->
   <taula nomt="linia_comanda">
     <camp nomc="id_linia" tipus="int" esnull="no" />
     <camp nomc="id_comanda" tipus="varchar" mida="10" esnull="no"/>
     <camp nomc="id_producte" tipus="int" esnull="no"/>
     <camp nomc="descripcio" tipus="varchar" mida="15" esnull="no" />
     <camp nomc="unitats" tipus="decimal" mida="7,2" esnull="no"
                                                    condicio=">0" />
     <camp nomc="preu_unitari_brut" tipus="decimal" mida="10,2"
                                        esnull="no" condicio=">0" />
     <camp nomc="dte" tipus="decimal" mida="4,2" esnull="no"
                                                   perdefecte="0" />
     <camp nomc="preu_net" tipus="decimal" mida="10,2" esnull="no"
                                                    condicio=">0" />
     <constriccio nomct="pk_linia_comanda" tipus="PK">
       <columna>id_linia</columna>
       <columna>id_comanda</columna>
```

```
    </constriccio>
    <constriccio nomct="fk_linia_comanda_producte" tipus="FK">
      <columna>id_producte</columna>
      <referencia nomtr="producte">
        <campsr>id_producte</campsr>
      </referencia>
    </constriccio>
    <constriccio nomct="fk_linia_comanda_comanda" tipus="FK">
      <columna>id_comanda</columna>
      <referencia nomtr="comanda">
         <campsr>id_comanda</campsr>
      </referencia>
    </constriccio>
  </taula>

<!-- TAULA: pertanyer -->
  <taula nomt="pertanyer">
    <camp nomc="id_producte" tipus="int" esnull="no" />
    <camp nomc="id_tipus_producte" tipus="int" esnull="no"/>
    <constriccio nomct="pk_pertanyer" tipus="PK">
      <columna>id_producte</columna>
      <columna>id_tipus_producte</columna>
    </constriccio>
    <constriccio nomct="fk_pertanyer_id" tipus="FK">
      <columna>id_producte</columna>
      <referencia nomtr="producte">
        <campsr>id_producte</campsr>
      </referencia>
    </constriccio>
    <constriccio nomct="fk_pertanyer_tipus" tipus="FK">
      <columna>id_tipus_producte</columna>
      <referencia nomtr="tipus_producte">
         <campsr>id_tipus_producte</campsr>
      </referencia>
    </constriccio>
  </taula>

<!--VISTA: wDeclaracio347 -->
  <vista nomv="wDeclaracio347">
    <select>
       <columnas>nif</columnas>
       <columnas>nom</columnas>
       <columnas>cognoms</columnas>
       <columnas>import_acumulat_compres</columnas>
       <taulaf>client_registrat</taulaf>
       <condiciow>import_acumulat_compres>3000</condiciow>
    </select>
  </vista>
</database>
```

Com es pot observar, a més a més de la creació de les taules i els seus corresponents camps i constriccions, segons la definició de la base de dades vista en els apartats anteriors, també s'hi ha afegit la creació d'un índex (a part dels propis que creen les constriccions per defecte), i també s'ha creat un exemple de vista "wDeclaracio347" que pot ser utilitzada per realitzar la declaració en front l'agència tributària dels clients que han comprat per imports superiors a 3.000 €.

2.4.4 Resultat d'aplicar la transformació a la BD d'exemple

Finalment i amb l'objectiu de completar l'exemple, es mostra el resultat d'aplicar les transformacions del full d'estil XSL definit anteriorment, sobre el document XML mostrat en l'apartat anterior. El resultat d'aquesta transformació, correspon al seguit de sentències SQL per a la creació de la base de dades de la "botiga on-line" que es mostra a continuació:

```
/*Estructura de la Base de Dades tendaonline*/

CREATE TABLE idioma(
   id_idioma int not null,
   descripcio varchar(30) not null,
   CONSTRAINT pk_idioma  PRIMARY KEY (id_idioma)
);

CREATE TABLE client(
   id_client int not null,
   CONSTRAINT pk_client  PRIMARY KEY (id_client)
);

CREATE TABLE tipus_client(
   id_tipus_client int not null,
   dte decimal(4,2) DEFAULT '0' not null,
   CONSTRAINT pk_tipus_client  PRIMARY KEY (id_tipus_client)
);

CREATE TABLE producte(
   id_producte int not null,
   preu_actual decimal(10,2) not null CHECK (preu_actual >0),
   es_oferta varchar(1) DEFAULT 'N' not null CHECK (es_oferta IN
                                                    ('N','S')),
   preu_oferta decimal(10,2) CHECK (preu_oferta >0),
   estoc_inicial decimal(12,2) DEFAULT '0' not null,
   estoc_actual decimal(12,2) DEFAULT '0' not null,
   estoc_notificacio decimal(12,2) DEFAULT '0' not null,
   CONSTRAINT pk_producte  PRIMARY KEY (id_producte)
);

CREATE TABLE tipus_producte(
   id_tipus_producte int not null,
   dte decimal(4,2) DEFAULT '0' not null,
   CONSTRAINT pk_tipus_producte  PRIMARY KEY (id_tipus_producte)
);

CREATE TABLE forma_enviament(
   id_enviament int not null,
   CONSTRAINT pk_forma_enviament  PRIMARY KEY (id_enviament)
);

CREATE TABLE forma_pagament(
   id_pagament int not null,
   CONSTRAINT pk_forma_pagament  PRIMARY KEY (id_pagament)
);

CREATE TABLE associacio(
   id_client int not null,
   id_tipus_client int not null,
```

```
   CONSTRAINT pk_associacio  PRIMARY KEY (id_client,id_tipus_client),
   CONSTRAINT fk_associacio_idclient  FOREIGN KEY (id_client)
                                       REFERENCES client(id_client),
   CONSTRAINT fk_associacio_tipusclient  FOREIGN KEY
          (id_tipus_client) REFERENCES tipus_client(id_tipus_client)
);

CREATE TABLE classificacio(
   id_classif int not null,
   id_tipus_generic int not null,
   id_tipus_especific int not null,
   CONSTRAINT pk_classificacio  PRIMARY KEY (id_classif),
   CONSTRAINT fk_classificacio_tgeneric  FOREIGN KEY
     (id_tipus_generic) REFERENCES tipus_producte(id_tipus_producte),
   CONSTRAINT fk_classificacio_tespec  FOREIGN KEY
    (id_tipus_especific) REFERENCES tipus_producte(id_tipus_producte)
);

CREATE UNIQUE INDEX idx_tipus_class ON classificacio
                     (id_tipus_generic ASC, id_tipus_especific DESC);

CREATE TABLE client_registrat(
   id_client int not null,
   nif varchar(9) not null,
   nom varchar(15) not null,
   cognoms varchar(30) not null,
   email varchar(30) not null,
   contrasenya varchar(10) not null,
   pregunta varchar(100),
   resposta varchar(50),
   data_naixement date not null,
   estat_civil varchar(1) DEFAULT 'S' not null CHECK (estat_civil in
                                                        ('C','S')),
   data_primera_compra date,
   data_darrera_compra date,
   import_acumulat_compres decimal(12,2) DEFAULT '0' not null,
   nombre_compres int DEFAULT '0' not null,
   baixa_logica varchar(1) DEFAULT 'N' not null CHECK (baixa_logica
                                                      in ('N','S')),
   CONSTRAINT pk_client_registrat  PRIMARY KEY (id_client),
   CONSTRAINT u_client_registrat  UNIQUE (email),
   CONSTRAINT fk_client_registrat  FOREIGN KEY (id_client) REFERENCES
                                                    client(id_client)
);

CREATE UNIQUE INDEX idx_nif_client_r ON client_registrat (nif ASC);

CREATE TABLE estat_comanda(
   id_estat int not null,
   CONSTRAINT pk_estat_comanda  PRIMARY KEY (id_estat)
);

CREATE TABLE comanda(
   id_comanda varchar(10) not null,
   id_client int not null,
   id_fact_env varchar(10) not null,
   total_comanda decimal(11,2) not null CHECK (total_comanda >0),
   data_comanda date not null,
   hora_inici_comanda date not null,
   hora_fi_comanda date not null,
   adre_ip_comanda varchar(16),
```

```
   id_estat int not null,
   num_transaccio int not null,
   data_transaccio date not null,
   id_resultat_transaccio int not null,
   id_pagament int not null,
   id_enviament int not null,
   data_lliurament date not null,
   hora_lliurament date,
   CONSTRAINT pk_comanda  PRIMARY KEY (id_comanda),
   CONSTRAINT fk_comanda_estat  FOREIGN KEY (id_estat) REFERENCES
                                          estat_comanda(id_estat),
   CONSTRAINT fk_comanda_enviament  FOREIGN KEY (id_enviament)
                            REFERENCES forma_enviament(id_enviament),
   CONSTRAINT fk_comanda_pagament  FOREIGN KEY (id_pagament)
                               REFERENCES forma_pagament(id_pagament)
);

CREATE TABLE desc_tip_client(
   id_tipus_client int not null,
   id_idioma int not null,
   descripcio_tipus_client varchar(50) not null,
   CONSTRAINT pk_desc_tip_client  PRIMARY KEY (id_tipus_client,
                                                        id_idioma),
   CONSTRAINT fk_desc_tip_client_id  FOREIGN KEY (id_tipus_client)
                            REFERENCES tipus_client(id_tipus_client),
   CONSTRAINT fk_desc_tip_client_idioma  FOREIGN KEY (id_idioma)
                                         REFERENCES idioma(id_idioma)
);

CREATE TABLE desc_env(
   id_enviament int not null,
   id_idioma int not null,
   descripcio varchar(50) not null,
   CONSTRAINT pk_desc_env  PRIMARY KEY (id_enviament, id_idioma),
   CONSTRAINT fk_desc_env_idioma  FOREIGN KEY (id_idioma) REFERENCES
                                                    idioma(id_idioma)
);

CREATE TABLE desc_estat(
   id_estat int not null,
   id_idioma int not null,
   descripcio varchar(50) not null,
   CONSTRAINT pk_desc_estat  PRIMARY KEY (id_estat, id_idioma),
   CONSTRAINT fk_desc_estat_idioma  FOREIGN KEY (id_idioma)
                                         REFERENCES idioma(id_idioma)
);

CREATE TABLE desc_pag(
   id_pagament int not null,
   id_idioma int not null,
   descripcio varchar(50) not null,
   CONSTRAINT pk_desc_pag  PRIMARY KEY (id_pagament, id_idioma),
   CONSTRAINT fk_desc_pag_idioma  FOREIGN KEY (id_idioma) REFERENCES
                                                    idioma(id_idioma)
);

CREATE TABLE desc_prod(
   id_producte int not null,
   id_idioma int not null,
   descripcio_curta varchar(30) not null,
   descripcio_llarga varchar(100),
```

```
   CONSTRAINT pk_desc_prod  PRIMARY KEY (id_producte, id_idioma),
   CONSTRAINT fk_desc_prod_id  FOREIGN KEY (id_producte) REFERENCES
                                                producte(id_producte),
   CONSTRAINT fk_des_prod_idioma  FOREIGN KEY (id_idioma) REFERENCES
                                                     idioma(id_idioma)
);

CREATE TABLE desc_tip_prod(
   id_tipus_producte int not null,
   id_idioma int not null,
   descripcio varchar(50) not null,
   CONSTRAINT pk_desc_tip_prod  PRIMARY KEY (id_tipus_producte,
                                                           id_idioma),
   CONSTRAINT fk_desc_tip_prod_tipusprod  FOREIGN KEY
    (id_tipus_producte) REFERENCES tipus_producte(id_tipus_producte),
   CONSTRAINT fk_desc_tip_prod_idioma  FOREIGN KEY (id_idioma)
                                        REFERENCES idioma(id_idioma)
);

CREATE TABLE factura_enviament(
   id_fact_env varchar(10) not null,
   nif varchar(9) not null,
   nom varchar(15) not null,
   cognoms varchar(30) not null,
   adreca varchar(45) not null,
   poblacio varchar(20) not null,
   codi_posta varchar(5) not null,
   pais varchar(15) not null,
   telefon varchar(9),
   fax varchar(9),
   email varchar(30) not null,
   nomenv varchar(15),
   cognomenv varchar(30),
   adrecaenv varchar(45),
   poblacioenv varchar(20),
   codi_postalenv varchar(5),
   paisenv varchar(15),
   CONSTRAINT pk_factura_enviament  PRIMARY KEY (id_fact_env)
);

CREATE TABLE linia_comanda(
   id_linia int not null,
   id_comanda varchar(10) not null,
   id_producte int not null,
   descripcio varchar(15) not null,
   unitats decimal(7,2) not null CHECK (unitats >0),
   preu_unitari_brut decimal(10,2) not null CHECK
                                                (preu_unitari_brut>0),
   dte decimal(4,2) DEFAULT '0' not null,
   preu_net decimal(10,2) not null CHECK (preu_net >0),
   CONSTRAINT pk_linia_comanda  PRIMARY KEY (id_linia, id_comanda),
   CONSTRAINT fk_linia_comanda_producte  FOREIGN KEY (id_producte)
                                    REFERENCES producte(id_producte),
   CONSTRAINT fk_linia_comanda_comanda  FOREIGN KEY (id_comanda)
                                       REFERENCES comanda(id_comanda)
);

CREATE TABLE pertanyer(
   id_producte int not null,
   id_tipus_producte int not null,
   CONSTRAINT pk_pertanyer  PRIMARY KEY (id_producte,
```

```
                                                    id_tipus_producte),
   CONSTRAINT fk_pertanyer_id  FOREIGN KEY (id_producte) REFERENCES
                                                  producte(id_producte),
   CONSTRAINT fk_pertanyer_tipus  FOREIGN KEY (id_tipus_producte)
                         REFERENCES tipus_producte(id_tipus_producte)
);

CREATE VIEW wDeclaracio347 AS
   SELECT  nif, nom, cognoms, import_acumulat_compres
   FROM client_registrat
   WHERE import_acumulat_compres>3000
   ;
```

2.5 Executar la transformació (Xml2Sql)

Fins aquest moment s'ha mostrat la definició del DTD que indica l'estructura i jerarquia dels elements i atributs, juntament amb les restriccions que han de complir els documents XML "vàlids", per tal que, segons aquesta especificació continguin l'estructura d'una base de dades. També s'ha mostrat com s'ha definit el full d'estil XSL que ens servirà com a base de la transformació d'un document XML "vàlid" en un conjunt de sentències SQL. Finalment, s'ha definit el model conceptual i lògic de la base de dades d'una botiga de venta on-line per internet, i la seva representació en XML, juntament amb el resultat de la transformació segons el full d'estil XSL, cosa que ha donat com a resultat el conjunt de sentències SQL que permetran la creació de la base de dades de la botiga.

Ara doncs, queda per definir com s'uneixen tots aquests elements per tal de poder fer efectiva la transformació d'un document XML "vàlid" en un fitxer SQL, utilitzant el full d'estil XSL.

Per tal de portar a terme aquesta tasca, Java ens proporciona un API que permeten el processament de documents XML. L'API JAXP (Java API for XML Processing), suporta un seguit de tecnologies per tal de processar documents XML, entre altres destaquen:

- SAX, Simple API for XML
- DOM, Document Object Model API from W3C
- XSLT, XML Style Sheet Language Transformations from W3C
- XPath, XML Path Language from W3C
- JDOM, "Java optimized" document object model API from jdom.org

En general, direm que una aplicació està basada en XML, si es capaç de consumir documents XML, aplicar la seva lògica de negoci o recuperar informació, i generar altres documents XML (o altres formats). Aquestes tres fases, poden descriure's genèricament amb més detall de la següent manera:

1. Processament d'entrada XML
    - Analitzar i validar el document font
    - Reconèixer/cercar informació important basant-se en la seva localització o etiquetatge en el document font.
    - Extreure informació un cop s'ha localitzat

    - Opcionalment, mapejar/unir la informació recuperada a objectes de negoci.
2. Maneig de la lògica de negoci
    - Processament real de la informació d'entrada resultant opcionalment en la generació d'informació de sortida.
3. Processament de sortida
    - Construir un model de document a generar amb DOM, JDOM, etc.
    - Aplicar fulls d'estil XSLT o serialitzar directament a XML.

SAX i DOM són els models de processament més comuns. Per utilitzar SAX per processar un document XML, s'ha de codificar mètodes per a manejar esdeveniments llançats per l'analitzador a mesura que troba diferents tokens del llenguatge de marques. Amb DOM, s'ha d'escriure codi capaç de passar a través d'una estructura de dades tipus arbre, creada per l'analitzador des del document font.

Donat que un analitzador SAX genera un flux temporal d'esdeveniments, s'ha de fer els quatre passos del processament d'entrada XML descrits anteriorment, en un únic cicle (anàlisi, reconeixement, extracció i mapeig); així, cada esdeveniment capturat, es mapejat immediatament i la informació important es passa juntament amb l'esdeveniment.

Utilitzant DOM, el processament de l'entrada XML es realitza, com a mínim, en dos cicles: primer, l'analitzador DOM crea una estructura de dades en forma d'arbre que modela el document font XML (arbre DOM), seguidament es passa a través de l'arbre DOM cercant la informació que es desitja extreure i es processa posteriorment. En aquest cas, l'últim cicle es pot repetir tantes vegades com sigui necessari, mentre l'arbre DOM estigui en memòria.

Per la seva banda, XSLT és un model de processament XML de nivell superior a SAX i DOM. XSLT, tal com s'ha indicat anteriorment, requereix la codificació d'un seguit de regles (plantilles) que seran aplicades en el moment en que es trobin els patrons especificats en el document font. Aquests patrons s'especifiquen utilitzant el llenguatge Xpath. Xpath s'utilitza per localitzar i extreure informació des del document font i esta especialment dirigit als passos 1.b i 1.c del processament de XML indicat anteriorment.

Així doncs, mentre que SAX i DOM necessiten de codificació en Java, XSLT (a part del propi motor d'invocació) únicament necessita la creació de fulls d'estil (XSL) que són en si mateixos documents XML.

En el cas que ens ocupa, i com s'ha pogut observar durant el transcurs d'aquest document, la tecnologia escollida per tal de fer el processament i la transformació dels documents XML "vàlids" dissenyats per contenir l'estructura d'una base de dades, en un seguit de sentències SQL que permetin la creació de la mateixa en un SGBD, ha estat la tecnologia XSLT.

Així doncs, la transformació dels documents XML en SQL, s'ha portat a terme mitjançant SAXON 7.9, el qual és un paquet que conté una col·lecció d'eines per processar documents XML, el qual compta com a components principals:

- Amb un processador XSLT, el qual implementa la versió 2.0 de XSLT i les recomanacions per a XPath del World Wide Web Consortium.

- Un processador XPath 2.0.
- Un processador XQuery 1.0.

Saxon però no implementa totalment XSLT 2.0, XPath 2.0 o XQuery 1.0. La seva omissió més notable és que no suporta els XML Schema, cosa que no ens afecta en aquest cas, donat que s'ha utilitzat un DTD per definir l'estructura del document XML.

Mitjançant Saxon es pot executar la transformació de documents XML "vàlids", directament des de la línia de comandes. La classe Java `net.sf.saxon.Transform` conté el programa que s'utilitza per aplicar el full d'estil sobre el document XML. Això es pot fer, aplicant la comanda:

```
java net.sf.saxon.Transform [opcions] document-xml full-estil
                                                    ↳[parametres]
```

Així doncs, en el suposat cas que el nostre document XML es digui `botiga.xml`, i el nostre full d'estil s'anomeni `database.xsl`, la transformació per l'obtenció d'un document anomenat `botiga.sql`, es pot portar a terme directament executant la següent comanda:

```
java net.sf.saxon.Transform botiga.xml database.xsl > botiga.sql
```

Igualment, la transformació XSL es pot portar a terme mitjançant un programa Java, utilitzant l'API JAXP.

Com es pot veure en el codi Java que es mostra a continuació, els passos a seguir en aquest cas, són els següents:

- Crear una nova factoria de transformadors.
- Crear un nou transformador des de la factoria, amb un full d'estil particular.
- Aplicar el full d'estil al document XML, per generar la sortida producte de la transformació.

El codi Java creat en aquest cas, és el següent:

```
import net.sf.saxon.OutputURIResolver;
import net.sf.saxon.om.DocumentInfo;
import net.sf.saxon.xpath.XPathEvaluator;
import org.xml.sax.*;
import org.xml.sax.helpers.XMLFilterImpl;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.sax.SAXResult;
import javax.xml.transform.sax.SAXSource;
import javax.xml.transform.sax.SAXTransformerFactory;
import javax.xml.transform.sax.TransformerHandler;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;
```

```
import java.io.*;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Properties;


public class Xml2Sql {
  public static void main(String argv[]) {

    String db_xml = argv[0];//el primer argument indica el fitxer XML
    String db_xsl = argv[1];// el segon argument indica el fitxer XSL

    try {
         xml2sql(db_xml, db_xsl);
    } catch (Exception ex) {
         handleException(ex);
    }
   }

  /* xml2sql fa efectiva la transformació del document XML d'entrada
   *  segons el full d'estil indicat (XSL), i reporta el resultat
   *  en un fitxer anomenat 'database.sql'
   */
  public static void xml2sql(String sourceID, String xslID)
     throws TransformerException, TransformerConfigurationException {

    // Crea una nova factoria de transformadors
    TransformerFactory tfactory = TransformerFactory.newInstance();

    // Crea un nou transformador des de la factoria, amb un full
    // d'estil particular
    Transformer transformer=tfactory.newTransformer(new
                                              StreamSource(xslID));

    // Aplica el full d'estil al document XML (sourceID), i escriu
    // el resultat com a sortida en el fitxer "database.sql"
    transformer.transform(new StreamSource(sourceID),
                          new StreamResult(new File("database.sql")));
 }

  private static void handleException(Exception ex) {
    System.out.println("EXCEPCIO: " + ex);
    ex.printStackTrace();

    if (ex instanceof TransformerConfigurationException) {
        System.out.println();
        System.out.println("Error");

        Throwable ex1 =
             ((TransformerConfigurationException) ex).getException();

        if (ex1!=null) {
            ex1.printStackTrace();

            if (ex1 instanceof SAXException) {
                Exception ex2 = ((SAXException) ex1).getException();

                System.out.println("Sub-excepcio interna: ");
                ex2.printStackTrace();
            }
        }
```

```
    }
  }
}
```

Un cop compilat, l'execució de la transformació utilitzant aquest programa, es pot portar a terme de la següent manera:

```
java Xml2Sql document-xml full-estil
```

Així doncs, seguint l'exemple anterior podem executar:

```
java Xml2Sql botiga.xml database.xsl
```

Donant com a resultat un fitxer anomenat "`database.sql`", amb sintaxi SQL.

Sigui quin sigui el mètode utilitzat per executar la transformació, el resultat és, com és de preveure, exactament el mateix.

3. Alternatives en la creació d'interfícies

La major part de les empreses, institucions, organismes governamentals, etc., disposen de grans volums d'informació continguts en diverses bases de dades, sovint heterogènies, gestionats per entorns fortament propietaris, d'on moltes vegades a resultat relativament difícil fer un ús de la informació continguda en elles per altres propòsits que els propis pels quals s'havia concebut.

Aquesta informació, ha circulat històricament entre els diferents departaments de l'empresa, o en les seves relacions comercials amb altres empreses o clients, en suport paper (mitjançant la transmissió per correu postal, fax, etc.). Aquest fet, comporta grans pèrdues de temps i de diners a causa del processament repetitiu de la mateixa informació, per part d'operaris que han de replicar múltiples vegades les mateixes dades en els diferents sistemes d'informació, amb els riscos evidents que això comporta en la transcripció de la mateixa (dades errònies, registres duplicats, etc.).

A mesura que les tecnologies de la informació i la comunicació s'han anat imposant, s'han anat obrin nous horitzons en l'accés a la informació, i han creat nous paradigmes de les relacions B2B I B2C. Cada cop més, aquestes empreses han anat tenint una major necessitat de disposar d'una major integració de la informació, tant internament per compartir dades entre els seus departaments, com externament per compartir dades amb altres empreses. Això ha significat per tant, la construcció d'interfícies d'intercanvi d'informació entre els diferents sistemes d'informació d'una mateixa empresa, o amb altres sistemes d'altres empreses, cosa que resulta sovint una tasca complexa.

3.1 Creació d'interfícies

Per realitzar l'intercanvi de dades d'una forma eficient, les empreses o departaments involucrats han de posar-se d'acord en un format comú que faciliti el processarment automàtic de dades abans d'enviar-los o rebre'ls.

Els agents que intervenen en la definició d'una interfícies, han de tenir en compte les següents elements (tot i que sovint només se'n tenen en compte alguns d'ells):

- Format de les dades d'acord amb els requeriments.
- Establiment de mecanismes de la seguretat de la transmissió.
- Forma de transmetre les dades sobre una xarxa de telecomunicacions.
- Forma de realitzar el processament de la informació un cop aquesta ha arribat al seu destinatari.

En molts casos però, només acostuma a tenir-se en compte alguns d'aquests elements. En tot cas, els que segur que hauran de quedar definits, són el primer i l'últim, és a dir, el format de les dades d'acord amb els requeriments, i la forma de realitzar el processament de la informació.

Així doncs, quan un reduït nombre de departaments o empreses decideixen intercanviar un tipus determinat informació, acostumen a establir un acord puntual

entre totes les parts on es defineix, entre altres coses, el format d'intercanvi de la informació i el seu posterior tractament i processament d'acord al tipus d'informació a transmetre.

Moltes vegades però, ha estat necessari la creació d'interfícies estandarditzades per poder donar cobertura a les necessitats d'intercanvi d'informació entre sectors que incloïen un gran nombre d'actors.

Alguns d'aquests estàndards, s'han limitat a definir únicament el "format dels fitxers" que utilitzarien per intercanviar la informació entre uns i altres. Altres estàndards però, han anat més enllà, i han definit també altres elements, com poden ser el sistema de comunicació, el procediment, seguretat, etc. de la informació.

Entre altres podem destacar alguns exemples d'aquestes interfícies que han estat desenvolupades o definides amb major o menor detall segons el cas. Són exemple d'això:

- Les normes del Consell Superior Bancari, les quals estableixen en un conjunt de quaderns, el conjunt de normes, procediments, formats de fitxers, formats de documents, etc., que defineixen, entre moltes altres coses, les relacions d'intercanvi d'informació entre empreses i entitats bancàries, o entre les pròpies entitats bancàries.
- Sistema INDALO per l'intercanvi de dades entre administracions públiques. Aquest fou un projecte de gran abast, que pretenia entre altres coses, definir els interfícies per a l'intercanvi d'informació entre els diferents estaments de les administracions públiques, el qual no ha estat mai definit en la seva totalitat. Tot i així, si que s'utilitzen determinades interfícies definides en aquest model, com per exemple, la transmissió de dades entre ajuntaments i INE (Instituto Nacional de Estadística), cadastre, etc.
- EDI (Intercanvi Electrònic de Dades). EDI és segurament el que té un major abast. Tradicionalment l'objectiu d'EDI ha estat reemplaçar formes de negoci tradicionals com factures, ordres de compres, etc., pels seus equivalents en format electrònic. Així, es pot definir l'EDI com la transferència entre ordinadors d'informació comercial en format estandarditzat per a facilitar l'adquisició, classificació, cerca, consulta, etc.
  Els dos estàndards més importants, ANSI X.12 i UN/EDIFACT, proporcionen controls bàsics per a garantir la integritat de les transaccions i la correcció del format de dades. ANSI X.12 és l'estàndard més utilitzat als Estats Units i al Canadà, mentre que EDIFACT és la norma més utilitzada a Europa. Tots dos sistemes defineixen un llenguatge alfanumèric per a l'organització de documents en format electrònic.

En tots els casos, una de les parts més importants en la definició d'aquestes interfícies, és el formats dels fitxers a intercanviar. La virtut en el cas de les interfícies estandarditzades, és que hi ha un organisme que vetlla pel compliment, adaptació i estandardització del mateix, cosa que fa que tinguin una gran acceptació.

3.1.1 Formatació de la informació mitjançant series alfanumèriques d'elements d'informació

En aquest cas, la codificació ve determinada per la separació de les parts que componen la seqüència de dades. Algunes opcions habituals són les següents:

- **Per longitud fixa:** Es tracta d'emplenar cada fila del fitxer amb el contingut d'un registre d'informació, i al seu torn, assignar un determinat espai fix en la fila, per cada element d'informació o camp. Aquest sistema és compacte però inflexible. Només sembla recomanable per a dades de longitud sempre fixa, ja que en dades de longitud variable, o es desaprofita molt d'espai o es corre el risc de no poder expressar alguna dada, i quan es dona aquest cas, no hi ha manera d'arreglar-ho, donat que es tracta d'un conveni entre els participants, cosa que implicaria haver d'acordar el canvi per a fer ajustos.
- **Delimitada:** Es tracta també d'emplenar cada fila del fitxer amb el contingut d'un registre d'informació, i en aquest cas, incloure els elements d'informació de cada registre en la fila separats per un símbol delimitador. Així doncs, s'ha de reservar un símbol per separar cada element d'informació (si el delimitador apareix entre les dades, s'ha de definir una manera "d'escapar-lo").
- **Per files:** En aquest últim cas, es tracta de disposar en cada fila del fitxer un elements d'informació, i s'estableix un acord per determinar el número de files que configuren un registre.

El problema d'utilitzar aquest tipus de codificació o format, en els intercanvis d'informació entre sistemes, són bàsicament:

- La informació que contenen aquests fitxers és sovint intel·ligible, i només es pot interpretar la informació, un cop realitzat un procés que tracti les dades. Per exemple, suposem un registre codificat amb longitud fixa, on s'expressi: (identificador de client, número de compte i import a pagar):

  ```
  120210540554515150150547487653800224544
  ```

  Com es pot comprovar, es fa difícil de distingir quin valor és quin, sense mirar el format en que esta codificat el fitxer.
- Aquest sistema de codificar la informació, no acostuma a disposar de cap mètode per validar la informació continguda en el mateix. Així doncs, és molt difícil poder avaluar si un determinat elements d'informació inclòs en un registre, compleix determinades restriccions, com per exemple: que correspongui a un determinat tipus de dades, acotar els possibles valors que pot agafar, etc.
- No existeix cap mètode per accedir a parts concretes de la informació, sense tractar prèviament tot/s el/s fitxer/s. Així doncs, per accedir a qualsevol elements d'informació inclòs a un fitxer codificat amb aquests formats, previ s'haurà de tractar i processar completament, per després accedir de forma còmoda a la informació.
- Un altre problema important d'aquests formats de codificació, també és la dificultat que implica realitzar qualsevol canvi en la informació que tingui continguda un fitxer. Així doncs, un petit canvi o eliminació d'un element d'informació de qualsevol d'aquests fitxers plans, pot ocasionar la pertorbació

de tot el fitxer. Així que normalment, quan es desitja realitzar qualsevol canvi en algun element d'informació, el més habitual és tornar a generar tot el fitxer.

- Sovint també, un problema que es produeix quan les interfícies són fruit d'un acord puntual, esdevé a l'hora de realitzar un canvi en l'estructura o format de la pròpia interfície. A cada canvi que es pretengui portar a terme, s'haurà d'arribar a acords novament amb totes les parts implicades.

3.1.2 Creació d'interfícies per a fitxers plans

Tal com s'ha dit anteriorment, un cop ha estat definida i acordada entre totes les parts, l'estructura del fitxer a transmetre per tal de poder realitzar l'intercanvi d'un tipus informació concreta, és necessari també que cada actor que intervé en l'intercanvi d'informació, implementi una interfície que li permeti realitzar-ne el processament.

Degut a les característiques del format d'aquests fitxers d'intercanvi (fitxers plans), normalment les interfícies que es creen per al seu processament, acostumen a ser molt especialitzades, i per tant, no és possible estandarditzar cap dels processos de càrrega i descàrrega. Així per exemple, cap element de la interfície pel tractament dels processos de càrrega i descàrrega dels enviaments entre l'INE i un Ajuntament, per la transmissió de les variacions en el padró d'habitants, és vàlid en la interfície que realitzarà el tractament dels processos de càrrega i descàrrega entre el mateix Ajuntament, i la gerència territorial del cadastre. Això és així fins i tot en aquest cas, en que han estat definits sota el mateix projecte d'estandardització dels enviaments entre administracions públiques, sota el marc del projecte INDALO citat anteriorment.

Aquesta alta especialització de les interfícies en el tractament d'informació transmesa amb fitxers que contenen informació formatada mitjançant series alfanumèriques d'elements d'informació (a partir d'ara, fitxers plans), és dona per la pròpia naturalesa del fitxers. Tal com s'ha dit anteriorment, d'antuvi no es disposa de cap tipus de control ni d'informació extra sobre el fitxer a processar, de manera que no podem aplicar cap mena de processament que ens ajudi a determinar quines són les característiques d'aquest fitxer.

D'aquesta manera, quan hom es planteja la creació d'una interfícies pel processament d'aquest tipus de fitxers, el primer que ha de fer, és creure's que el fitxer que li arribarà es troba en el format esperat. A partir d'aquí, aplica els mecanismes necessaris per fer la lectura del fitxer, i a partir d'aquesta, anar descomposant el document segons una informació que es troba preestablerta en la pròpia definició de la interfície. Un exemple d'això, podria ser el següent. Suposem el següent fitxer pla, amb informació relativa als clients d'una perruqueria:

```
01Sílvia Datsira i Colom   Rentar i marcar1906200409:00
02Alex Sabata i Pardell    Tallar          1906200409:00
01Arnau Sabata i Datsira   Tallar          1906200410:00
02Pepita Pérez Barjuan     Tenyir          1906200409:30
```

En aquest cas, la interfície que realitzarà el processament d'aquest fitxer contra el sistema d'agenda de la perruqueria, ha de suposar en primer lloc que el fitxer li arriba en el format adequat, donat que no disposa de cap mecanisme que li permeti la validació del document abans del seu processament. Així mateix, la interfície ha de conèixer exactament quin és el format d'aquest fitxer. D'aquesta manera, dins de la pròpia interfície, ha d'estar definit que els dos primers caràcters refereixen a l'identificador del perruquer/a que realitzarà el treball, que els caràcters que van de la posició 3 a la 27, indiquen el nom del client, que els següents caràcters (de la posició 28 a la 42) refereixen al tipus de treball que s'ha de realitzar, que els següents 8 caràcters refereixen a la data, i els altres 5 fins al final de línia indiquen l'hora en que s'iniciarà el treball.

Així mateix, el format del fitxer en el cas de l'exemple està disposada per "longitud fixa", de manera que la posició dels elements només es vàlida per la transmissió d'aquesta informació. D'aquesta manera, aquesta interfície no serà valida per llegir cap altre fitxer que contingui informació disposada de forma diferent. I al seu torn, no hem d'oblidar que aquesta és només una possible manera de representar informació en fitxers plans, així doncs, la informació també hagués pogut estar separada per el caràcter "|" (o qualsevol altre) per tal de delimitar-ne la separació entre un cap i un altre.

A més d'aquesta altíssima especialització de que s'ha de dotar la interfície per tal de poder interpretar el contingut dels fitxers en el moment de la seva lectura, també ha d'estar altament especialitzada en quan al processament que s'ha de fer amb la informació un cop aquesta ha estat llegida del fitxer. Així doncs, entre altres, la interfície haurà de saber exactament quin és el tractament que ha de fer a cada porció d'informació, també haurà de saber quina operació ha de fer amb la informació llegida, i finalment, haurà de saber com processar aquesta informació sobre la base de dades, si es dona el cas.

Aquesta altíssima especialització que es requereix en les interfícies per al processament de fitxers plans, es producte de la manca d'informació addicional de que disposen a l'hora de tractar els fitxers, donat que aquesta no està continguda dins del propi fitxer, ni es troba en cap fitxer de suport. Així mateix, tampoc no existeixen eines estandarditzades que permetin un preprocessament de la informació continguda en aquests fitxers, per tal d'exercir cap tipus de validació, localització d'informació, transformació, etc.

3.1.3 Formatació de la informació mitjançant XML

El llenguatge XML (eXtensible Markup Language) fou desenvolupat pel XML Working Group, format sota l'hospici del World Wide Web Consortium (W3C). Aquest llenguatge permet d'estructurar la informació, mitjançant la definició d'unes unitats d'informació anomenades *metadades*[2], que permeten de definir el tipus d'informació que volem emmagatzemar. D'aquesta manera cada element d'informació va acompanyat d'etiquetes que especifiquen el tipus d'informació al qual fa referència.

Es diu que un document XML és correcte (well-formed), quan l'estructura del document compleix una sèrie de característiques. Entre altres:

[2] Les metadades són una mena d'etiquetes que s'utilitzen per a marcar i classificar la informació.

- Hi ha un element pare (root), que conté tots els altres elements.
- Cadascuna de les etiquetes obertes que defineix un tipus d'informació té una altra etiqueta que tanca la definició, fent servir la parella.

```
<tipus_informació1>informació tipus1</tipus_informació1>
```

- Tots els elements d'informació estan etiquetats en una estructura en arbre.

La part *extensible* del llenguatge XML es refereix a la possibilitat de crear nous "diccionaris" d'etiquetes que permetin de catalogar la informació de cada organització.

Quan dos organitzacions o empreses desitgen compartir informació, prèviament han de conèixer, compartir i donar un mateix sentit als termes del "diccionari" d'etiquetes. La solució per a aconseguir aquesta compartició d'informació és assegurar que l'etiquetatge de la informació compleix un estàndard reconegut que especifiqui les etiquetes disponibles i la seva estructura, cosa que s'aconsegueix mitjançant la definició de tipus de documents (DTD), mitjançant XML Schema, etc.

Així doncs, es diu que un document XML és un document ben format, quan les dades que conté estan correctament estructurades; al seu torn, un document XML "ben format", és a més a més un document "vàlid", quan compleix una DTD determinada, és a dir, quan la informació s'ha etiquetat seguint els tipus definits per aquest estàndard.

D'entre els avantatges que proveeix XML per a la creació d'interfícies respecte la formatació de la informació mitjançant sèries alfanumèriques d'elements d'informació, se'n poden destacar els següents:

- La informació és fàcilment interpretable. La informació en aquest cas, està molt estructurada, i es veu clarament on comença i on acaba cada elements d'informació.
- XML disposa de mecanismes (com el DTD, XML Schema, etc.) que permeten definir la seva jerarquia i/o l'estructura, els elements i atributs que el composen, i les restriccions dels mateixos, cosa que permet la validació d'un fitxer XML abans de processar-lo. Per tant, permet la seva validació en quan a estructura, etiquetatge, i fins i tot, permet validar si la informació d'un determinat element d'informació, es correspon amb el tipus de dades esperat, o si es troba dins dels valors permesos que pot tenir aquell camp, entre altres.
- Així mateix, XML també disposa d'eines que permeten accedir a parts concretes (nodes) dins del document XML, com si d'un directori d'informació es tractés.
- Es relativament fàcil realitzar canvis en els elements d'informació continguts en un documents XML, sense haver de processar novament tot el document. Això és així, donat que la informació està continguda dins del fitxers de forma relativament clara, i per tant, és fàcil accedir a la informació que es desitja modificar. En tot cas, és força difícil pertorbar tot el document (com passava en els fitxers plans), i en cas que es cometés un error greu en la modificació del document que afectes a l'estructura del mateix, sempre pot ser validat abans del seu processament, tal com s'ha indicat anteriorment.

- Els canvis en les estructures dels documents, també acostumen a ser més fàcils quan parlem d'XML, donat que això només representa un canvi en el "diccionari" d'etiquetes i en l'estructura del fitxer; cosa que en aquest cas es simplifica molt, donat que s'ha d'arribar a un acord sobre quin serà el nou DTD o XML Shema a utilitzar.

Utilitzant XML, és possible trenca la dinàmica seguida habitualment en l'intercanvi d'informació mitjançant interfícies, on un document es crea -> es transmet -> es rep -> es processa utilitzant programes individuals. Així doncs, podem parlar d'objectes actius que tenen processos associats, i que depenent de la informació que contingui, permetin executar unes accions o unes altres.

XML per tant, no és simplement una manera més de representar un fitxer pla alfanumèric que conté uns determinats elements d'informació estructurats segons un sistema d'etiquetes, sinó que va més enllà, donat que disposa de mecanismes estandarditzats per a la seva validació, accés i representació entre altres.

Avui en dia, són moltes les empreses, administracions i entitats, que estan migrant les seves interfícies per la transmissió d'informació en les seves relacions B2B o B2C, o entre els diferents departaments de la mateixa, passant de les interfícies tradicionals normalment estandarditzats mitjançant acords puntuals sobre la manera d'intercanviar fitxers plans, a interfícies definides en XML. En aquest sentit, podríem parlar de multitud de casos, com per exemple: Una nova revisió de EDI, anomenada EDI/XML, el qual pretén ser un nou marc per a l'intercanvi electrònic de dades en l'entorn del comerç electrònic; o la creació d'un entorn de tramitació d'expedients administratius entre les diferents administracions públiques, que agafa XML com a base per al processament i intercanvi de la informació.

3.1.4 Creació d'interfícies per a fitxers XML

En el cas en que l'intercanvi d'informació es realitzi mitjançant document XML, i a diferència del cas anterior, no és imprescindible que tots els actors que intervenen es posi d'acord en el format (en aquest cas en l'estructura) els documents XML a transmetre. De fet n'hi ha prou en que cada part anunciï com desitja que siguin els documents XML que espera rebre pel seu processament (mitjançant DTD's, XML Schemas, etc.). Això evidentment, no implica que no es puguin establir acords entre les parts.

Sigui com sigui, en aquest cas també serà necessari implementar interfícies que permetin el tractament de la informació transmesa, per tal de poder-ne fer un ús. Però a diferència de l'anterior (interfícies per a fitxers plans), els documents XML ens donen tota la informació sobre el contingut del fitxer, i a més, sovint es disposa d'informació complementaria relacionada amb el document, però externa a la interfície, que ens proveeix informació sobre l'estructura del mateix (fitxers DTD, XML Schema, etc.). Finalment, i tal com s'ha dit en l'apartat anterior, XML disposa d'un conjunt d'eines estandarditzades que ens permeten la validació, la cerca i l'accés a la informació, entre altres.

Una possible representació de l'exemple de "l'agenda de la perruqueria" mostrat en l'apartat sobre la creació d'interfícies per a fitxers plans, amb XML podria ser el següent:

```
<agenda_perru>
   <reserva>
      <perruquer>01</perruquer>
      <client>Sílvia Datsira i Colom</client>
      <tipus_treball>Rentar i marcar</tipus_treball>
      <dia>19/06/2004</dia>
      <hora>09:00</hora>
   </reserva>
   <reserva>
      <perruquer>02</perruquer>
      <client>Alex Sabata i Pardell</client>
      <tipus_treball>Tallar</tipus_treball>
      <dia>19/06/2004</dia>
      <hora>09:00</hora>
   </reserva>
   <reserva>
      <perruquer>01</perruquer>
      <client>Arnau Sabata i Datsira</client>
      <tipus_treball>Tallar</tipus_treball>
      <dia>19/06/2004</dia>
      <hora>10:00</hora>
   </reserva>
   <reserva>
      <perruquer>02</perruquer>
      <client>Pepita Pérez Barjuan</client>
      <tipus_treball>Tenyir</tipus_treball>
      <dia>19/06/2004</dia>
      <hora>09:30</hora>
   </reserva>
</agenda_perru>
```

En funció de les necessitats, aquesta estructura podria ser diferent. Sigui com sigui l'estructura del document XML, queda clar només a simple vista, que el document és molt més entenedor fins i tot per a l'ull humà.

Com s'ha dit, les estructures dels documents XML poden ser definits mitjançant DTD, XML Schema, etc. Suposem que s'ha definit el següent DTD per l'exemple de l'agenda de la perruqueria:

```
<!ELEMENT agenda_perru (reserva+)>
<!ELEMENT reserva (perruquer+,client,tipus_treball+,dia,hora)>
<!ELEMENT perruquer (#PCDATA)>
<!ELEMENT client (#PCDATA)>
<!ELEMENT tipus_treball (#PCDATA)>
<!ELEMENT dia (#PCDATA)>
<!ELEMENT hora (#PCDATA)>
```

Tot i que la informació del document XML per l'agenda de la perruqueria que em posat en aquest cas, conté exactament la mateixa informació que en l'exemple del fitxer pla, observant el document DTD veiem que en realitat tenim més opcions en

aquest cas quan demanem hora en aquesta perruqueria. En concret, en aquest cas segons el tipus de treball que hem demanat, pot ser que ens atengui més d'un perruquer/a. Així mateix, també podem demanar més d'un servei. Així doncs, el següent document XML seria vàlid:

```
<agenda_perru>
   <reserva>
      <perruquer>01</perruquer>
      <client>Sílvia Datsira i Colom</client>
      <tipus_treball>Rentar i marcar</tipus_treball>
      <tipus_treball>Tractament facial</tipus_treball>
      <dia>19/06/2004</dia>
      <hora>09:00</hora>
   </reserva>
</agenda_perru>
```

Per tant, la representació de la informació en XML pot ser molt més complexa que en els fitxers plans, sense complicar el propi document.

Tal com s'ha dit, XML disposa d'eines estàndards que ens permet avaluar que el document XML sigui vàlid, només indicant quin és el seu DTD associat (un document vàlid, és un document ben format i que compleix l'estructura d'un DTD concret).

Així mateix, també podrem disposar d'eines estàndards que ens permeten accedir a part de la informació, sense haver de processar tot el fitxer, així doncs, podríem consultar tots els clients que ens han visitat durant un dia concret, sense haver de tractar la informació relacionada amb el tipus de treball que s'han realitzat, hora, etc.

Si ens centrem en els treballs propis de la interfície, i en el tractament que aquesta ha de fer dels documents XML, podem afirmar que a diferència del cas anterior, en aquest cas, tenim resolt amb eines estandarditzades tota la fase de lectura i interpretació que la interfície haurà de fer de la informació continguda en el fitxer. Per tant, aquesta part de la interfície no depèn en absolut del tipus de document (no és "especialitzada").

Pel que fa al tractament posterior de la informació per part de la interfície, un cop aquesta ja ha estat validada i interpretada, i tal com s'ha vist en el capítol anterior, cal destacar que XML ens aporta una gran quantitat de recursos per tal que la gestió d'aquesta sigui àgil. Fent un petit recordatori de quines són algunes de les tecnologies més destacades:

- SAX (Simple API for XML) per tal de poder anar disparant accions a mesura que es llegeix el document XML, en funció de la informació trobada dins del document
- DOM (Document Object Model API from W3C) el qual ens guarda l'estructura de la informació llegida en memòria, per tal de poder accedir a diferents part de la informació continguda en aquest, tantes vegades com ens sigui necessari, sense haver de tornar a llegir el document XML.
- XSLT (XML Style Sheet Language Transformations from W3C) per tal de poder realitzar transformacions del document XML, a altres documents XML, o a altres formats.

Utilitzant i combinant totes aquestes tecnologies que ens proveeix XML, podem implementar interfícies que siguin molt més estàndards, i no tant dependents de la informació que es transmet.

4. Interfície per l'intercanvi d'informació amb la BD de la Botiga On-line

En aquest capítol es detalla els disseny i la implementació d'un sistema d'intercanvi d'informació entre la base de dades de la Botiga On-line dissenyada en un capítol anterior, i qualsevol sistema d'informació exterior. Aquest intercanvi d'informació es realitza mitjançant processos de càrrega i descàrrega de documens XML.

Els processos de càrrega i descàrrega dissenyats per a la construcció d'aquesta interfície, no són una simple importació/exportació d'informació en vers la base de dades, sinó que disposen d'una certa "intel·ligència" que els capacita per detectar possibles errades de format i/o de contingut. En el cas de detectar errades en els processos de càrrega/descarrega, la informació no es desestima, sinó que resta en el sistema pendent de la intervenció (manipulació/correcció) per part de l'agent emissor.

En aquest capítol, s'exposa el disseny i implementació dels fonaments de l'estructura de la interfície i els processos de càrrega/descàrrega d'informació (insertar, reenviar, esborrar, descarregar) i el sistema de control de la qualitat dels mateixos, deixant per més endavant (següent capítol) el disseny dels processos de manipulació (correcció) dels enviaments fallits per part de l'agent emissor i que ja estiguin precarregats en el sistema.

4.1 Objectius

L'objectiu general d'aquesta segona part del projecte és l'elaboració d'una interfície de càrrega/descarrega d'informació entre la base de dades de la Botiga On-line dissenyada en un capítol anterior, i qualsevol altre sistema d'informació exterior. En concret, s'ha dividit el procés de construcció d'aquesta interfície en tres fases:

- Disseny de l'estructura general de la interfície, per als processos de càrrega/descàrrega, tenint en compte l'existència del sistema de control de qualitat.
- Disseny i construcció de l'estructura de suport a la interfície i als processos de càrrega/descàrrega, per tal de proveir i garantir el control de la qualitat.
- Disseny de les eines necessàries per permetre la identificació i correcció d'enviaments fallits.

Tal com s'ha dit anteriorment, aquest capítol cobreix la creació de la interfície, i per tant, l'objectiu d'aquest està centrat en el disseny general de la interfície i de l'estructura que li dona suport pel tractament del control de qualitat, i en la implementació de totes dues (fase 1 i 2). Per tant, es deixa pel següent capítol el disseny de les eines necessàries per permetre la identificació i correcció d'enviaments fallits (fase 3).

Així doncs, els objectius a cobrir són:

- Disseny de l'arquitectura de la interfície.
- Disseny dels casos d'ús corresponent a la interfície.

- Disseny del diagrama d'estats que ha de seguir el tractament dels registres per als processos de càrrega/descàrrega.
- Disseny del diagrama de seqüències.
- Disseny de la Descripció del Tipus de Document (DTD) que han de complir els documents XML a processar, per tal que aquests siguin "vàlids".
- Disseny i implementació dels elements que donen suport als processos de càrrega/descàrrega i control de qualitat de la interfície.
- Implementació de la interfície.

És un objectiu general d'aquest projecte, i en concret de la construcció de la interfície, fer que aquesta gaudeixi de la màxima independència possible del sistema operatiu en la que s'executi, del sistema gestor de bases de dades (SGBD) contra la que treballi, i finalment, dotar-la de la màxima d'independència possible de la base de dades per la qual ha estat dissenyada. D'aquesta manera, es pretén que aquesta mateixa interfície pugui treballar en qualsevol altre sistema operatiu, actuar contra qualsevol altre SGBD i exercir tasques de càrrega/descàrrega per a qualsevol base de dades amb uns mínims retocs o adaptacions, o com a mínim, que aquests es puguin portar a terme amb facilitat, sense haver de redissenyar i reprogramar tota la interfície.

4.2 Disseny de la interfície

Tal com s'ha exposat anteriorment, la interfície a dissenyar ha de permetre la càrrega i/o descàrrega d'informació entre la base de dades de la Botiga On-line i qualsevol altre sistema d'informació exterior, mitjançant l'intercanvi de documents XML.

Donat que aquesta interfície, no és una simple eina d'importació/exportació lleugera d'informació, s'ha dotat el sistema (tant a nivell de base de dades, com a nivell d'interfície) per tal d'encabir-hi un control de qualitat que permeti l'identificació, classificació i tractament posterior de les possibles errades de format i/o de contingut que puguin aparèixer.

Els processos de càrrega d'informació, han de suportar el processament de documents XML que denotin la incorporació d'un registre nou a la base de dades (insertar), la modificació d'un o diversos registres (reenviar), o l'eliminació d'informació de la base de dades (esborrar). Així mateix, els processos de descàrrega d'informació, han de suportar el processament de documents XML que denotin una consulta sobre una taula o una vista de la base de dades. En aquest segon cas, la interfície ha de generar un nou document XML amb els registres resultants de la consulta (descarregar).

El subsistema de control de qualitat és l'encarregat d'avaluar el nivell de correctesa dels documents XML a tractar. Aquest control de qualitat s'aplica sobre qualsevol document d'entrada, indistintament que es tracti d'insercions, reenviaments, esborraments o descàrregues. Aquests sistema de control de qualitat, està centrat bàsicament en:

- Determinar que el document XML a processar estigui correctament identificat, i al mateix temps, sigui un document "vàlid" segons les especificacions establertes (l'especificació del DTD).

- Determinar que es compleixen les funcions de qualitat definides i precarregades en el sistema, ja sigui a nivell de camps o de registre:
    - A nivell de camp, s'avalua individualment que els diferents camps (o alguns d'aquests) continguts en el document XML siguin correctes (tipus de dades, format o contingut) segons les especificacions de la configuració de qualitat. Per exemple, que un determinat camp d'una taula sigui un NIF correcte, o sigui del tipus data, etc.
    - A nivell de registre, i en funció dels requeriments i particularitats de cada taula de la base de dades, s'avalua que la informació a processar sigui consistent i integra, i per tant, que no contingui errades a nivell lògic. El control de qualitat a nivell de registre, només s'aplica en el cas de les "insercions" i els "reenviaments", i no en els "esborraments" o "descàrregues", donat que ens aquests últims la informació continguda en el document XML pot ser incompleta. Per exemple, una de les qüestions que es poden avaluar a nivell de registre, és que el conjunt de camps d'un determinat registre tinguin coherència entre ells, així doncs, es pot avaluar entre altres, que la data d'inici d'una comanda sigui inferior a la data de la mateixa comanda, entre molts altres.

      Quan és diu que en els "esborraments" o "descàrregues" la informació a nivell de registre pot ser incompleta, es fa referència a que pot ser que en una consulta sobre l'exemple anterior, només s'indiqui la data de d'inici de la comanda, i no la data de final (p.e.: consultar tots els registres en que la data d'inici de la comanda sigui igual a "19/05/2004", sense indicar informació sobre la data de final).
- Determinar que es compleixen les constriccions i restriccions propis de la base de dades.

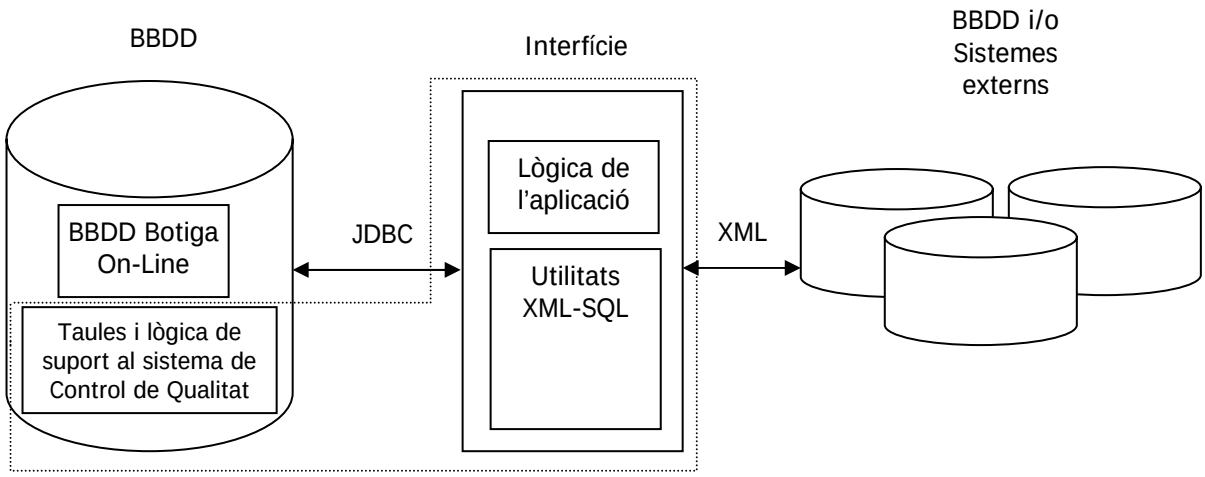
En cada cas, s'ha d'avaluar el nivell d'error obtingut en el procés de qualitat, i en funció d'aquest, el document XML ha de restar incorporat al sistema de forma diferent, segons sigui el seu nivell d'error. Així doncs, l'enviament d'un document pot ser incorporat al sistema segons sigui el seu nivell d'errors:

- *Amb errors crítics*: En cas de trobar-se errors en la identificació del document, per no tractar-se d'un document XML vàlid, etc.
- *Amb errors lleus*: En cas de detectar-se errors de poca magnitud, però que impossibiliten el processament definitiu de la informació a la base de dades.
- *Incorporat amb errors lleus*: En cas de detectar-se erros de poca magnitud, que no impossibiliten el processament de la informació a la base de dades.
- *Incorporat definitivament*: En cas de no detectar-se errors en l'enviament.

En els següents apartats es va desgranant el disseny de la interfície.

4.2.1 Arquitectura de la interfície

L'arquitectura escollida pel disseny de la interfície és la següent:



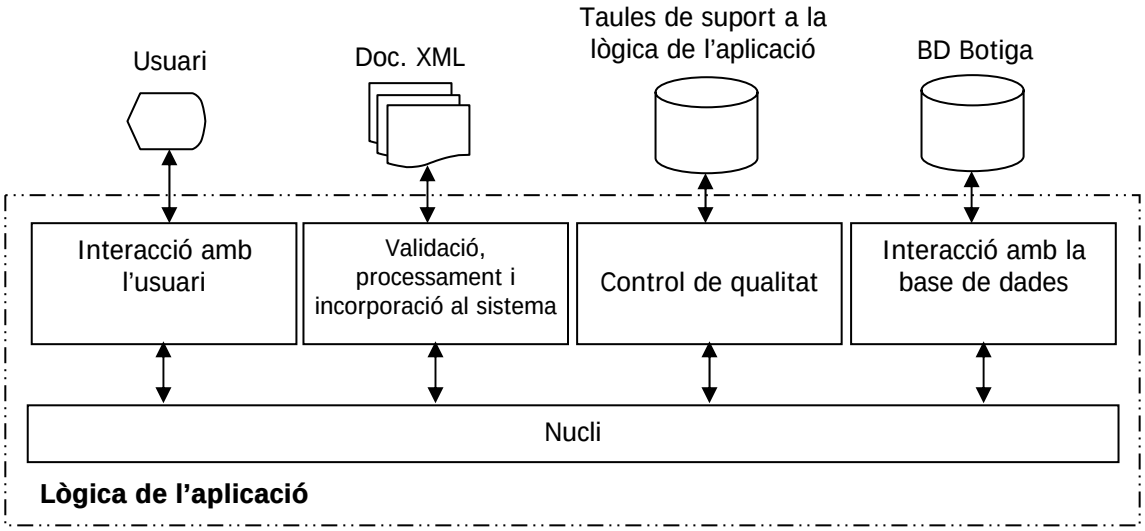
En aquesta arquitectura, es pot observar com tots els intercanvis d'informació (processos de càrrega/descàrrega) entre la base de dades de la Botiga On-Line, i qualsevol altre sistema d'informació, es realitza mitjançant l'intercanvi de documents XML.

Al seu torn, el conjunt d'elements que integren la interfície es troben a cavall entre una aplicació independent que fa de frontissa entre les enviaments de documents XML i la base de dades, i un seguit de taules i procediments o funcions hostatjats en la pròpia base de dades.

Així doncs, l'aplicació independent implementa el nucli de la interfície, els processos de lectura i processament del documents XML, i implementa les transaccions contra el SGBD. Altrament, el seguit de taules i procediments o funcions hostatjats en la pròpia base de dades, conformen l'estructura de suport a la interfície en els processos de càrrega i descàrrega, els quals garantiran i donaran suport al sistema de control de qualitat.

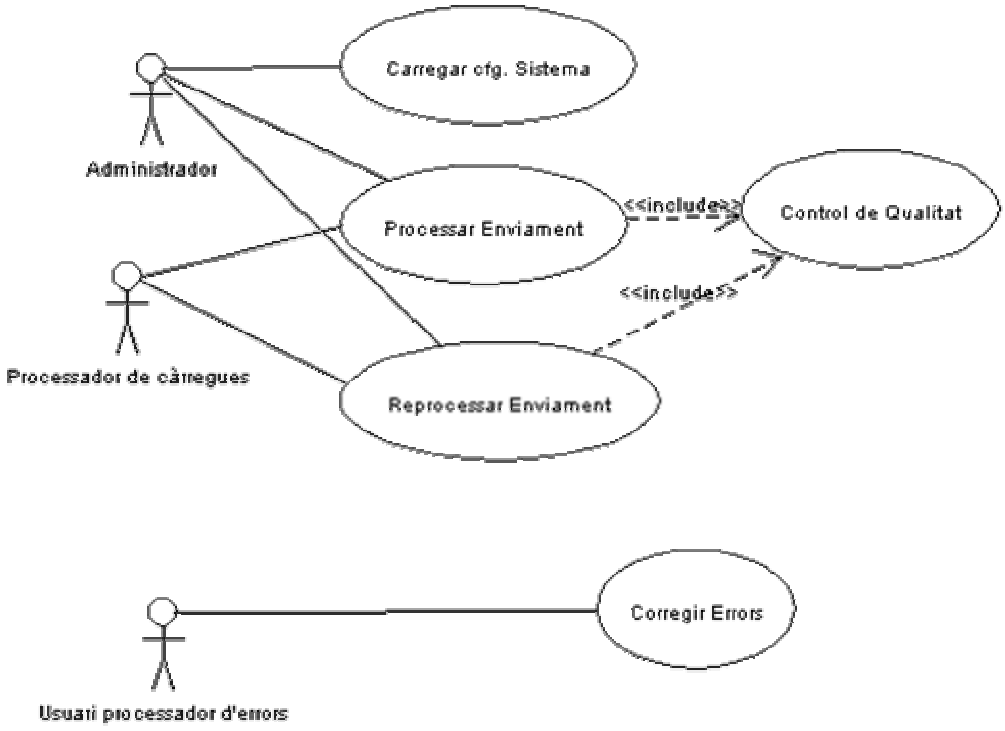
D'aquesta manera, s'ha fugit d'una implementació monolítica que probablement milloraria el rendiment general de la interfície, i s'ha apostat per un sistema més similar a una implementació per nivells, la qual dota al sistema d'una major modularitat, i per tant, proveeix de més flexibilitat per al manteniment o migració de les funcionalitats de la mateixa. No s'ha d'oblidar, que un dels objectius que s'han marcat en l'inici d'aquest capítol ha estat que el resultat final d'aquest disseny i implementació, dones com a resultat una interfície que fàcilment pogués ser migrada a altres sistemes operatius, altres SGBD o a altres bases de dades.

Una representació en nivells de l'arquitectura d'aquesta interfície, seria per tant:



La finalitat per tant de l'elecció d'aquesta arquitectura, no és altra que dotar al sistema de modularitat i la màxima independència possible en front el sistema operatiu, SGBD i base de dades de la Botiga On-Line.

4.2.2 Diagrama de casos d'ús



Cas d'ús número 1: *"Carregar cfg. Sistema"*

Resum de la funcionalitat: Carrega la configuració que especifica la relació entre alguns dels camps de les diferents taules, amb les funcions d'avaluació de la qualitat

que hauran de complir. Al mateix temps, també indica quines són les funcions a les que s'haurà de sotmetre les diferents taules que aquí s'especifiquin. En ambdós casos, també s'indica quin és el grau d'error que representa l'incompliment d'alguna d'aquestes funcions.

Paper dins del treball de "l'actor": és un cas d'ús bàsic pel sistema, donat que si el sistema no està configurat, o no està configurat adequadament, el control de qualitat desapareix, convertint la interfície en un simple traspàs d'informació.

Actors: Administrador de la base de dades, de la interfície o del subsistema de control de qualitat.

Casos d'ús relacionats: cap

Precondició: El sistema no està processant cap altra comanda. La configuració del sistema de control de qualitat no ha estat carregada prèviament.

Postcondició: El sistema de control de qualitat està correctament configurat.

Cas d'ús número 2: *"Processar enviament"*

Resum de la funcionalitat: es llegeix el document XML especificat, es determina el seu format juntament amb la seva qualitat (resultat del control de qualitat) i s'incorpora al sistema, ja sigui carregant-lo directament a la base de dades (en cas de no existència d'errors), o reservant-lo en alguna de les taules especialment creades per emmagatzemar-hi documents XML que tenen algun nivell d'error.

Paper dins el treball de "l'actor": és part fonamental de la feina del "processador de càrregues". Sense aquest cas d'ús no es poden fer "insercions", "reenviaments", "esborraments" o "descarregues" contra la BBDD.

Actors: Processador de càrregues/Administrador

Casos d'ús relacionats: Control de qualitat

Precondició: L'enviament no ha estat processat anteriorment.

Postcondició: L'enviament és processat al sistema, ja sigui en la BBDD destí (Botiga on-line), o en alguna de les taules de suport a la gestió d'errors.

Cas d'ús número 3: *"Reprocessar enviament"*

Resum de la funcionalitat: Donat un enviament previ amb errors, es torna a processar per tal de determinar si ja es vàlid per ser processat contra la base de dades. En cas de no ser vàlid, es torna a avaluar el seu nivell d'errors, i es torna a emmagatzemar en les taules de suport a la gestió d'errors. Donat que pot haver variat el seu nivell d'errors, pot ser que canviï la seva destinació en aquestes taules.

Paper dins del treball de "l'actor": és un cas d'ús molt important pel sistema, donat que permet que enviaments processats amb errors i emmagatzemats al sistema, puguin ser reprocessats per ser inclosos definitivament en la BBDD.

Actors: Processador de càrregues/Administrador

Casos d'ús relacionats: Control de qualitat

Precondició: L'enviament ha estat processat anteriorment en el sistema, i no ha estat incorporat a la BBDD degut a l'existència d'errors.

Postcondició: L'enviament és emmagatzemat al sistema, ja sigui en la BBDD destí (Botiga on-line), o en alguna de les taules de suport a la gestió d'errors.

Cas d'ús número 4: *"Control de qualitat"*

Resum de la funcionalitat: Donat un camp o un registre integrant d'un enviament, n'avalua la seva qualitat, i determina el nivell d'error.

Paper dins del treball de "l'actor": és un cas d'ús molt important pel sistema, donat que sense aquest cas d'ús, el sistema es converteix en una simple importació/exportació d'informació.

Actors: cap

Casos d'ús relacionats: Processar enviament/Reprocessar enviament

Precondició: El camp o registre està pendent de l'avaluació de la seva qualitat, i per tant, no s'ha determinat el seu nivell d'error.

Postcondició: L'enviament té associat un nivell de qualitat (error), el qual pot ser zero en cas de tractar-se d'un enviament correcte.

Cas d'ús número 5: *"Corregir errors"*[3]

Resum de la funcionalitat: Donat un usuari emissor de documents XML, identifica tots els enviaments fallits (que no han superat el control de qualitat) que aquest hagi pogut generar, i si s'escau, en permet la seva edició i manipulació (correcció).

Paper dins del treball de "l'actor": Es tracta d'un cas d'ús que presta un valor afegit a la interfície, donat que permet a tot emissor de documents XML assegurar-se que els seus enviaments s'han processat adequadament, i si no ha estat així, en permet la seva identificació, i en funció del cas, la seva manipulació (correcció).

Actors: Usuari processador d'errors

Casos d'ús relacionats: Cap

Precondició: L'usuari processador d'errors, és un usuari vàlid.

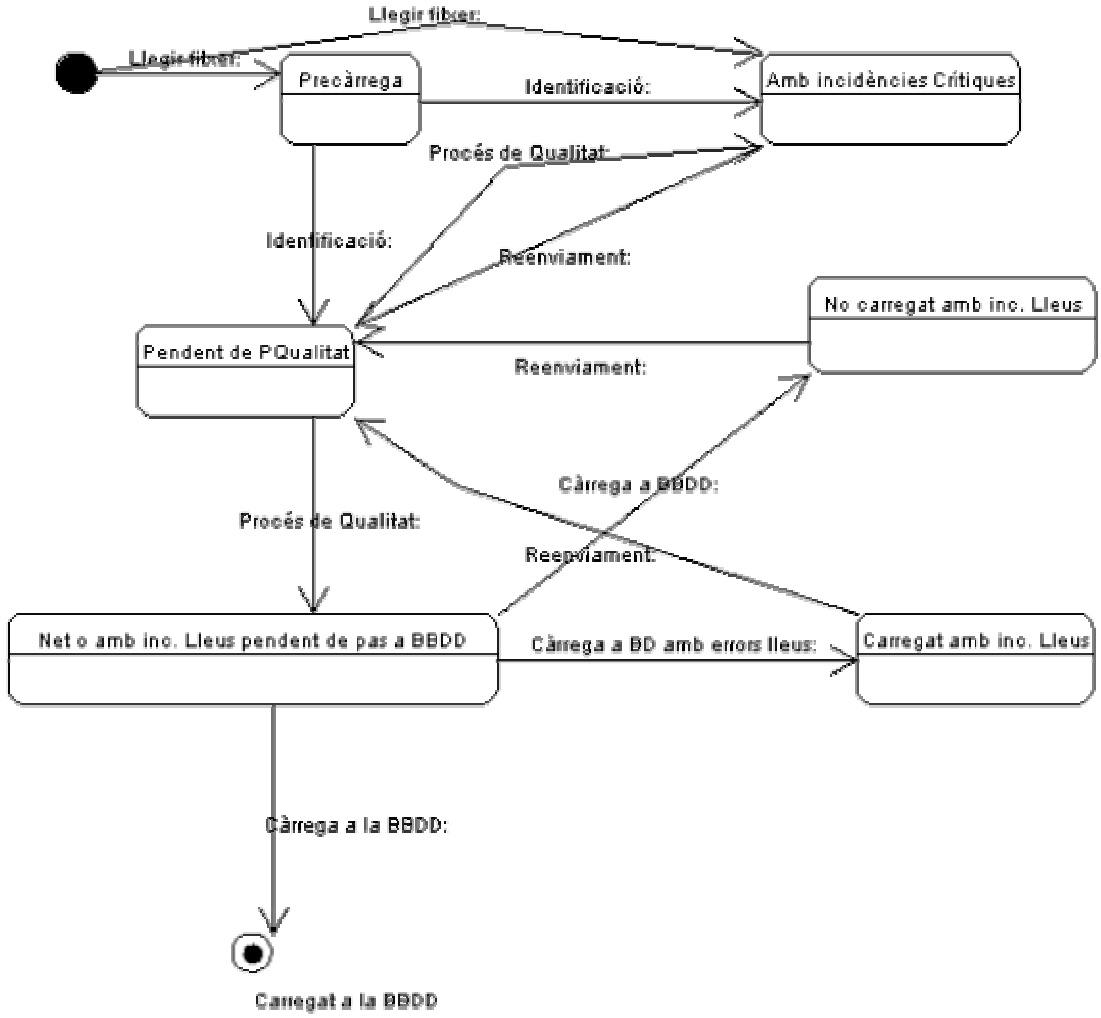
Postcondició: En funció de l'existència d'enviaments fallits, aquests són identificats. I en el cas d'haver-hi exercit alguna acció correctora, aquests seran actualitzats. En

[3] El disseny de l'eina que permet la correcció d'errors, i que està relacionada amb aquest cas d'ús, es descriu en el següent capítol.

cap cas però no seran reprocessats, essent aquesta una acció que dependrà de la intervenció del "processador de càrregues" o de "l'administrador".

4.2.3 Diagrama d'estats

El tractament dels registres segueix un diagrama d'estats, segons el moment en què es trobin durant la fase de càrrega al sistema dels documents XML. Així doncs, el diagrama d'estats és el següent:



Detall de cadascun dels estats:

Estat: "Precàrrega"

Descripció: Estat embrionari. L'enviament s'ha llegit, vàlidat i està carregat com un arbre DOM.

Estat: "Pendent de PQualitat"

Descripció: L'enviament està pendent de ser processat pel sistema de control de qualitat, i per tant, no s'ha establert el seu possible nivell d'error.

Estat: "Amb incidències crítiques"

Descripció: L'enviament està identificat de forma no correcta, o a superat el nivell d'error "lleu" en el processament del sistema de control de qualitat.

Estat: "No carregat amb inc. Lleus"

Descripció: L'enviament ha generat errors en el processament del sistema de control de qualitat. Tot i que aquests errors són considerats com a lleus pel sistema, no s'ha pogut completar el procés de "càrrega a la BBDD".

Estat: "Carregat amb inc. Lleus"

Descripció: L'enviament ha generat errors lleus en el processament del sistema de control de qualitat, però ha estat possible realitzar el procés de "càrrega a la BBDD".

Estat: "Net o amb inc. Lleus pendent de pas a BBDD"

Descripció: L'enviament no ha generat errors (o aquests han estat lleus) en el processament del sistema de control de qualitat. L'enviament es disposa a ser carregat a la BBDD.

Detall de cadascuna de les transicions:

Transició: "Llegir fitxer"

Descripció: Es llegeix el fitxer (enviament), i es determina si aquest és un document XML "vàlid". En cas afirmatiu és incorporat al sistema en l'estat "Precàrrega", en cas contrari, es passa a l'estat "Amb incidències Crítiques".

Transició: "Identificació"

Descripció: S'avalua que l'enviament estigui correctament identificat. Si és així, el document passa a l'estat "Pendent de PQualitat", en cas contrari es passa a l'estat "Amb incidències Crítiques".

Transició: "Procés de qualitat"

Descripció: Avalua la qualitat de l'enviament, i en determina el nivell d'error. En funció del nivell d'error obtingut, es passa als estats "Amb incidències Crítiques" o a "Net o amb inc. Lleus pendent de pas a BBDD".

Transició: "Càrrega a la BD amb errors Lleus"

Descripció: Realitza la "càrrega a la BBDD" de l'enviament, marcant el registre amb errors lleus.

Transició: "Reenviament"

Descripció: Canvia a l'estat "Pendent de PQualitat", per tal que torni a ser avaluada la qualitat del document.

Transició: "Càrrega a la BBDD"

Descripció: Realitza la càrrega definitiva de l'enviament a la BBDD, si aquesta és possible, i si no passa a l'estat "No carregat amb inc. Lleus".

4.2.4 Diagrames de seqüències

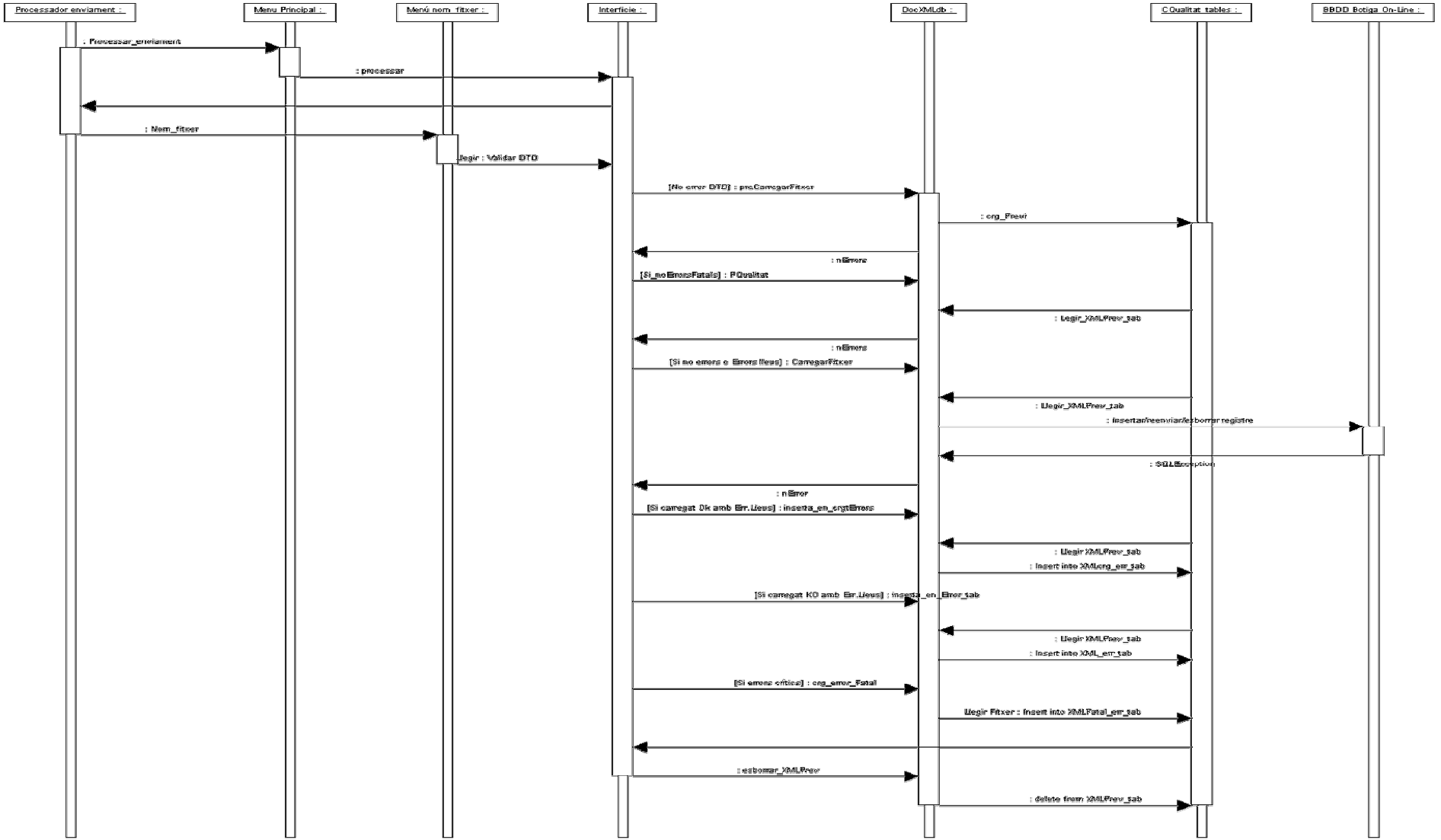
Aquests diagrames de seqüències, són una especificació del cas d'ús "Processar enviament", el qual inclou "Control de qualitat".

En primer lloc es mostra el diagrama de seqüències del processos de càrrega d'informació a la BBDD, això és: "inserció", "reenviament" i "esborrament". Acte seguit, es mostra el diagrama de seqüències relatiu als processos de descàrrega d'informació de la BBDD ("descarregar").

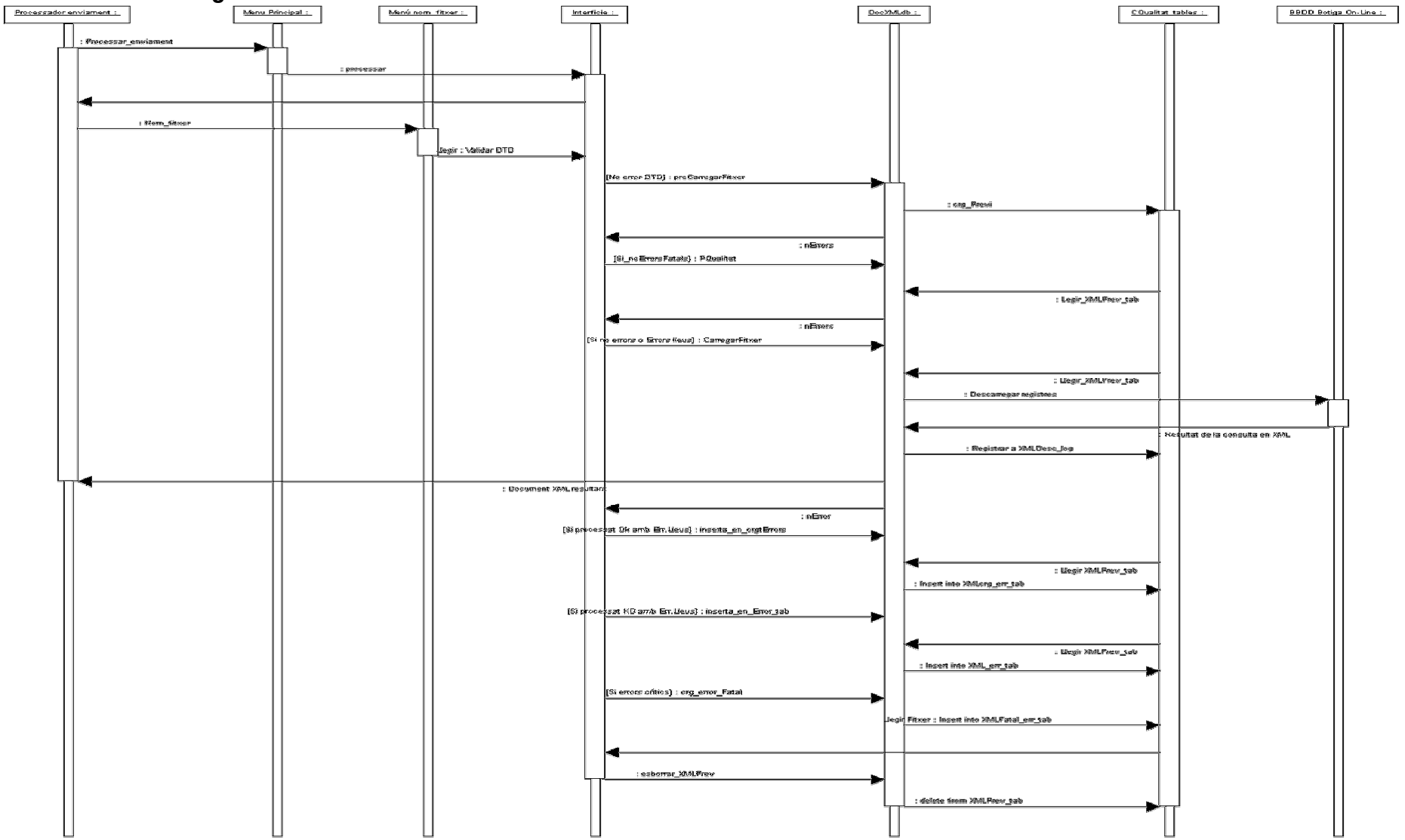
De fet, aquests dos diagrames de seqüències són en realitat un mateix, i en la construcció de la interfície, s'ha implementat com si es tractés d'un únic diagrama. El motiu de mostrar-los per separat, no és altre que aportar més claredat en la seqüència que es porta a terme en un i altre cas. Així doncs, com es pot observar, la major part de les seqüències del diagrama són comunes en ambdós casos.

El següent diagrama de seqüències ha estat realitzat amb l'eina "ArgoUML" versió 0.10 (donat que amb la versió 0.14 no funciona l'opció), el qual té diversos errors de notació UML per a la confecció de diagrames de seqüències reconeguda pels mateixos autors de l'eina (manual de l'ArgoUML 0.14 Capítol 17 pàgina 283). Així doncs, el diagrama de seqüències que aquí es mostra, no es pot entendre com un diagrama formal, però s'inclou a per tal d'aportar una major comprensió del sistema implantat.

Procés de Càrrega: Inserció/reenviament/esborrament



Procés de descàrrega:



Com es pot observar, en ambdós casos les classes de control utilitzades en el diagrama de seqüències, no s'anomenen igual que els casos d'ús implicats. Això és així per una qüestió de claredat en la notació. Així doncs, en les classes de control utilitzades en el diagrama de seqüències s'han utilitzat els noms reals de la implementació.

4.2.5 Especificació del format dels fitxers dels enviaments

Tal com s'ha anat comentant durant tot el capítol, l'intercanvi d'informació entre la base de dades Botiga On-Line i qualsevol sistema d'informació exterior, es realitzarà mitjançant l'enviament de documents XML vàlids. Així doncs, en aquest apartat es defineix com ha de ser un document XML vàlid.

Per tal de definir com han de ser les documents XML, s'ha dissenyat un DTD (Definició de Tipus de Document), que haurà de constar com a tipus de document en tots els enviaments que la interfície processi. La Definició del Tipus de Document, és la següent:

```
<!ELEMENT enviament (registre)>
<!ATTLIST enviament emissor CDATA #REQUIRED>
<!ATTLIST enviament num CDATA #REQUIRED>

<!ELEMENT registre (camp+)>
<!ATTLIST registre operacio (INSERTAR | REENVIAR | ESBORRAR |
                                        DESCARREGAR) #REQUIRED>
<!ATTLIST registre taula CDATA #REQUIRED>

<!ELEMENT camp EMPTY>
<!ATTLIST camp nom ID #REQUIRED>
<!ATTLIST camp valor CDATA #IMPLIED>
<!ATTLIST camp condicio (SI) #IMPLIED>
```

Així doncs, un document XML "vàlid", consta d'un element anomenat `enviament` que té com atributs `emissor` i `num`. El cos de l'element `enviament`, consta d'un únic element anomenat `registre`, el qual té com atributs `operacio` i `taula`. El cos de l'element `registre`, estarà format per un o diversos elements anomenats `camp`, els quals tindran com a atributs `nom`, `valor` i `condicio`.

A més, destacar les restriccions de l'atribut `operacio`, el qual només pot contenir els valors: `INSERTAR`, `REENVIAR`, `ESBORRAR` i `DESCARREGAR`. En la mateixa línia, destacar les restriccions de l'atribut `condicio`, que en cas d'estar inclòs en algun camp d'algun enviament, només pot contenir el valor `SI`.

El fet de fer constar el nom de la taula en tots els enviaments, dota a la interfície d'independència respecte a la base de dades contra la que s'estigui operant.

D'aquesta manera, s'especifica que en tot enviament ha de constar el nom de l'emissor i el número d'emissió d'aquest. S'estableix com a norma del sistema que tot nom d'emissor ha de ser exactament de 5 caràcters alfanumèrics, mentre que el

número d'enviament ha de ser exactament de 3 caràcters alfanumèrics. Aquesta descripció de "nom d'emissor" i "número d'enviament", ha de correspondre a més a més i de forma exacta, amb el nom del fitxer XML, el qual haurà de ser la composició correcta d'aquests dos elements. Així doncs, per exemple, la cinquena emissió d'un document per part de l'usuari Alex Sabata, haurà de ser de l'estil: `emissor="alexs" num="005"`; per la seva banda, el nom del fitxer d'aquest enviament haurà de dir-se `alexs005.xml`.

Així mateix, i tal com denota el DTD, en tot enviament ha de constar el tipus d'operació a realitzar, la qual podrà ser: INSERTAR, REENVIAR, ESBORRAR o DESCARREGAR. Així mateix, s'indica sobre quina taula s'ha de realitzar l'operació.

Finalment, s'especifica quins camps es veuen afectats o estan inclosos en la operació. La informació mínima que ha de contenir l'element camp ha de ser el nom del mateix, i addicionalment, es pot fer constar el valor que s'especifica pel mateix, i si es tracta d'un camp de condició o no. En aquest últim punt, el camp condició denota si un camp ha de ser inclòs un una sentència condicional de l'operació a executar, cosa que només té sentit en les operacions REENVIAR, ESBORRAR i DESCARREGAR.

Un exemple de document XML "vàlid", per cadascuna de les operacions possibles, podria ser:

Exemple d'una operació "Insertar":

```
<?xml version="1.0"?>
<!DOCTYPE enviament SYSTEM " interficie.dtd">
<enviament emissor="alexs" num="001">
 <registre operacio="INSERTAR" taula="idioma">
  <camp nom="ID_IDIOMA" valor="2"/>
  <camp nom="DESCRIPCIO" valor="ANGLÈS"/>
 </registre>
</enviament>
```

Per tal que tot sigui correcte, el nom del fitxer haurà de ser `alexs001.xml`.

Exemple d'una operació "REENVIAR":

```
<?xml version="1.0"?>
<!DOCTYPE enviament SYSTEM "interficie.dtd">
<enviament emissor="alexs" num="002">
 <registre operacio="REENVIAR" taula="client_registrat">
  <camp nom="ID_CLIENT" valor="1" condicio="SI"/>
  <camp nom="NIF" valor="43714998w"/>
  <camp nom="NOM" valor="Alex"/>
  <camp nom="COGNOMS" valor="Sabata i Pardell"/>
  <camp nom="EMAIL" valor="asabatap@ajtarrega.es"/>
  <camp nom="CONTRASENYA" valor="1234"/>
  <camp nom="PREGUNTA" valor="ets del barca?"/>
  <camp nom="RESPOSTA" valor="si"/>
```

```
  <camp nom="DATA_NAIXEMENT" valor="22/12/1972 0:0:0"/>
  <camp nom="ESTAT_CIVIL" valor="C"/>
  <camp nom="DATA_PRIMERA_COMPRA" valor="1/1/2004 0:0:0"/>
  <camp nom="DATA_DARRERA_COMPRA" valor="17/5/2004 0:0:0"/>
  <camp nom="IMPORT_ACUMULAT_COMPRES" valor="333"/>
  <camp nom="NOMBRE_COMPRES" valor="5"/>
  <camp nom="BAIXA_LOGICA" valor="N"/>
 </registre>
</enviament>
```

En aquest cas, el nom del fitxer haurà de ser `alexs002.xml`.

Exemple d'una operació "Esborrar":

```
<?xml version="1.0"?>
<!DOCTYPE enviament SYSTEM "interficie.dtd">
<enviament emissor="alexs" num="003">
 <registre operacio="ESBORRAR" taula="client_registrat">
  <camp nom="ID_CLIENT" valor="1" condicio="SI"/>
 </registre>
</enviament>
```

En aquest cas, el nom del fitxer haurà de ser `alexs003.xml`.

Exemple d'una operació "DESCARREGAR":

```
<?xml version="1.0"?>
<!DOCTYPE enviament SYSTEM "interficie.dtd">
<enviament emissor="alexs" num="004">
 <registre operacio="DESCARREGAR" taula="client_registrat">
  <camp nom="NIF" valor="43714998w" condicio="SI"/>
 </registre>
</enviament>
```

En aquest cas, el nom del fitxer haurà de ser `alexs004.xml`.

En aquest últim cas, i tractant-se d'una operació de descàrrega, la interfície generarà (si tot és correcte) un document XML que guardarà en forma de fitxer.

Donat que no es té constància de quina és l'estructura en que l'emissor del document XML amb la operació de descàrrega desitja rebre el document, s'ha realitzat una descàrrega estàndard. En cas que fos necessari, sempre es pot aplicar alguna transformació sobre el document XML resultant, per tal de convertir-lo a un document XML específic per a cada receptor.

En aquest cas, el document resultat de l'operació de descàrrega de l'enviament `alexs004.xml`, seria:

```
<?xml version = "1.0"?>
<ROWSET>
 <ROW num="1">
  <ID_CLIENT>1</ID_CLIENT>
  <NIF>43714998w</NIF>
  <NOM>Alex</NOM>
  <COGNOMS>Sabata i Pardell</COGNOMS>
  <EMAIL>asabatap@ajtarrega.es</EMAIL>
  <CONTRASENYA>1234</CONTRASENYA>
  <PREGUNTA>ets del barca?</PREGUNTA>
  <RESPOSTA>si</RESPOSTA>
  <DATA_NAIXEMENT>1/6/2004 0:0:0</DATA_NAIXEMENT>
  <ESTAT_CIVIL>C</ESTAT_CIVIL>
  <DATA_PRIMERA_COMPRA>2/2/2004 0:0:0</DATA_PRIMERA_COMPRA>
  <DATA_DARRERA_COMPRA>3/2/2004 0:0:0</DATA_DARRERA_COMPRA>
  <IMPORT_ACUMULAT_COMPRES>333</IMPORT_ACUMULAT_COMPRES>
  <NOMBRE_COMPRES>1</NOMBRE_COMPRES>
  <BAIXA_LOGICA>N</BAIXA_LOGICA>
 </ROW>
</ROWSET>
```

Aquest document serà guardat amb el nom de fitxer TENDAxxx.xml, on "xxx" correspon al valor del número de descàrregues que s'hagin realitzat en la interfície més un. Tal com s'explica més endavant, les descàrregues són guardades també en les taules de suport de la interfície, per tal de tenir un registre de les descàrregues efectuades (log). És a través d'aquesta taula d'on s'obté el següent número de descàrrega que posteriorment és utilitzat en la composició del nom del fitxer. Així doncs, si previ a aquesta operació de descàrrega s'haguessin produït 7 descàrregues, el nom del fitxer d'aquesta última seria: tenda008.xml.

4.2.6 Disseny del Control de Qualitat

Habitualment, les interfícies de càrrega d'informació per a bases de dades, no proveeixen de cap sistema de control de qualitat, o aquest és molt simple. Normalment, aquestes interfícies simples, llegeixen els fitxers d'entrada, i realitzen els processos oportuns per la càrrega directa a les taules de la base de dades. En aquest cas, en produir-se problemes o incidències durant la càrrega de la informació a la base de dades, és el propi SGBD el que genera un seguit d'excepcions, de manera que avorten el procés d'importació, truncant la importació dels registres. En aquest cas, es desestima l'enviament. Així doncs, en cap cas, aquestes interfícies simples, són capaces de detectar errors de contingut, o de format, que no estan especificats dins de les pròpies constriccions de la base de dades.

En el cas que ens ocupa, el sistema de control de qualitat és l'element que marca la diferència entre els processos de càrrega d'informació d'aquesta interfície, amb altres sistemes d'importació d'informació.

Aquest sistema de control de qualitat ha estat dissenyat per detectar anticipadament els errors que pugui portar inclosos l'enviament a tractar, ja siguin de contingut o de format.

Addicionalment i de forma relacionada amb el sistema de control de qualitat, es permet el tractament dels enviaments fallits. Així doncs, donada la seva relació, trobarem parts del disseny del sistema de control de qualitat que estan orientats al seu tractament. El disseny complert pel tractament dels enviaments fallits però, es mostra amb detall en el següent capítol.

La interfície en general, i el sistema de control de qualitat en concret, ha estat dissenyats per tal que sigui el mínim dependent de la base de dades contra la que treballen. Així doncs, s'ha complementat el sistema a nivell de taules i lògica de l'aplicació, per tal de fer possible aquesta independència. Com es podrà observar, aquesta independència no és 100% real, donat que algunes funcions d'avaluació de la qualitat requereixen tenir lligams molt estrets amb la base de dades de la Botiga On-line.

4.2.6.1 Disseny de les taules de suport de la Interfície i al control de qualitat

Per tal de facilitar el processos de càrrega/descàrrega i control de la qualitat dels diferents documents XML a través de la interfície, i al seu torn, poder gestionar el reprocessament (reenviament) d'enviaments produïts anteriorment i que hagin estat qualificats amb algun nivell d'error per part del control de qualitat, s'han creat un seguit de taules de suport als processos de la interfície, que li permetran:

- Emmagatzemar els enviaments de forma temporal, mentre duri el procés de càrrega realitzat per la interfície.
- Emmagatzemar i classificar els enviaments que han resultat fallits dins el sistema, per tal que puguin tornar a ser accessibles i poder-los manipular i reprocessar (reenviar).
- Emmagatzemar les descàrregues realitzades, per tal de disposar d'un control sobre les mateixes (log).
- Emmagatzemar la configuració del control de qualitat.

Les taules creades amb aquesta finalitat, són:

Taula: XMLprev_tab

Quan un enviament es processat per primer cop, i després de comprovar-ne la seva "validés" i la correcta identificació del mateix, o quan es produeix el reenviament d'un registre processat anteriorment, el registre s'emmagatzema de forma temporal en la taula XMLprev_tab, per tal que sigui accessible per tots els processos de càrrega/descàrrega i control de qualitat, fins que passa a ser emmagatzemat definitivament en la BBDD o temporalment en altres taules segons l'avaluació d'errors. Així doncs, XMLprev_tab és una taula de treball, on s'emmagatzemen temporalment els registres durant l'execució dels processos de càrrega/descàrrega.

Un cop finalitzats els processos de càrrega/descàrrega d'un enviament, aquesta taula ha de quedar lliure dels continguts propis de l'enviament que s'ha tractat.

```
create table XMLprev_tab (
   nom_emissor varchar2(5) not null,
```

```
   num_enviament varchar2(3) not null,
   operacio varchar(10) not null,
   nom_taula varchar(20) not null,
   xml varchar2(4000) not null,
   condicio varchar(100),
   CONSTRAINT pk_XMLprev PRIMARY KEY (nom_emissor, num_enviament)
);
```

On `nom_emissor` i `num_enviament`, identifiquen de forma única un registre en la taula. Aquests camps contindran el nom de l'emissor i el número d'enviament del mateix. Es tracta per tant, dels atributs de l'element `enviament` del document XML original.

`operacio` i `nom_taula`, identifiquen quina operació s'ha de realitzar amb el registre en vers una taula de la base de dades. Es tracta per tant, dels atributs de l'element `registre` (`operacio` i `taula` respectivament) del document XML original.

`xml` emmagatzema el conjunt d'elements `camp` del document XML original, transformats i adaptats totalment, per tal de fer més àgil els processos propis de la interfície, i processament final del registre. Així doncs, donat un element `camp` del document XML original, aquest es emmagatzemat al camp `xml` de la taula en forma de: `<nom_del_camp>valor_de_camp</nom_del_camp>` on `nom_del_camps` correspon al contingut de l'atribut `nom` de l'element `camp`, i `valor_de_camp`, correspon al contingut de l'atribut valor del mateix element. En el camp `xml`, també s'hi afegeix a l'inici del mateix la cadena `<rowset><row num=″1″>`, i es tanca amb `</row></rowset>`. Així doncs, el camp `xml` contindrà totes les instàncies de l'element camp del document XML original, de la següent manera:

```
<rowset>
   <row num=″1″>
     <nom_del_camp1>valor_del_camp1</nom_del_camp1>
     <nom_del_camp2>valor_del_camp2</nom_del_camp2>
     ...
     <nom_del_campn>valor_del_campn</nom_del_campn>
   </row>
</rowset>
```

Finalment, el camp `condicio` contindrà una cadena dels valors de l'atribut `nom` de tots els elements `camp` de document original, que estiguin identificats mitjançant l'atribut `condició=″SI″.` Per tal de separar un `nom_de_camp` d'un altre, s'ha utilitzat el caràcter "|". A l'inici d'aquesta cadena, s'identifica el número de camps que s'hi conté. Un exemple del format d'aquesta cadena, pot ser `1|ID_CLIENT`.

Així doncs, tot enviament (document XML) incorporat al sistema, es interpretat i transformat per tal de poder-lo emmagatzemar de forma adequada en aquesta taula, i així permetre un processament més àgil sobre els continguts de l'enviament.

Taules: XMLerr_tab, XMLcrgerr_tab

La finalitat d'aquestes dos taules, es molt similar. Es tracta d'emmagatzemar registres que durant el procés de càrrega/descàrrega i control de qualitat, han estat identificats amb errors lleus, ja sigui en el seu format o en el contingut. En el cas de XMLerr_tab, s'emmagatzemen tots aquells registres que han estat identificats amb errors lleus i que no han pogut ser processats a la base de dades, mentre que en XMLcrgerr_tab, es guarden aquells registres, que tot i contenir errors lleus, si que han pogut ser processats en la base de dades.

En els dos casos, es tracta de mantenir un enviament en el sistema, fins que, a través de la intervenció de l'usuari que ha realitzat l'enviament fallit, puguin ser identificades, i si s'escau, corregides i reprocessades definitivament en el sistema.

Donat que l'estructura de la taula és idèntica en ambdós casos, es mostra la seva estructura de la següent manera:

```
create table [XMLerr_tab, XMLcrgerr_tab] (
   nom_emissor varchar2(5) not null,
   num_enviament varchar2(3) not null,
   operacio varchar(10) not null,
   nom_taula varchar(20) not null,
   nivell_error int not null,
   xml varchar2(4000) not null,
   condicio varchar(100),
   CONSTRAINT [pk_XMLerr, pk_XMLcrgerr] PRIMARY KEY
                                  (nom_emissor, num_enviament)
);
```

Com es pot observar, l'estructura d'aquesta taula és molt similar a la de la taula `XMLprev_tab`. De fet, es manté l'estructura, i la representació en el contingut de cadascun dels camps, ja exposades anteriorment. En aquest cas però, s'ha afegit un camp més anomenat `nivell_error`, el qual identifica el nivell d'error detectat durant el control de qualitat.

Taula XMLfatal_err_tab

En cas que en la lectura del fitxer XML original, s'hagin detectats errors de format (no sigui un document XML vàlid), o s'hagin detectats errors en la identificació (els camps que identifiquen l'enviament, no tinguin el format exposat anteriorment, o no es corresponguin amb el nom del fitxer), o finalment, que la quantitat d'errors del document superi la consideració "d'errors lleus", el registre a carregar, serà llegit directament del fitxer XML original, i carregat directament en aquesta taula. L'estructura de la qual, és la següent:

```
create table XMLfatal_err_tab (
   nom_emissor varchar2(5) not null,
   num_enviament varchar2(3) not null,
   xml varchar2(4000) not null,
   CONSTRAINT pk_XMLfatal_err PRIMARY KEY (nom_emissor,
                                           num_enviament)
);
```

On `nom_emissor` i `num_enviament`, són recollits directament del nom del fitxer XML original. Mentre que en aquest cas, el camp `xml` contindrà la seqüència literal del contingut del fitxer original.

Taula XMLdesc_log:

A diferència de les anteriors, aquesta taula no conté les dades de cap enviament, sinó que emmagatzema tots els continguts de les descàrregues que s'hagin realitzat a través de la interfície, producte d'algun enviament amb l'operació "DESCARREGAR" inclosa. Així doncs, la funció principal d'aquesta taula és actuar com un "log" de descàrregues. L'estructura d'aquesta taula, és la següent:

```
create table XMLdesc_log (
   descarrega varchar2(3) not null,
   sol_licitant varchar2(5) not null,
   nsol_licitud varchar2(3) not null,
   xml varchar2(4000) not null,
   CONSTRAINT pk_desc_log PRIMARY KEY (descarrega)
);
```

On `descarrega` indica el número d'ordre en que s'han produït cadascuna de les descàrregues incloses en la taula.

`sol_licitant` i `nsol_licitud`, identifiquen quin ha estat l'emissor de l'ordre de descàrrega, i quin era l'identificador de l'enviament en que s'especificava l'operació.

Finalment `xml` conté el document XML que ha estat descarregat, i per tant, inclòs en el fitxer `TENDAxxx.xml` (on `xxx=descarrega`).

Taula cfg_tab:

En aquest cas, aquesta taula tampoc no conté les dades de cap enviament o registre concret, sinó que identifica la relació entre alguns dels camps de les diferents taules, o taules senceres, i les funcions de qualitat que han de superar. Juntament amb la relació (taula, camp, funció), també s'identifica el valor de l'error que es produeix en cas de no superar-se una funció concreta del control de qualitat.

L'estructura de la taula cfg_tab, és:

```
create table cfg_db (
   taula varchar(20) not null,
   camp varchar(20) not null,
   funcio varchar(25) not null,
   nerror int not null,
   CONSTRAINT pk_cfg_db PRIMARY KEY (taula,camp,funcio)
);
```

On:

- `taula`: refereix al nom de la taula de la base de dades.
- `camp`: refereix al camp de la taula relacionada amb la funció de qualitat. En aquest cas, s'utilitza el caràcter "*****", per tal de denotar que la funció de qualitat és a nivell de registre, i per tant, que afecta (o pot afectar) a diversos camps de la taula. En aquest segon cas, les funcions de qualitat no seran estàndards i vàlides per qualsevol base de dades, sinó que seran especifiques per la BBDD Botiga On-line.
- `funció`: refereix al nom de la funció (en aquest cas programada en PL/SQL), que determina si el parell (taula, camp), supera o no el control de qualitat referida en aquesta funció.
- `nerror`: refereix al grau d'error que implica el fet de no superar la funció de qualitat pel parell (taula, camp).

Així doncs, no és la funció la que retorna el nivell d'error a la funció de qualitat, sinó que ve determinat per la taula de configuració, en funció del parell (taula, camp). Això és així, per que s'ha considerat que per una mateixa funció, es pot fer consideracions a nivell d'errors diferents en funció de quina sigui la taula o camps implicats en l'avaluació.

Per tal de fer més àgil, configurable i transparent el procés de càrrega dels registres de configuració en aquesta taula, i per tant, al sistema, s'ha dissenyat un procés que permet la càrrega de la informació de configuració a partir d'un document XML. Per exemple, un document XML de configuració vàlid, pot ser:

```
<?xml version="1.0"?>
<ROWSET>
   <ROW num="1">
      <TAULA>client_registrat</TAULA>
      <CAMP>NIF</CAMP>
      <FUNCIO>es_nif</FUNCIO>
      <NERROR>1</NERROR>
   </ROW>
   <ROW num="2">
      <TAULA>client_registrat</TAULA>
      <CAMP>*</CAMP>
      <FUNCIO>client_registrat_ok</FUNCIO>
      <NERROR>4</NERROR>
   </ROW>
   <ROW num="3">
      <TAULA>idioma</TAULA>
      <CAMP>DESCRIPCIO</CAMP>
      <FUNCIO>no_nul</FUNCIO>
      <NERROR>2</NERROR>
   </ROW>
   ...
</ROWSET>
```

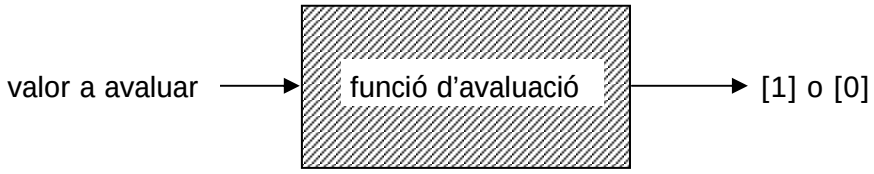
4.2.6.2 Disseny de funcions de control de qualitat

Per tal d'aconseguir que la interfície, sigui d'alguna manera independent també del sistema de control de qualitat, s'han dissenyat les funcions que avaluen el nivell d'error d'un camp o d'un registre, de forma separada de nucli de la lògica de l'aplicació de la interfície.

Així doncs, aquestes funcions d'avaluació del nivell d'error, han esta dissenyades i implementades amb PL/SQL.

Per tal de dissenyar aquestes funcions, s'ha fet una divisió entre les funcions que actuen i avaluen l'existència d'error a nivell de camp, i les funcions que actuen i avaluen l'existència d'error a nivell de registre.

- Funcions que avaluen l'existència d'errors a nivell de camp: Aquestes funcions han estat programades amb PL/SQL, i són totalment independents de la base de dades i de la interfície. No així del SGBD (en aquest cas Oracle), en tot cas, en trobar-se separades de la resta de components de la interfície, resulta fàcil reprogramar-les sense haver de tocar la resta de components.
  El disseny d'aquestes funcions, a més a més, s'ha realitzat de manera que sigui estandarditzada la seva invocació, i el valor obtingut com a retorn. Així doncs, qualsevol funció d'avaluació de l'existència d'errors a nivell de camp, es invocat passant com a paràmetre el valor del camp a avaluar. El seu retorn és 1 o 0, en funció de si és correcte o no.



- Funcions que avaluen l'existència d'errors a nivell de registre: Aquestes funcions també han estat programades amb PL/SQL, però en aquest cas, són totalment dependents de la base de dades. Així doncs, no s'entén la utilització d'aquestes funcions fora de l'àmbit de la base de dades a la qual fan referència (en aquest cas, base de dades de la Botiga On-Line). Al igual que passa amb les anteriors funcions, aquestes també són dependents del SGBD.
  El disseny d'aquestes funcions però, també ha realitzat de manera que sigui estandarditzada la seva invocació, i el valor obtingut com a retorn. Així doncs, qualsevol funció d'avaluació de l'existència d'errors a nivell de registre, es invocat passant com a paràmetre el document XML contingut en la taula XMLprev_tab. El seu retorn és 1 o 0, en funció de si és correcte o no.
  En aquest cas, la funció parseja el document XML rebut per paràmetre, i n'analitza la seva qualitat a partir de la informació continguda en el document XML.

4.2.6.3 Implementació de les funcions PL/SQL de control de qualitat

Funcions que avaluen l'existència d'errors a nivell de camp:

Tal com s'ha dit anteriorment, aquestes funcions són independents de la base de dades.

Funció `es_nif`: Avalua si el valor passat com a paràmetre, és un NIF ben format o no.

```
create or replace function es_nif (nif in char) return integer
is
   lleNif char(1);
   charNif varchar2(8);
   numNif INTEGER;
   caracters varchar2(24):='TRWAGMYFPDXBNJZSQVHLCKET';
   pos INTEGER:=0;

   begin
      lleNif:=upper(substr(nif,9,1));
      charNif:=substr(nif,0,8);
      numNif:=to_number(charNif);

      pos:=(numNif mod 23)+1;

      if lleNif=substr(caracters,pos,1) then
         return (1);
      else
         return (0);
      end if;
end es_nif;
/
```

Funció `es_numeric`: Avalua si el valor passat per paràmetre és un valor de tipus numèric.

```
create or replace function es_numeric (val in char) return
integer is
   id numeric;

   begin
      id:=val;
      return(1);
   exception
      when value_error then
         return(0);
end es_numeric;
/
```

Funció `no_nul`: Determina si el valor passat per paràmetre, conté algun caràcter, dígit o expressió.

```
create or replace function no_nul (val in char) return integer
is
   begin
      if length(val)>0 then
         return(1);
      else
         return(0);
      end if;
end no_nul;
/
```

Funció `es_data_anterior_avui`: Determina si el valor passat per paràmetre, és un valor de tipus data i el seu contingut es inferior a la data del sistema (la qual es suposa que és correcta, i per tant, marca la data actual).

```
create or replace function es_data_anterior_avui (val in char)
return integer is
   data date:=val;
   begin
      if data>sysdate() then
         return(0);
      else
         return(1);
      end if;
end es_data_anterior_avui;
/
```

Funció `es_tant_per_cent`: Comprova que el valor passat per paràmetre, sigui superior a 0 i inferior a 100.

```
create or replace function es_tant_per_cent (val in char)
return integer is
   tpc float :=val;
   begin
      if tpc>=0 and tpc<100 then
         return(1);
      else
         return(0);
      end if;
end es_tant_per_cent;
/
```

Funcions que avaluen l'existència d'errors a nivell de registre:

Tal com s'ha dit anteriorment, aquestes funcions són totalment dependents de la base de dades.

Funció client_registrat_ok: Avalua diferents camps del registre que es processa, i determina si la seva disposició i valors compleix amb la lògica de la taula.

Així doncs, en aquesta funció, s'avalua el següent:

- Que la data de la primera compra que hagi realitzat un client sigui inferior o igual a la data de la darrera compra.
- Que la data de la darrera compra sigui inferior a la data d'avui (data del sistema)
- Que en cas que s'especifiqui que el client ha realitzat un número determinat de compres, el import acumulat d'aquestes ha de ser superior a 0.

```
create or replace function client_registrat_ok(src_xmlFile in
                                        char) return integer is
   cr_p xmlparser.parser;
   cr_doc xmldom.DOMDocument;
   cr_nl xmldom.DOMNodeList;
   cr_len number;
   cr_n xmldom.DOMNode;
   cr_e xmldom.DOMElement;
   cr_tagName varchar2(30);
   cr_tagValue varchar2(100);
   cr_childNode xmldom.DOMNode;
   data_1compra date;
   data_ucompra date;
   import_compres float;
   num_compres int;
   nlpos int:=0;
begin
   cr_p:=xmlparser.newParser;
   xmlparser.setValidationMode(cr_p,FALSE);
   xmlparser.parseBuffer(cr_p,src_xmlFile);
   cr_doc:=xmlparser.getDocument(cr_p);
   cr_nl:=xmldom.getElementsByTagName(cr_doc, '*');
   cr_len:=xmldom.getLength(cr_nl);
   for j in 0..cr_len-1 loop
      cr_n:=xmldom.item(cr_nl,j);
      cr_e:=xmldom.makeElement(cr_n);
      cr_tagName:=xmldom.getTagName(cr_e);
      if cr_tagName in ('DATA_PRIMERA_COMPRA') then
       if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            nlpos:=instr(xmldom.getNodeValue(cr_childNode),
                                                  chr(32));
            data_1compra:=substr(xmldom.getNodeValue
                                  (cr_childNode),1,nlpos-1);
         end if;
      elsif cr_tagName in ('DATA_DARRERA_COMPRA') then
       if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
```

```
            cr_childNode:=xmldom.getFirstChild(cr_n);
            nlpos:=instr(xmldom.getNodeValue(cr_childNode),
                                                  chr(32));
            data_ucompra:=substr(xmldom.getNodeValue
                                  (cr_childNode),1,nlpos-1);
         end if;
      elsif cr_tagName in ('IMPORT_ACUMULAT_COMPRES') then
       if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            import_compres:=xmldom.getNodeValue(cr_childNode);
         end if;
      elsif cr_tagName in ('NOMBRE_COMPRES') then
       if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            num_compres:=xmldom.getNodeValue(cr_childNode);
         end if;
      end if;
   end loop;

   xmldom.freeDocument(cr_doc);
   xmlparser.freeParser(cr_p);

   if data_1compra>data_ucompra then
      return (0);
   elsif data_ucompra>sysdate() then
      return (0);
   elsif import_compres >0 and num_compres<=0 then
     return (0);
   else
      return(1);
   end if;
end client_registrat_ok;
/
```

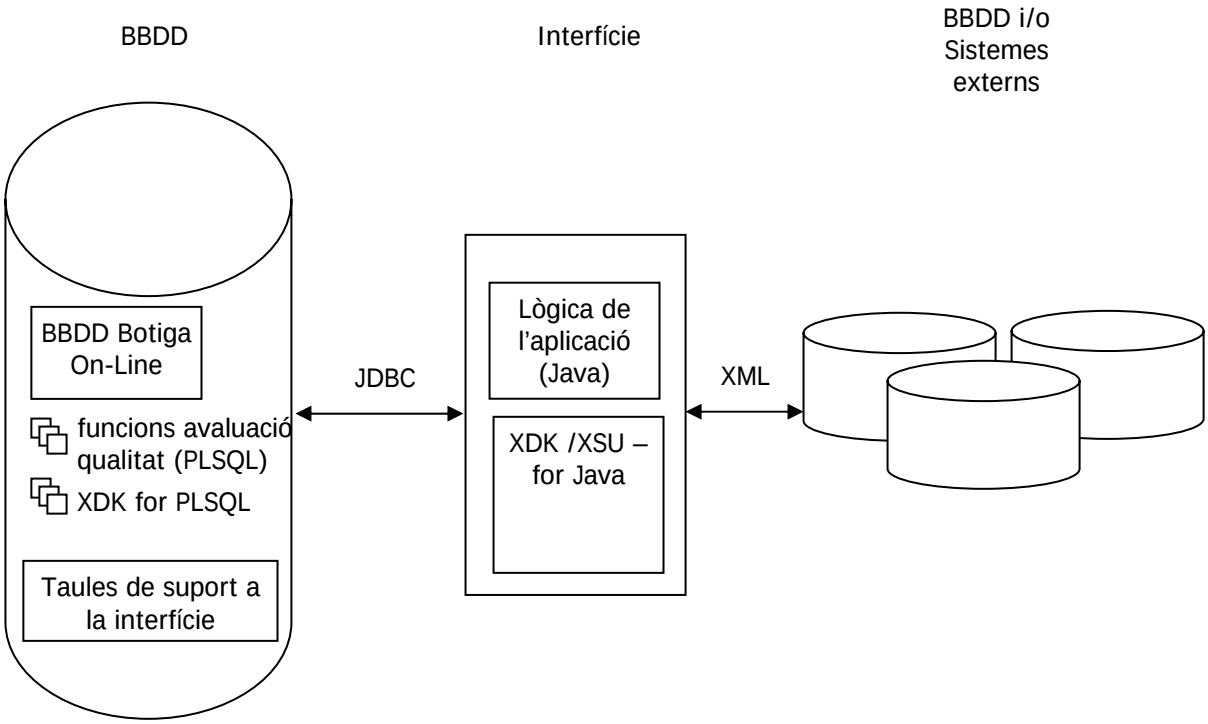
Per tal de no "contaminar" massa més aquesta part més descriptiva de la memòria, i donat que s'ha considerat que no aportava un major valor afegit a la descripció de la creació de les funcions de qualitat que actuen a nivell de registre, i vista la seva semblança i mida, s'ha decidit incloure la resta de funcions de qualitat que actuen a nivell de registre (juntament amb aquesta) a l'annex I d'aquesta memòria. Les funcions que es poden trobar en aquest annex són: `client_registrat_ok`, `producte_ok`, `comanda_ok`, `linia_comanda_ok`.

En aquest cas, en les funcions d'avaluació de l'existència d'errors a nivell de registre, i donat que com a paràmetre es passa el document XML contingut en el camp `xml` de la taula `XMLprev_tab`, es necessari utilitzar les utilitats XDK per a PL/SQL, i així permetre parserjar el document XML d'entrada.

4.3 Implementació de la Interfície

4.3.1 Arquitectura i algorisme principal

Un cop dissenyats per una banda els casos d'ús, el diagrama d'estats i el diagrama de seqüències, i per un altre costat dissenyat el sistema de control de qualitat i mostrada part de la seva implementació, és el moment de tornar a fer una mirada a l'arquitectura, incorporant-hi una mica més de detall:



Per a la implementació de la interfície, s'ha utilitzat el llenguatge de programació Java per implementar la lògica de l'aplicació, el qual utilitza el parser de XDK d'Oracle per a Java per al tractament dels fitxers d'entrada. Al seu torn, en la lògica de l'aplicació Java, també s'utilitza JDBC per realitzar accessos a les bases de dades, i als procediments (o funcions) emmagatzemats en aquestes.

Al seu torn, i com ja s'ha vist anteriorment s'ha programat amb llenguatge PL/SQL, les funcions d'avaluació pel control de qualitat, tant a nivell de camp, com a nivell de registre.

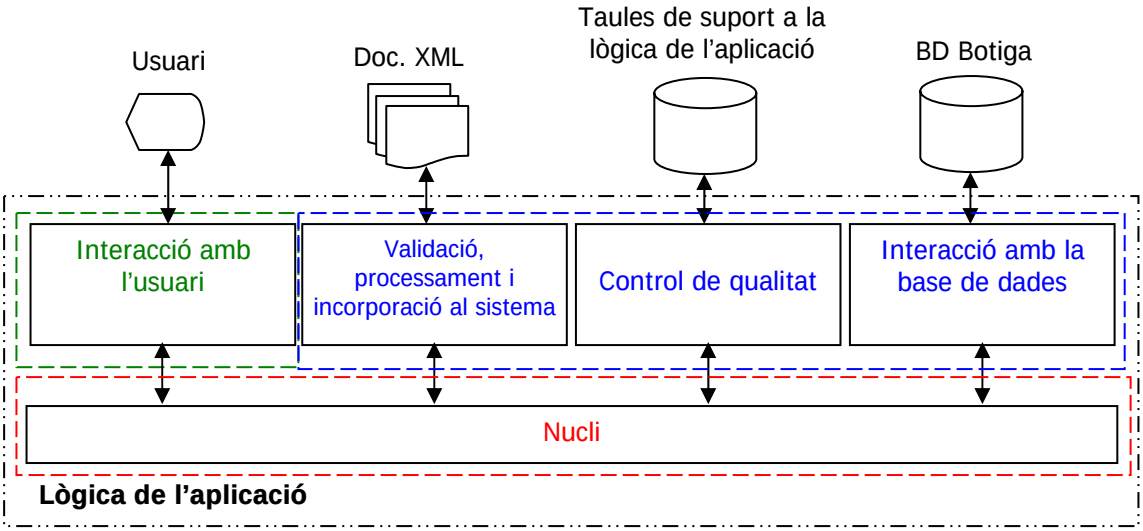
La importància en la definició d'aquesta arquitectura, radica en la seva modularitat. Així doncs, per un costat trobem la lògica de l'aplicació, la qual està programada en Java de manera que és portable a qualsevol sistema operatiu que sigui capaç d'executar una màquina virtual de Java.

Al seu torn, hi ha una clara separació entre la lògica de l'aplicació i les funcions dissenyades per avaluar el control de qualitat. D'aquesta manera s'aconsegueix una major independència de la lògica de l'aplicació de la interfície, i la base de dades de la Botiga On-Line. Així doncs, en el supòsit que es volgués migrar la interfície per utilitzar-la en una altra base de dades, només caldria tornar a programar, les

funcions d'avaluació de qualitat que actuen a nivell de registre (les funcions que actuen a nivell de camp no faria falta tocar-les, i com a molt s'haurien d'ampliar), sense tocar per a res la lògica de l'aplicació programada en Java. Cal recordar que en els documents XML dissenyats anteriorment, hi ha tota la informació necessària per tal d'accedir a les taules de la base de dades, cosa que permet que la lògica de l'aplicació no hagi de saber d'entrada que ha de fer, sinó que totes les accions s'interpreten dels documents XML i es porten a terme.

Tota la informació referent a la implementació de la part de la base de dades (taules de suport a la interfície, funcions emmagatzemades, etc.), ja ha estat descrita en els apartats anteriors. Ara per tant, cal dissenyar i implementar aquesta "caixa negra" que fins al moment s'ha anomenat "lògica de l'aplicació", i que ja hem decidit que s'implementarà en Java pels motius descrits anteriorment. Per tal de fer-ho, s'ha procurat també que sigui raonablement modular, per tal que pugui proveir al sistema d'un grau més de flexibilitat.

Recuperem per tant la representació en nivells de l'arquitectura de la interfície mostrada a l'inici d'aquest capítol, per tal de poder conceptualitzar millor el disseny i implementació de la lògica de l'aplicació:



Com es pot comprovar, s'ha modelat la lògica de l'aplicació en tres parts clarament diferenciades, i identificades amb els colors verd, blau i vermell.

- Assenyalat amb color verd, s'identifica la part de la lògica de l'aplicació que refereix a la interacció amb l'usuari, i en concret, amb els actors dels casos d'ús "Administrador" i "Processador de càrregues". D'aquesta manera, es pretén flexibilitzar la interfície fent que sigui més fàcil fer canvis en la seva presentació.
- Assenyalat amb color blau, s'identifica la part de la lògica de l'aplicació que interactua amb el sistema gestor de bases de dades. D'aquesta manera, es pretén flexibilitzar la interfície per tal que sigui més fàcil realitzar migracions que afectin al SGBD.
  Al seu torn, i també assenyalat de color blau, s'identifica la part de la lògica de l'aplicació que tracta i parseja els documents XML. Això s'ha fet així, ja que s'ha utilitzat les el XML Parser d'Oracle per tal de tractar i parsejar els documents, cosa que per tant, també es vincula d'alguna manera amb el SGBD escollit per a aquesta ocasió.

- Finalment, i assenyalat amb color vermell, s'identifica la part de la lògica de l'aplicació que coordina tots els processos de la mateixa. Com es pot comprovar per tant, aquesta part es totalment independent de la interacció amb l'usuari, del tractament dels documents XML, i fins i tot, del SGBD. Per tant, aquesta part no necessita cap canvi, fem el tipus de migració que fem.

Per a la implementació de la lògica de l'aplicació, i essent conseqüents amb aquesta estructura, s'ha creat 3 classes, de les quals es destaquen les dos més importants:

- Classe Interficie: Es tracta del cos principal de la lògica de l'aplicació (nucli). Interactua amb la classe menú per tal de admetre comandes escollides per l'usuari i coordina l'execució de les accions sol·licitades per aquest. Després es comenta l'algorisme de la funció main() definida en aquesta classe.
- Classe DocXMLdb: Conté el seguit de mètodes i funcions, que executen les accions sol·licitades per un objecte interfície.
    - Realitza les transformacions sobre els documents XML.
    - Es realitzen les consultes, insercions, reenviament i esborraments sobre les taules.
    - També coordina el sistema de control de qualitat.

Algorisme principal del processament de la interfície. Aquest algorisme, s'ha plantejat més com la seqüència d'esdeveniments que realitza el cos principal del processament de la interfície, que no pas com una especificació d'un pseudocodi formal.

```
funció principal() retorna (res)
   solicita l'entrada d'un usuari i password vàlids;
   crea una instància de la classe menúInterficie;
   Mentre no s'hagi sol·licitat sortir de l'aplicació ...
      Si sol·licita processar un nou enviament o reenviament.
         Sol·licita l'entrada del nom del fitxer;
         Invoca la precàrrega del fitxer;
         Si precàrrega sense errors
             s'executa el control de qualitat;
             Si no hi ha errors en el control de qualitat
               S'intenta carregar l'enviament a la BBDD;
             Sinó, si errors del control de qualitat són Lleu;
               També s'intenta carregar l'enviament a la BBDD;
               si s'ha realitzat amb èxit
                 carrega l'enviament a la taula XMLcrgerr_tab;
               fi_si;
             fi_si;
         fi_si;
         Si detectats errors lleus, i no carregat a la BBDD;
            carrega l'enviament a la taula XMLerr_tab;
         fi_si;
         Si s'han detectat errors crítics
            carrega l'enviament a la taula XMLFatal_err_tab;
         fi_si;
         Elimina el fitxer de la taula XMLprev_tab;
      Sinó, si sol·licita una altra operació
         l'executa;
      fi_si
   Fi_mentre;
Fi_funció;
```

Tal com he dit, no es tracta d'un algorisme formal sobre el processament del cos principal de la interfície, però denota les característiques principals del seu funcionament.

Tant la implementació de la classe Interficie, com la de els classes DocXMLdb i MenuInterficie, es poden consultar en l'annex II d'aquesta memòria.

4.3.2 Llenguatges de programació i eines utilitzades

Per tal de implementar la interfície, s'ha utilitzat els següents llenguatges de programació i eines (incloent també els llenguatges i eines utilitzats en la implementació del control de qualitat):

- Java 2 SDK versió 1.4.0
- PLSQL
- Oracle 9i XDK for Java
- Oracle 9i XDK for PLSQL
- Oracle JDBC Drivers release 8.1.7

Els següents editors:

- JCreator versió 2.0
- oXygen versió 3.1 (en període d'avaluació)
- MS-Editor "el Edit" (no pot faltar mai)
- Notepad (tampoc no pot faltar mai)

Sistema Gestor de Base de Dades (SGBD):

- Oracle 8i Personal Edition for Windows 98

4.3.3 Instal·lació

La instal·lació de fet, de tots el recursos necessaris per tal d'executar aquesta interfície, no té més complicació que anar executant els respectius "setup's" de cadascuna de les aplicacions, i anar confirmant totes les preguntes que els diversos assistents d'instal·lació ens van realitzant. Fins i tot, ens podem despreocupar força de l'ordre en que realitzem la instal·lació, tot i que es recomanable instal·lar en primer lloc el Java 2 SDK 1.4.0, després l'Oracle 8i Personal Edition for Windows 98, i finalment l'Oracle 9i XDK.

Hi ha una part però, en la configuració i instal·lació del sistema que no és tant trivial, essent l'explicació d'aquesta part, el motiu principal pel qual s'ha introduït aquest apartat. Es tracta de la càrrega del parser per a PL/SQL que disposa XDK, i gràcies al qual, es possible l'execució de les funcions que avaluen la qualitat dels enviaments a nivell de registre.

Així doncs, un cop es disposa de tot els sistema correctament instal·lat (Java 2 SDK, Oracle 8i i Oracle 9i XDK), haurem de seguir els següents passos:

1) Incorporar el parser PL/SQL a la base de dades
   En aquest cas, es necessari incorporar dos fitxers jar: `xmlparserv2.jar` i `xmlplsql.jar`, utilitzant el comandament loadjava, el qual normalment es troba en el directori `$ORACLE_HOME/bin`. Aquests fitxers contenen les classes Java utilitzades pel parser PLSQL XML.
   Així doncs, per exemple suposem que tenim creats a la base de dades l'usuari pfc amb el password pfc. La línia que haurem d'escriure per tal carregar els fitxers .jar que contenen les classes de Java, seran les següents:

```
% loadjava –user pfc/pfc –r –v xmlparserv2.jar
% loadjava –user pfc/pfc –r –v xmlplsql.jar
```

   Finalment, caldrà incorporar tots els procediments i funcions que permeten utilitzar el parser. El qual l'executarem des de l'eina SQL*Plus d'Oracle, prèviament havent entrat amb l'usuari pfc/pfc

```
% sqlplus pfc/pfc
```

   I posteriorment, executarem el següent comandament:

```
SQL>@xmlload
```

2) Finalment, és necessari dotar a l'usuari (en aquest cas pfc), per tal que tingui garantits els privilegis de seguretat de Java, per tal de poder interaccionar amb les classes que han estat carregades en l'anterior punt.
   Així doncs, ens connectarem a la base de dades amb un usuari que disposi de privilegis suficients (per exemple l'usuari sys o system), i farem l'assignació de permisos:

```
SQL> grant javauserpriv to pfc;
SQL> grant javasyspriv to pfc;
```

Fet això, ja podrem compilar i executar la interfície utilitzant el comandament:

```
% java Interficie
```

4.3.4 Execució i utilització de la interfície

Donat que la interfície no està dotada d'una presentació massa amigable, no s'ha fet cap captura de pantalla per tal de mostrar-ne la seva execució. Tot i així, s'ha cregut oportú fer algunes mínimes indicacions sobre el seu funcionament.

Així doncs, un cop executada la interfície (amb el comandament `java Interficie`), el primer que apareix, és un text que demana la introducció de l'usuari i el password. Aquesta introducció s'ha de fer sense espais, i separant l'usuari del password mitjançant el caràcter "/". Així doncs, un possible exemple és: pfc/pfc.

Cal tenir present que l'usuari i el password, no es validen fins que no es realitza alguna operació, per tant, en cas d'error, o en cas que requerim canviar d'usuari, es pot utilitzar l'opció (3) del menú d'usuari.

Un cop introduït per primer cop el usuari i el password, apareix el menú d'usuari, el qual mostra les següents opcions:

(1) Processar un nou enviament o Reprocessar un anterior
(2) Carregar configuració de qualitat
(3) Canviar User/Password
(4) Sortir

En primer lloc, i en cas que no s'hagi fet anteriorment, el primer que s'haurà de fer, és utilitzar l'opció (2), cosa que farà la carrega de la configuració del sistema de control de qualitat. Així doncs, haurem de tenir en el directori de treball un fitxer anomenat `CFG_DB.xml`, el qual ha de contenir un document XML amb l'especificació del sistema de qualitat de la interfície. El format d'aquest document XML, ha estat vist anteriorment.

Un cop identificats, i carregada la configuració del sistema de control de qualitat, ja es pot començar a processar enviaments. Aquesta acció, es realitza utilitzant l'opció (1) del menú.

Els diferents enviaments, han de trobar-se continguts en fitxers en el directori de treball, correctament identificats.

Cada cop que es desitja processar un enviament nou, ens demana el nom del fitxer, i en cas que aquest sigui vàlid, el llegeix del directori de treball i realitza el processament propi de la interfície. En els missatges de finalització de processament, es pot identificar si l'acció s'ha completat amb èxit, o altrament ha fallat, i si ha fallat, indica si l'error és lleu i s'ha pogut processar la informació amb la BBDD; o si l'error és lleu i no s'ha pogut processar la informació; o finalment, si l'error és crític.

En cas que un enviament hagi estat processat amb errors lleus o crítics, i com a conseqüència no s'hagi fer la càrrega a la base de dades (amb els enviaments amb errors lleus que si s'hagin carregat, no hi ha aquesta opció), es permet tornar a reenviar, i per tant reprocessar, el document XML per tal que siguin tornats a validar contra el sistema de control de qualitat, i contra la base de dades (si es dona el cas), un cop aquests hagin estat tractats (corregits) mitjançant l'eina de tractament d'enviaments fallits dissenyada en el següent capítol.

En aquest últim cas, no és necessari disposar del fitxer que contenia originalment el document XML. Com és lògic, la informació per a processar un reenviament s'extreu de les taules de suport a la interfície que contenen els enviaments fallits, i que mitjançant l'eina Web que després es mostra, poden ser corregits.

5. Tractament d'enviaments fallits

En els processos de càrrega i descarrega de documents XML a través de la interfície descrita en el capítol anterior, i gràcies al sistema de control de qualitat de que disposa aquesta interfície, sovint es poden detectar errors lleus o greus, que provoquen que la informació continguda en l'enviament no es pugui processar correctament en la base de dades de la Botiga On-line. En aquests casos, i tal com s'ha descrit en l'anterior capítol, la informació no es desestima, sinó que resta emmagatzemada en les taules de suport de la interfície segons el seu grau d'error, per tal que puguin ser tractats (corregits) posteriorment, i així puguin tornar a ser validats pel control de qualitat, i incorporats definitivament a la BBDD o identificats novament amb errors.

Si bé tots els processos propis de la interfície pel tractament dels documents XML (precàrrega, control de qualitat, processament de les operacions especificades en l'enviament a la BBDD, incorporació a les taules d'enviaments fallits, reenviament de documents XML continguts en les taules d'enviaments fallits, etc.), han estat dissenyats i mostrada la seva implementació en el capítol anterior, en aquest capítol, es pretén establir el disseny d'una eina que permeti l'edició i el tractament dels enviaments fallits per part de l'agent emissor.

Sovint, quan es detecta l'existència d'un enviament fallit l'administrador del sistema, o l'operari a qui s'hagi delegat la tasca, ha de notificar aquest fet a l'emissor per tal que aquest pugui esmenar l'error, ja sigui reenviant novament el document, o indicant a l'operari (o administrador) quines són les dades correctes per tal que la informació superi el control de qualitat i pugui ser processada a la BBDD sense errors. Aquest possible tractament dels enviaments fallits, comporta sovint pèrdues de temps importants, tant a l'emissor del document, com sobretot al receptor.

En aquests capítol es mostra el disseny d'una eina que permet estalviar els inconvenients derivats del tractament dels enviaments fallits (sobretot a la part receptora), i així mateix, dota d'autonomia a l'emissor a l'hora de identificar i corregir els enviaments fallits que hagi generat.

5.1 Objectius

En aquest cas, es tracta de dissenyar una eina complementària a la interfície, que permeti la identificació, edició i tractament (correcció) dels enviaments fallits, per part de la persona o empresa que els hagi originat. Aquesta eina per tant, ha de permetre l'accés als registres relacionats amb aquesta persona o empresa, de les taules de la interfície que donen suport al tractament dels enviaments fallits. En concret, es tracta de facilitar la identificació dels enviaments que aquest hagi realitzat amb errors, i l'edició d'aquests enviaments per tal que puguin ser corregits. Les taules que hauran de ser accessibles són:

- Taula que hostatja el contingut dels fitxers que no han estat processats a la base de dades, donat que s'ha considerat que contenien errors crítics.
- Taula que hostatja els documents XML que, tot i contenir errors lleus, no han estat processats a la base de dades.

- Taula que hostatja els documents XML que, tot i haver estat processats a la base de dades, s'ha detectat que contenien errors lleus.

En tot cas, cada emissor només podrà veure els enviaments fallits que aquest hagi realitzat.

Així doncs, els objectius a cobrir són:

- Definir l'abast de l'eina de tractament d'enviaments fallits
- Identificació de l'arquitectura sobre la que es realitza el disseny de l'eina pel tractament dels enviaments fallits.
- Disseny dels procediments PL/SQL Web.
- Disseny de les pantalles d'interacció amb l'eina.

5.2 Abast de l'eina de tractament d'enviaments fallits

Tal com s'ha comentat anteriorment, es tracta de dissenyar una eina complementaria a la interfície definida en el capítol anterior, que permeti la identificació, edició i tractament (correcció) dels enviaments fallits, per part de la persona o empresa que els hagi originat. D'aquesta manera es pretén facilitar les tasques de correcció dels enviaments que hagin resultat fallits, permetent a l'emissor dels mateixos l'accés a l'edició i correcció sense necessitar de la intervenció de la persona que ha rebut i processat els fitxers contra la interfície.

Les característiques principals d'aquesta eina, són:

- Independència de la base de dades contra la que actua la interfície (en aquest cas, la base de dades de la Botiga On-line).
- Independència del nucli de la interfície. L'eina de tractament dels enviaments fallits, només depèn de l'estructura de les taules que donen suport al tractament d'enviaments fallits de la interfície.
- La identificació, edició i tractament (correcció) dels enviaments fallits, s'ha de poder realitzar sense la intervenció de la persona que executa el tractament de l'enviament fallit contra la interfície.
- Ha de permetre la identificació, edició i tractament dels enviaments amb errors crítics.
- Ha de permetre també la identificació, edició i tractament dels enviaments amb errors lleus que no han estat carregats a la BBDD.
- Ha de permetre la identificació dels enviaments amb errors lleus que hagin estat carregats a la BBDD. En cap cas, no es permet l'edició i tractament d'aquests enviaments.

Tal com s'indica en aquest últim punt, només es permet la "identificació" dels enviaments amb errors lleus que hagin estat carregats a la BBDD, i en cap cas, no es permet l'edició i tractament (correcció) d'aquests enviaments. Això és així, per evitar conflictes o inconsistències entre les dades que han estat processades en la base de dades a través d'un enviament amb errors lleus, i les dades que resten en la taula "d'enviaments fallits carregats a la BBDD" relacionats amb l'enviament. Per tal de prendre aquesta decisió de disseny, s'ha tingut en compte el risc que pot suposar el fet de produir-se modificacions/eliminacions en els registres de la BBDD afectats per l'enviament amb errors lleus, que pugin produir la no coincidència entre aquests i les

dades emmagatzemades en la taula "d'enviaments fallits carregats en la BBDD". Per tal d'expressar millor aquest risc, es planteja un exemple concret:

> Suposem un enviament que pretén la inserció, entre altres, de les següents dades a la taula "Client_registrat" de la base de dades Botiga On-line:

NIF	...	data_darrera_compra	import_acumulat_compres	nombre_compres
43714998x		15/04/2004	1500,00	5

> Aquest enviament, seria processat per la interfície com un enviament amb errors lleus (per l'errònia introducció de la lletra del NIF, donat que el correcte és 43714998w), i per tant, es registraria l'enviament en la taula XMLCrgerr_tab, però permetria la inserció a la base de dades.
>
> Suposem que aquest "client_registrat" realitza una altra compra mitjançant els aplicatius propis del processament de vendes de la Botiga On-Line. Quedant el registre de la següent manera:

NIF	...	data_darrera_compra	import_acumulat_compres	nombre_compres
43714998x		20/05/2004	1750,30	6

> Donat que els aplicatius propis del processament de vendes de la Botiga On-Line, probablement no disposen de la capacitat d'actualitzar els registres de la taula XMLCrgerr_tab, es produeix una no coincidència entre les dades que conté la BBDD en explotació de la Botiga On-Line, i el registre emmagatzemat en la taula XMLCrgerr_tab.
>
> Així doncs, si es permetés la correcció i posterior reenviament del registre per part de la persona que ha realitzat l'enviament original, es podria perdre la informació relativa a l'última venda. Altrament, si es permetés la manipulació directa sobre les taules de la base de dades, per tal d'esmenar l'error, es podria perdre el control de la qualitat de la correcció.
>
> En aquest cas, es considera preferible que l'emissor pugui assabentar-se de l'error a través d'aquesta eina, i notificar la correcció a l'operador/administrador per tal que esmeni l'error de forma consensuada.

És evident que això no és un problema en el cas dels enviaments amb errors lleus o crítics no carregats a la base de dades, donat que al no haver-se processat la informació en la BBDD, no pot sofrir alteracions. Així doncs, en aquests casos l'eina de tractament d'enviaments fallits permetrà l'edició i tractament dels enviaments per a la seva correcció. Tal com s'ha descrit en el capítol anterior, la interfície disposa d'un mecanisme pel reenviament d'aquests registres corregits, de manera que es torna a avaluar l'enviament mitjançant el sistema de control de qualitat, i en determina la càrrega a la BBDD o el retorn a les taules amb errors, segons sigui el seu estat.

5.3 Disseny

5.3.1 Arquitectura

Per tal de permetre la identificació, edició i tractament (correcció) dels enviaments fallits, sense la intervenció de la persona que executa el tractament de l'enviament contra la interfície, s'ha previst el desenvolupament de l'aplicació (eina pel tractament d'enviaments fallits) en un entorn Web (WWW – World Wide Web), utilitzant el paquet PLSQL Web.

Això comporta avantatges importants a la persona que realitza els enviaments, donat que li permet tenir la informació en el moment que ho desitgi sobre els enviaments que ha realitzat i que han resultat fallits. Així mateix, podrà fer el tractament d'aquests enviaments fallits, sense requerir cap programa o instal·lació especial, donat que en tindrà prou amb un simple navegador Web.

El fet de dissenyar aquesta aplicació basant-la en un entorn web, també té avantatges per la banda servidora (receptora dels enviaments), entre les quals destaquen:

- *Lògica sempre al costat del servidor*: el PLSQL Web s'executa en el servidor web, en concret mitjançant el mòdul anomenat mod_plsql, cosa que significa que el pes dels processos de verificació en les transaccions corren al costat del servidor i no del client.
- *Instal·lació/actualització al costat del servidor*: al trobar-se tot el codi al costat del servidor, la instal·lació/actualització de l'aplicació es fa sempre en les màquines servidores, sense intervenció i molèstia als clients.
- *Independència de la plataforma*: Sigui quin sigui el sistema operatiu del clients, aquests podran disposar d'algun navegador que interpretarà l'aplicació. D'aquesta manera, el client no haurà de fer cap canvi per poder suportar el processament de l'aplicació, i així mateix, no s'hauran de desenvolupar diverses versions de la mateixa eina, per tal d'adaptar-se als diferents sistemes dels diferents clients.

PLSQL Web es basa en una barreja de dos llenguatges: PLSQL i HTML, i el seu funcionament es molt similar a altres llenguatges com ASP, JSP, PHP, ..., donat que tots en essència el que fan és generar pàgines HTML. Això facilita fins i tot, el fet de poder migrar de sistema de base de dades, donat que en el cas de canviar la base de dades Oracle, per una altra (per exemple, MS-SQL Server, PostgreSQL, MySQL, Informix, etc.), sempre es podria reprogramar la lògica de l'aplicació (de l'eina), sense transcendir els canvis al client.

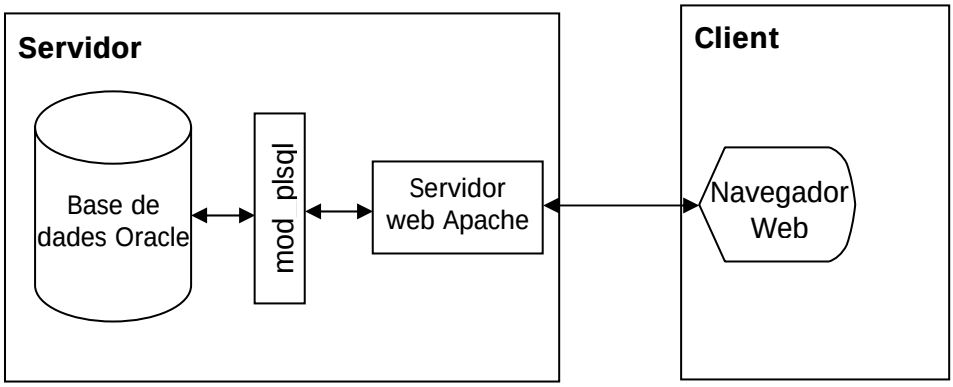
Per tal de desenvolupar un sistema PLSQL Web es necessiten com a mínim els següents components:

- Base de dades Oracle
- Servidor web Apache
- Modul PLSQL Web (mod_plsql, també anomenat PLSQL Gateway)
- Navegador web.

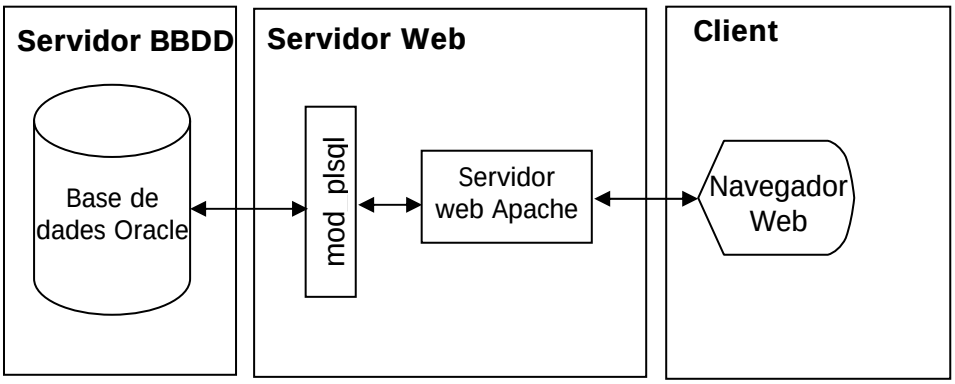
Els tres primers són la part servidor, és a dir, és la part que conté les dades i les presentacions web a les que el client accedirà mitjançant el quart component (navegador). Així doncs, es tracta d'una arquitectura client-servidor, el qual es pot definir mitjançant qualsevol dels següents dos esquemes:

- **En dos nivells:** En aquest cas, les dades (taules `XMLErr_tab`, `XMLFatal_err_tab` i `XMLCrgerr_tab` utilitzades per la interfície per emmagatzemar els enviaments fallits, segons el seu nivell d'error) i la lògica de l'aplicació (PLSQL Web) estan en un ordinador servidor i el client hi accedeix mitjançant la interfície (navegador web) de l'aplicació.

  D'aquesta manera, el client que disposa d'un navegador, no accedeix directament a la informació, sinó que ho fa mitjançant el servidor web, el qual mitjançant el mòdul mod_plsql recull els dades del servidor Oracle, essent el flux de dades bidireccional.



- **En tres nivells**: En aquesta configuració, les dades es troben en un ordinador servidor amb la base de dades, mentre que en un altre servidor es troba el servidor web; finalment, per un altre costat es troba el client que accedeix a l'aplicació mitjançant un navegador.



A l'hora de determinar quin ha de ser l'esquema escollit, cal valorar entre altres el nombre de transaccions amb la base de dades. Així doncs, si aquestes es preveu que no siguin molt elevades, el model a dos nivells és perfectament factible.

5.3.2 Disseny dels procediments PL/SQL Web

Per tal que l'eina pel tractament dels enviaments fallits, pugui desenvolupar les tasques que s'han descrit anteriorment, es dissenyen els següents procediments PLSQL Web:

- `consulta_enviaments_errorCritic`: Mostra la relació d'enviaments que han estat classificats per la interfície com a fallits amb errors crítics, que ha realitzat un emissor concret.
- `consulta_enviaments_errorLleu`: Mostra la relació d'enviaments que ha realitzat un emissor concret, i que han estat classificats per la interfície com a fallits amb errors lleus, i que no han pogut ser carregats a la base de dades.
- `consulta_enviaments_errorCrgLleu`: Mostra la relació d'enviaments que han estat classificats per la interfície com a fallits amb errors lleus, però que han pogut ser carregats a la base de dades, i que ha realitzat un emissor concret.
- `edita_enviament_errorCritic`: Permet l'edició del contingut d'un enviament que ha estat classificat per la interfície com a fallit amb errors crítics.
- `edita_enviament_errorLleu`: Permet l'edició del contingut d'un enviament que ha estat classificat per la interfície com a fallit amb errors lleus, i que no ha pogut ser carregat a la base de dades.
- `modificar_enviament_errorCritic`: Permet la modificació del contingut d'un enviament que ha estat classificat per la interfície com a fallit amb errors crítics, amb les dades obtingudes de l'edició de l'enviament (edició realitzada amb `edita_enviament_errorCritic`).
- `modificar_enviament_errorLleu`: Permet la modificació del contingut d'un enviament que ha estat classificat per la interfície com a fallit amb errors lleus i que no ha pogut ser carregat a la base de dades, amb les dades obtingudes de l'edició de l'enviament (edició realitzada amb `edita_enviament_errorLleu`).

El pseudocodi d'aquests procediments PLSQL Web, és el següent:

Algorisme del procediment `consulta_enviaments_errorCritic`:

```
Procediment consulta_enviaments_errorCritic és
   consulta número d'enviament de la taula XMLFatal_err_tab
                          relacionats amb l'emissor connectat;
   mostra cada número d'enviament resultant de la consulta com
                  un hiperenllaç a edita_enviament_errorCritic;
fi del procediment consulta_enviaments_errorCritic;
```

Algorisme del procediment `consulta_enviaments_errorLleu`:

```
Procediment consulta_enviaments_errorLleu és
   consulta número d'enviament i nivell d'error de la taula
               XMLErr_tab relacionats amb l'emissor connectat;
   mostra cada número d'enviament i nivell d'error resultant
                           de la consulta com un hiperenllaç
                           a edita_enviament_errorLleu;
```

```
fi del procediment consulta_enviaments_errorLleu;
```

Algorisme del procediment consulta_enviaments_errorCrgLleu:

```
Procediment consulta_enviaments_errorCrgLleu és
   consulta número d'enviament i nivell d'error de la taula
            XMLCrgerr_tab relacionats amb l'emissor connectat;
   mostra cada número d'enviament resultant i nivell d'error;
fi del procediment consulta_enviaments_errorCrgLleu;
```

Algorisme del procediment edita_enviament_errorCritic:

```
Procediment edita_enviament_errorCritic (num_enviament) és
   consulta registre de la taula XMLFatal_err_tab relacionat
               amb el num_enviament i amb l'emissor connectat;
   composa un formulari que referencia a
              modificar_enviament_errorCritic, amb una àrea de
              text que conté el cos del fitxer enviat;
fi del procediment edita_enviament_errorCritic;
```

Algorisme del procediment edita_enviament_errorLleu:

```
Procediment edita_enviament_errorLleu (num_enviament) és
   consulta document xml contingut en XMLErr_tab relacionat
               amb el num_enviament i amb l'emissor connectat;
   parseja el document xml;
   composa un formulari que referencia a
               modificar_enviament_errorLleu, amb una relació
               de tots els [element] [contingut], ambdós
               editables;
fi del procediment edita_enviament_errorLleu;
```

Algorisme del procediment modificar_enviament_errorCritic:

```
Procediment modificar_enviament_errorCritic (contingut de
                            l'àrea de text, num_enviament) és
   actualitza el contingut del registre de la taula
               XMLFatal_err_tab, relacionat amb el número
               d'enviament i amb l'emissor connectat;
   mostra una plana de confirmació de l'execució de
               l'operació, i un enllaç a l'inici;
fi del procediment modificar_enviament_errorCritic;
```

Algorisme del procediment modificar_enviament_errorLleu:

```
Procediment modificar_enviament_errorLleu (relacions [element]
                           [contingut], número d'enviament) és
   composar els "elements" amb els seus corresponents
               "continguts" en forma de document XML;
   actualitza el contingut del registre de la taula
               XMLErr_tab, relacionat amb el número
               d'enviament i amb l'emissor connectat;
   mostra una plana de confirmació de l'execució de
               l'operació, i un enllaç a l'inici;
```

```
fi del procediment modificar_enviament_errorLleu;
```

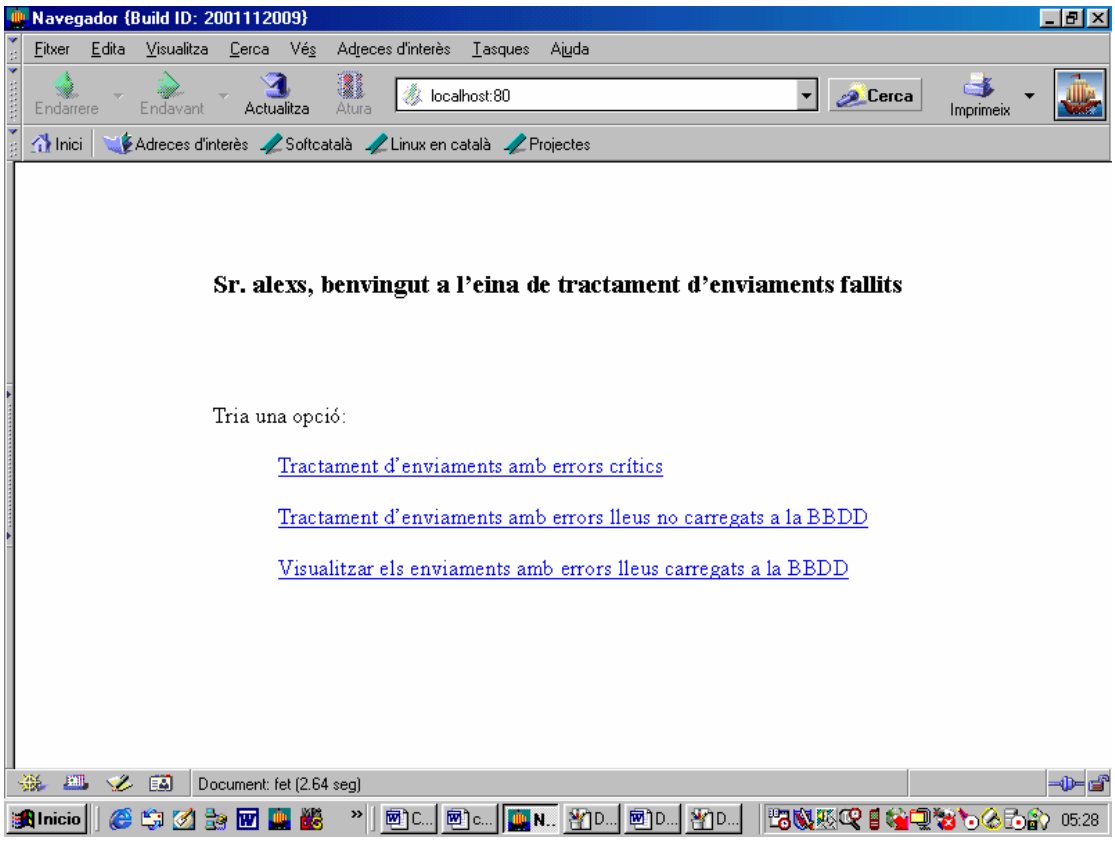
Com es pot observar en l'especificació d'aquests algorismes, en tot cas només s'accedeix a les taules XMLFatal_err_tab, XMLErr_tab o XMLCrgerr_tab, segons sigui el cas, i per tant, en cap cas no es fa cap accés a cap taula de la base de dades de la Botiga On-line. Així doncs, es pot afirmar que l'eina pel tractament dels enviaments fallits és totalment independent de la base de dades.

Al seu torn, aquestes taules no són manipulades directament pel nucli de la interfície (tal com es pot observar en el capítol anterior). De manera que també es pot afirmar que aquesta eina és independent del nucli de la interfície.

Finalment, i tal com es pot observar en els algorismes aquí presentats, l'eina dissenyada permet la identificació, edició i tractament dels enviaments amb errors crítics o amb errors lleus, que no han estat carregats a la BBDD, i també permet la identificació dels enviaments amb errors lleus carregats a la BBDD.

5.3.3 Disseny de les pantalles

Pantalla d'inici de l'eina pel tractament d'enviaments fallits:



Aquesta pantalla es mostra un com l'usuari ha superat la identificació i autenticació amb èxit.

Pantalla de consulta dels enviaments amb errors crítics:

En aquesta pantalla es mostra un exemple en el qual s'han identificat tres enviaments amb errors crítics, en concret, s'han identificat com a tal els enviaments 001, 006 i 028.

En cada cas, l'usuari pot punxar sobre qualsevol dels números d'enviaments, cosa que el conduirà a l'edició de l'enviament escollit.

Pantalla de consulta dels enviaments amb lleus no carregats a la BBDD:

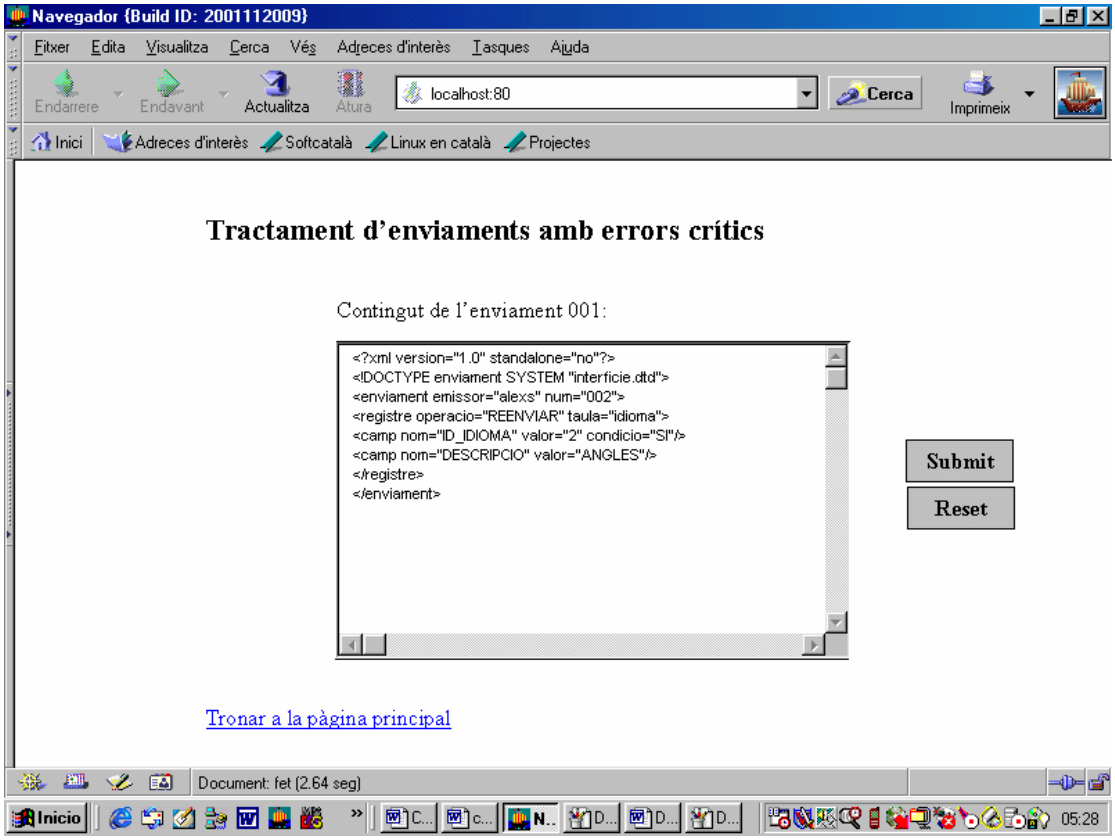
En aquesta pantalla es mostra un exemple en el qual s'han identificat tres enviaments amb errors lleus que no han estat carregats a la BBDD, juntament amb el nivell d'error produït en cada cas, en concret, s'han identificat com a tal els enviaments 004, 048 i 092.

En cada cas, l'usuari pot punxar sobre qualsevol dels números d'enviaments, cosa que el conduirà a l'edició de l'enviament escollit.

Pantalla de consulta dels enviaments amb lleus carregats a la BBDD:

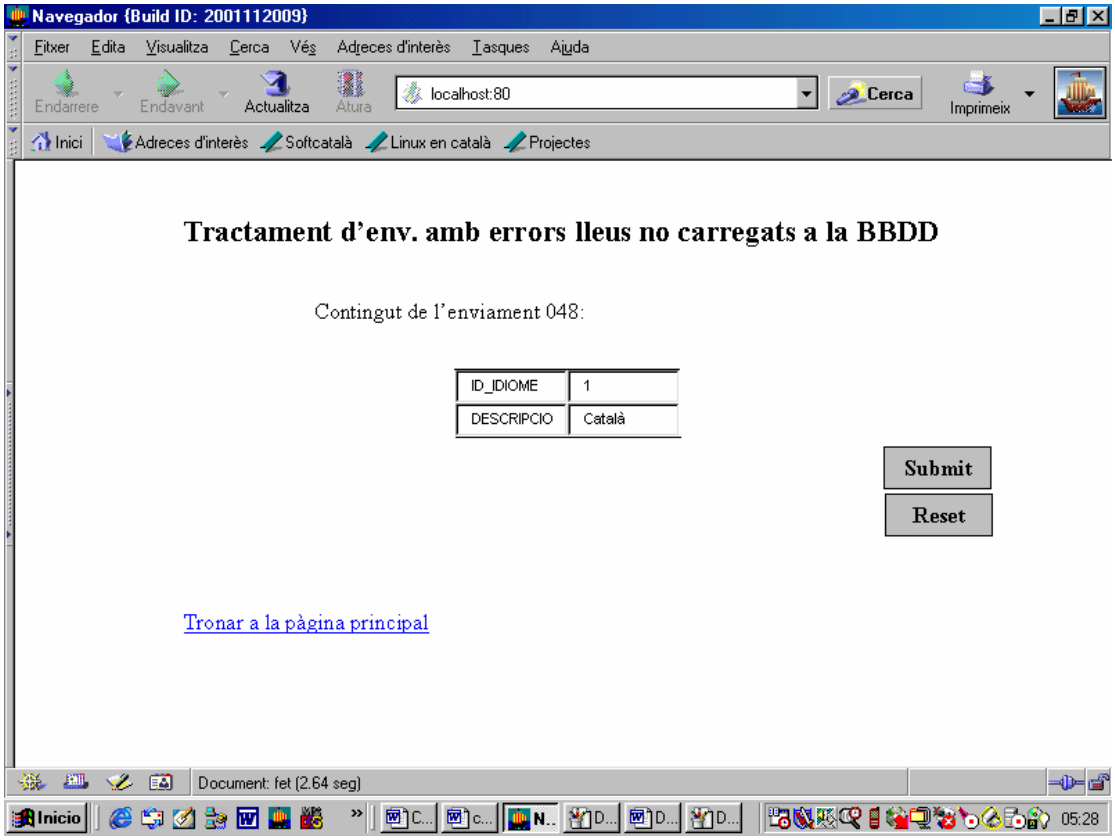
En aquesta pantalla es mostra un exemple en el qual s'han identificat dos enviaments amb errors lleus que han estat carregats a la BBDD, juntament amb el nivell d'error produït en cada cas, en concret, s'han identificat com a tal els enviaments 012 i 107.

Pantalla d'edició d'un enviament amb errors crítics:



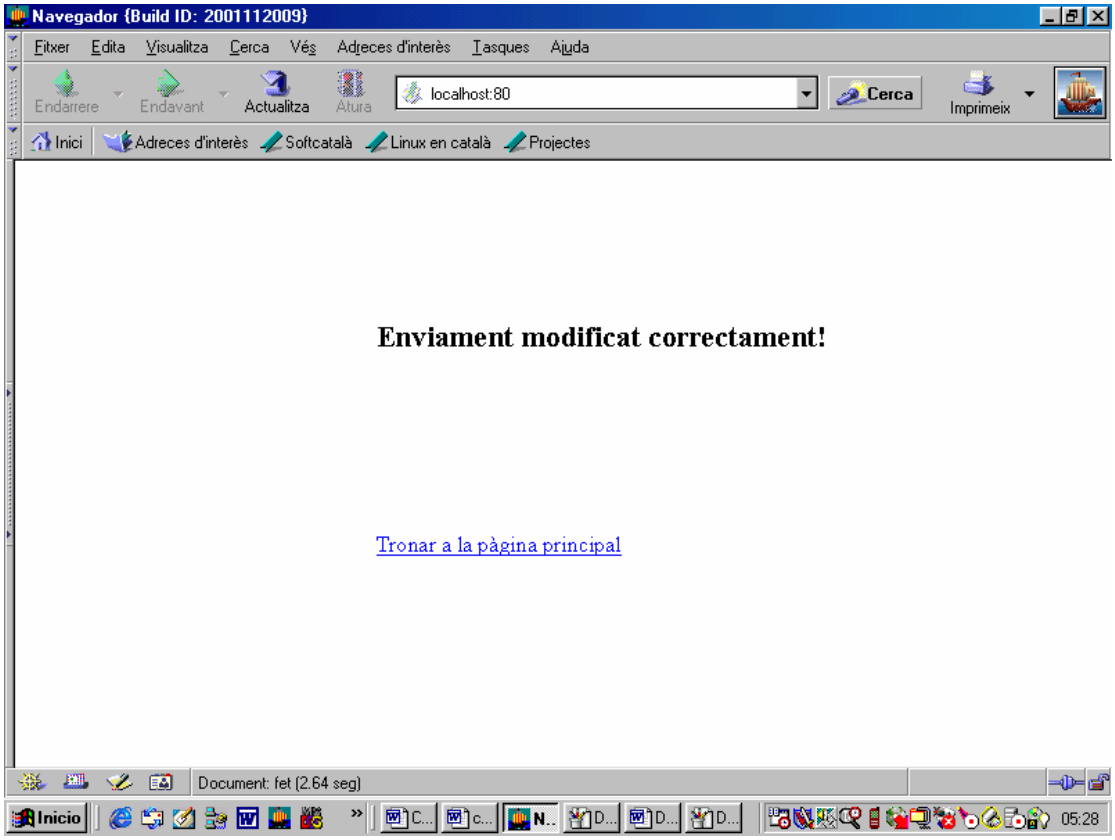
Com es pot observar en l'exemple exposat en el disseny d'aquesta pantalla, l'error en aquest enviament es va produir per què no es correspon el número de l'enviament especificat en el nom del fitxer (és el que la interfície escull com a número d'enviament vàlid en cas de conflicte, en aquest cas 001), amb el número d'enviament especificat en el cos del document XML (en aquest cas 002). Tal com s'ha indicat en el capítol anterior, aquest és un possible cas d'enviament amb errors crítics.

Pantalla d'edició d'un enviament amb errors lleus no carregat a la BBDD:



Com es pot observar en l'exemple exposat en el disseny d'aquesta pantalla, l'error en aquest enviament es va produir per què no es correspon el nom del camp de l'identificador de l'idioma (ID_IDIOME) amb el nom del camp a la BBDD (ID_IDIOMA).

Pantalla de confirmació d'una modificació:



Aquesta pantalla serà utilitzada tant per la confirmació de la modificació d'un enviament fallit amb errors crítics, com en el cas d'una modificació d'un enviament fallit amb errors lleus no carregat a la BBDD.

Conclusions

Com ja s'ha dit repetidament durant el desenvolupament d'aquesta memòria, l'intercanvi d'informació entre diferents sistemes, aplicatius, bases de dades, etc., i per tant, la definició dels formats dels enviaments que els han de fer possibles i el seu posterior tractament i processament en cadascun d'aquests sistemes, és un problemàtica que s'accentua a mesura que les necessitats d'intercanvi i integració de la informació en les relacions B2B i B2C van agafant un major protagonisme.

En mig de tota la barreja de format i sistemes que hom va desenvolupant de forma arbitrària (mitjançant l'establiment d'acords puntuals) per tal de gestionar aquests intercanvis d'informació, XML es mostra com una alternativa seriosa, oferint un meta-llenguatge que permet definir "llenguatges" de marcat adequats a usos determinats, i per tant, a definir formats per la representació de la informació a intercanviar d'una forma clara i validable.

A més, el conjunt d'eines i mecanismes que podem trobar sobre l'òrbita de XML, ens permeten la construcció d'interfícies flexibles, que no requereixen estar especialitzades amb el format de la informació a intercanviar.

Finalment, les pèrdues de temps i de diners que implica el processament manual d'aquesta informació, i el creixen tractament en línia (on-line) de les transaccions, requereixen d'una major integració de tots els fluxos d'informació, que no dona marge al tractament diferit d'aquesta, i menys en cara al seu processament amb errors. És per això que es fa necessària, la creació d'interfícies com la que aquí es presenta, que garanteixin la qualitat dels enviaments, i permetin en cas d'error, la identificació i possibilitat de correcció d'aquests amb la màxima celeritat i mínim esforç possible.

Així doncs, aquest projecte es situa en el centre neuràlgic d'una problemàtica que fa perdre molt temps i diners a les diferents empreses, administracions, etc. I ho fa, aportant una solució a aquest problema, àgil, portable i flexible, vàlida per ser incorporada en la major part dels sistemes d'informació actuals, amb un mínim esforç de reprogramació. I tot això gràcies a la modularitat, llenguatge de programació de la lògica de l'aplicació (Java), i a XML, que gràcies al seu format estructurat i extensible, permet definir tota classe de documents per a l'intercanvi d'informació.

Línies futures de treball

Durant el desenvolupament d'aquest projecte, s'ha fet referència bàsicament a la utilització de fitxers que contenien documents XML, on gràcies a la intervenció d'un operari/administrador, eren llegits del propi fitxer i processats a través de la interfície per ser carregats/descarregats al/del sistema. Això suposa per tant, un processament manual dels enviaments contra la interfície, que trenca la dinàmica d'integració total entre sistemes que persegueix l'esperit del projecte.

Essent així, una possible futura línia de treball, es podria centrar en aconseguir la integració total entre sistemes, establint mecanismes mitjançant els quals es permeti una comunicació contínua entre les diferents entitats que intervenen en l'intercanvi d'informació, cosa que es podria aconseguir a través de l'enviament de missatges correctament identificats. És així, que caldria analitzar les diferents alternatives que existeixen en al voltant d'XML, per tal d'oferir una solució sòlida a aquesta limitació.

Així mateix, i saltant a l'altre costat de la interfície (interacció de la interfície contra el SGBD), tot i les facilitats que ens proveeix JDBC per l'accés a la base de dades, i la solidés i eines que proveeix el SGBD escollit pel desenvolupament d'aquest projecte (Oracle), no deixa de ser una dificultat, el fet d'estar utilitzant un sistema que bàsicament entén d'SQL. Així, tots els processos que impliquen un intercanvi d'informació, requereixen sovint d'un processament i tractament previ dels documents XML que intervenen en l'intercanvi.

D'aquesta manera, tot i els avantatges que les característiques de XML proveeix, entre altres de flexibilitat de l'intercanvi de dades entre diferents sistemes d'informació, una última consideració necessària pel desenvolupament definitiu d'XML en les empreses, administracions, etc., és l'emmagatzematge de la informació. Així doncs, les atribucions de que disposa XML per l'intercanvi de dades depèn de les capacitats d'emmagatzematge de que disposi. XML no pot ser una opció seria de negoci per l'intercanvi d'informació, si no es pot emmagatzemar de forma eficient. La gran quantitat d'informació gestionada, fa que les conversions de format i l'emmagatzematge de dades en fitxers siguin solucions poc robustes i gens eficients.

La solució idònia per l'emmagatzematge de dades en format XML, passa per que aquesta pugui disposar de sistemes de gestió de la informació que siguin capaces de gestionar aquestes dades de forma nativa.

L'emmagatzematge de dades en format XML natiu disposa de diversos avantatges sobre la gestió realitzada per les bases de dades relacionals convencionals. Així doncs, no és necessària cap conversió de formats i l'estructura dels documents es manté. Els sistemes relacionals no estan pensats per la gestió jeràrquica de dades, que si disposa XML, per lo qual es necessari mecanismes de conversió a mida per l'emmagatzematge de la informació en aquest format. Altrament, els llenguatges de consulta típics com SQL tampoc no estan preparats per la búsqueda d'informació en sistemes XML.

Així doncs, una nova possible línia futura de treball, podria ser, per un costat dotar a la interfície d'una major independència, redissenyant tots els elements (funcions de qualitat, taules, etc.) creats per al suport a la interfície i al sistema control de

qualitat, per tal que operessin de forma independent al SGBD en el que es pugui trobar hostatjada la base de dades fent que aquests estiguin suportats en una base dades XML nativa. Finalment, i per un altre costat, es podria obrir també la possibilitat de que la interfície pugues operar de forma exclusiva contra bases de dades natives XML.

Glossari

arbre: Representació de l'estructura de dades que forma un document XML, en què un element (l'element d'arrel) conté la resta d'elements.

atribut: En general, paraula o la frase que s'adjunta a un substantiu per a qualificar-lo o especificar-lo; és una funció essencial dels adjectius. En XML, complement de l'arbre XML que conté informació terminal (que no es pot refinar posteriorment en XML).

Disseny conceptual: Etapa del disseny d'una base de dades que obté una estructura de la informació de la futura BD independent de la tecnologia que es vol emprar.

Disseny lógic: Etapa del disseny d'una base de dades que parteix del resultat del disseny conceptual i el transforma de manera que s'adapti al model de l'SGBD amb el qual es desitja implementar la base de dades.

document object model (DOM): Model o API (aplication program interface) per a referenciar parts d'un document HTML o XML co si fos una estructura de dades arborescent. Es pot utilitzar des de diversos llenguatges com Java, Javascript, C, C++, etc.

document type definition (DTD): Format heretat de l'SGML que permet d'expressar la sintaxi vàlida d'un document XML usant un llenguatge especial.

DOM: Vegeu document object model

DTD: Vegeu document type definition

EDI (Electronic Data Interchange): És un estàndard que proporciona un conjunt de formats de missatges per que les empreses puguin intercanviar les seves dades de forma automàtica, a través d'algun tipus de medi electrònic.

esquema: Vocabulari XML per a descriure les normes d'estructura, tipus de dades i valors, que s'apliquen a les dades estructurades que puguin contenir un document XML determinat.

estil: Mode, manera, forma. Es refereix a la separació entre les estructures de dades i la presentació. Com a resultat de la separació, és possible de canviar o associar amb facilitat diferents estils a la mateixa informació estructurada.

extensible markup language (XML): Vegueu llenguatge d'etiquetatge extensible.

JDBC (Java DataBase Connectivity): és l'API de Java que permet a les aplicacions interactuar directament amb motors de base de dades. JDBC inclou maneig de connexions a base de dades, execució de sentències SQL, suport de transaccions, etc. JDBC proveeix una interfície de programació única que independitza a les aplicacions del motor de la base de dades i del mecanisme de connexió.

llenguatge d'etiquetatge extensible: Especificació deseenvolupada pel Consorci Web (W3C), basat en l'SGML, un altre llenguatge de marques més complex. L'XML està dissenyat per conseguir dades estructurades i permet que la comunitat pugui definir les seves pròpies marques, això facilita la definició, transmissió, validació i interpretació de dades entre aplicacions i entre organitzacions.

llenguatge d'etiquetatge generalitzat estàndard (SGML): Llenguatge de marques publicat el 1986 per a expressar en un mateix document les dades i la seva estructura lògica. L'HTML i l'XML són subconjunts de l'SGML definits pel Consorci Web.

SAX: Vegeu simple API for XML.

SGBD: Sigla de Sistema de Gestió de Bases de Dades.

SGML: Vegeu llenguatge d'etiquetatge generalitzat estàndard.

simple API for XML (SAX): És un dels models de processament més comuns. Per utilitzar SAX per processar un document XML, s'ha de codificar mètodes per manejar esdeveniments llançats per l'analitzador segons troba diferents tokens del llenguatge de marques.

standard generalized markup language: Vegeu llenguatge d'etiquetatge generalitzat estàndard.

transformació: A partir d'una estructura de dades o arbre, fet d'obtenir-ne una altra d'equivalent, encara que de forma diferent, per mitjà d'una o de diverses operacions determinades.

UML: Vegeu Unified Modeling Language.

Unified Modeling Language (UML): És un model per a la construcció de programari orientat a l'objecte que l'OMG ha proposat com a estàndard d'ISO i que aquests darrers anys s'ha imposat a tot el món. L'UML constra d'un conjunt de tipus de diagrames interrelacionats que serveixen per a descriure diversos aspectes de l'estructura i la dinàmica del programari.

validar: Verificar la validesa d'un document davant l'especificació d'XML (vàlid) i eventualment davant l'especificació d'un vocabulari (DTD o esquema).

XML: Vegeu llenguatge d'etiquetatge extensible

XML Path Language from W3C (XPath): Utilitat que permet localitzar i extreure informació des d'un document XML.

XML stylesheet language (XSL): Llenguatge en el que estan escrits els fulls d'estil que possibiliten les transformacions de documents XML. El full d'estil XSL especifica com es mostraran les dades XML, mentre que XSLT utilitza les instruccions de formateig del full d'estil per realitzar la transformació.

XML stylesheet language transformations (XSLT): Fou definit pel grup de treball XSL de W3C, i descriu un llenguatge per transformar documents XML en altres documents XML o en altres formats.

XPath: Vegeu XML Path Language from W3C.

XSL: Vegeu XML stylesheet language

XSLT: Vegeu XML stylesheet language transformations

Bibliografia

Rusty E. (2001-2002). Processing XML with Java.

Rusty E. (2001-2002). XML Bible, second Edition.

Porfolto T. (1998). PL/SQL User's Guide and Reference. Oracle Corporation.

Karanjit S. Siyan (1997). Todo Visual J++. Editorial: InforBook's

Ramirez A.; Vanpeperstraete P.; Rueckert A.; Odutola K.; Bennett J.; Tolke L. (2000-2003). ArgoUML User Manual – A tutorial and reference description

Oracle 9i XML API Reference – XDK and Oracle XML DB Release 2 (9.2). url: download-west.oracle.com/docs/cd/B10501_01/appdev.920/a96616/toc.html

Merelo J.J. Tutorial de XSL: Generación de páginas Web usando XSLT y XML. url: http://geneura.ugr.es/~jmerelo/XSLT

Navarro J.M.: Iniciación a Oracle8. url: http://users.servicios.retecal.es/sapivi

Polo O. (2004). Introducción a PLSQL Web.

Manual del Curso de PL/SQL.

Java e Integración com Bases de Datos: La API JDBC.

Navarro L.; Marquès J.M. (2002). Apunts de l'assignatura d'Arquitectura de computadors II. Barcelona: Universitat Oberta de Catalunya.

Soriano M. (2002). Capítol 1 dels apunts de l'assignatura de Comerç electrònic. Barcelona: Universitat Oberta de Catalunya.

Alavedra O.; Rius M.A. (2002). Capítol 3 dels apunts de l'assignatura de Comerç electrònic. Barcelona: Universitat Oberta de Catalunya.

Cuenca A.; Torres H.; Grau A. Un nuevo marco para el intercambio electrónico de datos: XML/EDI. Granada: Universidad de Granada.

http://www.xml.com: Informació relacionada amb XML.

http://www.oracle.com: Informació relacionada amb Oracle

http://saxon.sourceforge.net: Informació relacionada amb el processador XSLT Saxon.

http://www.w3.org/XML: Informació relacionada amb XML.

http://www.programacion.com/java/tutorial/javaxml. Comparación de las Tecnologías Java para XML.

http://www.programacion.net/ . Apuntes de XML.

http://www.programacion.net/. APIs de Java para XML.

http://java.sun.com/j2se/1.4.2/docs/api/javax/xml

Annex I: Codi PLSQL de les funcions per l'avaluació del control de qualitat

```
-------------------------------------------------------------------------
-- Funcions d'avaluació del sistema de qualitat a nivell de registre
-------------------------------------------------------------------------
-- Funcions:
--          client_registrat_ok
--          producte_ok
--          comanda_ok
--          linia_comanda_ok
-------------------------------------------------------------------------
-- En tots els casos, es rep per paràmetre d'entrada el document XML
-- que conté la informació del registre, parseja i avalua el seu contingut
-- segons el cas, i retorna 1 o 0, en funció de si les comprovacions s'han
-- completat amb èxit o no.
-------------------------------------------------------------------------

-------------------------------------------------------------------------
-- Funció: client_registrat_ok
-- Descripció:
--  Parseja el document, i obté els valors corresponents a la data de la
--  primera compra, la data de la darrera compra, i l'import acumulat de
--  compres, juntament amb el número de compres. Avalua que aquestes dades ---
tinguin coherència entre elles, i amb la data del sistema, i si és així,
--  retorna un 1, en cas contrari, retorna un 0.
-------------------------------------------------------------------------
create or replace function client_registrat_ok(src_xmlFile in char) return integer is
   cr_p xmlparser.parser;
   cr_doc xmldom.DOMDocument;
   cr_nl xmldom.DOMNodeList;
   cr_len number;
   cr_n xmldom.DOMNode;
   cr_e xmldom.DOMElement;
   cr_tagName varchar2(30);
   cr_tagValue varchar2(100);
   cr_childNode xmldom.DOMNode;
   data_1compra date;
   data_ucompra date;
   import_compres float;
   num_compres int;
   nlpos int:=0;

begin
   -- Parseja el document
   cr_p:=xmlparser.newParser;
   xmlparser.setValidationMode(cr_p,FALSE);
   xmlparser.parseBuffer(cr_p,src_xmlFile);
   cr_doc:=xmlparser.getDocument(cr_p);

   cr_nl:=xmldom.getElementsByTagName(cr_doc, '*');
   cr_len:=xmldom.getLength(cr_nl);

   -- Recull els valors dels elements que ens interessa verificar
   for j in 0..cr_len-1 loop
      cr_n:=xmldom.item(cr_nl,j);
      cr_e:=xmldom.makeElement(cr_n);
      cr_tagName:=xmldom.getTagName(cr_e);
      if cr_tagName in ('DATA_PRIMERA_COMPRA') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            nlpos:=instr(xmldom.getNodeValue(cr_childNode),chr(32));
          data_1compra:=substr(xmldom.getNodeValue(cr_childNode),1,nlpos-1);
         end if;
      elsif cr_tagName in ('DATA_DARRERA_COMPRA') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
          cr_childNode:=xmldom.getFirstChild(cr_n);
          nlpos:=instr(xmldom.getNodeValue(cr_childNode),chr(32));
          data_ucompra:=substr(xmldom.getNodeValue(cr_childNode),1,nlpos-1);
         end if;
      elsif cr_tagName in ('IMPORT_ACUMULAT_COMPRES') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
```

```
            cr_childNode:=xmldom.getFirstChild(cr_n);
            import_compres:=xmldom.getNodeValue(cr_childNode);
         end if;
      elsif cr_tagName in ('NOMBRE_COMPRES') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            num_compres:=xmldom.getNodeValue(cr_childNode);
         end if;
      end if;
   end loop;

   xmldom.freeDocument(cr_doc);
   xmlparser.freeParser(cr_p);

-- Realitza les comprovacions oportunes i retorna el resultat de
-- la comprovació
   if data_1compra>data_ucompra then -- Si la data de la primera compra és
                                     -- superior a l'última
      return (0);   -- dades no correctes, i per tant retorna 0 (error)
   elsif data_ucompra>sysdate() then -- Si la data de l'última compra és
                                     -- superior a la d'avui
      return (0); -- dades no correctes, i per tant retorna 0 (error)
   elsif import_compres >0 and num_compres<=0 then
-- Si exist. import_compres, però no s'han fet compres...
     return (0); -- dades no correctes, i per tant retorna 0 (error)
   else
      return(1); -- Si no es produeix cap dels casos anterior, dades correctes.
   end if;

end client_registrat_ok;
/


--------------------------------------------------------------------------
-- Funció: producte_ok
-- Descripció:
--  Parseja el document, i obté els valors corresponents al preu del producte
--  quan aquest està d'oferta, el preu que té en l'actualitat, i l'indicador
--  de si el producte està en aquests moments d'oferta. Avalua que aquestes
--  dades tinguin coherència entre elles, i si és així, retorna un 1, en cas
--  contrari, retorna un 0.
--------------------------------------------------------------------------
create or replace function producte_ok(src_xmlFile in char) return integer is
   cr_p xmlparser.parser;
   cr_doc xmldom.DOMDocument;
   cr_nl xmldom.DOMNodeList;
   cr_len number;
   cr_n xmldom.DOMNode;
   cr_e xmldom.DOMElement;
   cr_tagName varchar2(30);
   cr_tagValue varchar2(100);
   cr_childNode xmldom.DOMNode;
   preu_actual float:=0;
   preu_oferta float:=0;
   es_oferta varchar2(1);
begin
   -- Parseja el document
   cr_p:=xmlparser.newParser;
   xmlparser.setValidationMode(cr_p,FALSE);
   xmlparser.parseBuffer(cr_p,src_xmlFile);
   cr_doc:=xmlparser.getDocument(cr_p);

   cr_nl:=xmldom.getElementsByTagName(cr_doc, '*');
   cr_len:=xmldom.getLength(cr_nl);

   -- Recull els valors dels elements que ens interessa verificar
   for j in 0..cr_len-1 loop
      cr_n:=xmldom.item(cr_nl,j);
      cr_e:=xmldom.makeElement(cr_n);
      cr_tagName:=xmldom.getTagName(cr_e);
      if cr_tagName in ('PREU_ACTUAL') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            preu_actual:=xmldom.getNodeValue(cr_childNode);
         end if;
      elsif cr_tagName in ('PREU_OFERTA') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
```

```
            cr_childNode:=xmldom.getFirstChild(cr_n);
            preu_oferta:=xmldom.getNodeValue(cr_childNode);
         end if;
      elsif cr_tagName in ('ES_OFERTA') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            es_oferta:=xmldom.getNodeValue(cr_childNode);
         end if;
      end if;
   end loop;

   xmldom.freeDocument(cr_doc);
   xmlparser.freeParser(cr_p);

-- Realitza les comprovacions oportunes i retorna el resultat de
-- la comprovació
   if es_oferta<>'S' and preu_actual>=preu_oferta then
      -- Si el producte no està d'oferta i el preu actual és superior al
      -- d'oferta
      return(1); -- les dades són correctes.
   elsif es_oferta='S' and preu_actual=preu_oferta then -- Si el producte està
d'oferta i el preu actual és igual al d'oferta...
      return(1); -- les dades són correctes.
   else
      return(0); -- altrament, les dades no són correctes
   end if;
end producte_ok;
/

--------------------------------------------------------------------------
-- Funció: comanda_ok
-- Descripció:
--  Parseja el document, i obté els valors corresponents a la data de lliurament
--  lliurament de la comanda, i a la data de la propia comanda. Avalua que
--  aquestes dades tinguin coherència entre elles, i si és així, retorna un
--  1, en cas contrari, retorna un 0.
--------------------------------------------------------------------------
create or replace function comanda_ok(src_xmlFile in char) return integer is
   cr_p xmlparser.parser;
   cr_doc xmldom.DOMDocument;
   cr_nl xmldom.DOMNodeList;
   cr_len number;
   cr_n xmldom.DOMNode;
   cr_e xmldom.DOMElement;
   cr_tagName varchar2(30);
   cr_tagValue varchar2(100);
   cr_childNode xmldom.DOMNode;
   data_lliurament date;
   data_comanda date;
   nlpos int:=0;
begin
   -- Parseja el document
   cr_p:=xmlparser.newParser;
   xmlparser.setValidationMode(cr_p,FALSE);
   xmlparser.parseBuffer(cr_p,src_xmlFile);
   cr_doc:=xmlparser.getDocument(cr_p);

   cr_nl:=xmldom.getElementsByTagName(cr_doc, '*');
   cr_len:=xmldom.getLength(cr_nl);

   -- Recull els valors dels elements que ens interessa verificar
   for j in 0..cr_len-1 loop
      cr_n:=xmldom.item(cr_nl,j);
      cr_e:=xmldom.makeElement(cr_n);
      cr_tagName:=xmldom.getTagName(cr_e);
      if cr_tagName in ('DATA_LLIURAMENT') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            nlpos:=instr(xmldom.getNodeValue(cr_childNode),chr(32));
            data_lliurament:=
                        substr(xmldom.getNodeValue(cr_childNode),1,nlpos-1);
         end if;
      elsif cr_tagName in ('DATA_COMANDA') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
          cr_childNode:=xmldom.getFirstChild(cr_n);
          nlpos:=instr(xmldom.getNodeValue(cr_childNode),chr(32));
          data_comanda:=substr(xmldom.getNodeValue(cr_childNode),1,nlpos-1);
```

```
         end if;
      end if;
   end loop;

   xmldom.freeDocument(cr_doc);
   xmlparser.freeParser(cr_p);

-- Realitza les comprovacions oportunes i retorna el resultat de
-- la comprovació
   if data_lliurament<data_comanda then
      return(0);
   else
      return(1);
   end if;
end comanda_ok;
/


--------------------------------------------------------------------------
-- Funció: linia_comanda_ok
-- Descripció:
--  Parseja el document, i obté els valors corresponents al preu unitari
--  brut del producte, al descompte que sobre aquest i pugui haver-hi, i el
--  preu net resultant. Avalua que aquestes dades tinguin coherència entre
--  elles, i si és així, retorna un 1, en cas contrari, retorna un 0.
--------------------------------------------------------------------------
create or replace function linia_comanda_ok(src_xmlFile in char) return integer is
   cr_p xmlparser.parser;
   cr_doc xmldom.DOMDocument;
   cr_nl xmldom.DOMNodeList;
   cr_len number;
   cr_n xmldom.DOMNode;
   cr_e xmldom.DOMElement;
   cr_tagName varchar2(30);
   cr_tagValue varchar2(100);
   cr_childNode xmldom.DOMNode;
   preu_brut float:=0;
   preu_net float:=0;
   dte float:=0;
   temp float:=0;
begin
   -- Parseja el document XML
   cr_p:=xmlparser.newParser;
   xmlparser.setValidationMode(cr_p,FALSE);
   xmlparser.parseBuffer(cr_p,src_xmlFile);
   cr_doc:=xmlparser.getDocument(cr_p);

   cr_nl:=xmldom.getElementsByTagName(cr_doc, '*');
   cr_len:=xmldom.getLength(cr_nl);

   -- Recull els valors dels elements que ens interessa verificar
   for j in 0..cr_len-1 loop
      cr_n:=xmldom.item(cr_nl,j);
      cr_e:=xmldom.makeElement(cr_n);
      cr_tagName:=xmldom.getTagName(cr_e);
      if cr_tagName in ('PREU_UNITARI_BRUT') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            preu_brut:=xmldom.getNodeValue(cr_childNode);
         end if;
      elsif cr_tagName in ('PREU_NET') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            preu_net:=xmldom.getNodeValue(cr_childNode);
         end if;
      elsif cr_tagName in ('DTE') then
         if xmldom.isNull(xmldom.getFirstChild(cr_n))=FALSE then
            cr_childNode:=xmldom.getFirstChild(cr_n);
            dte:=xmldom.getNodeValue(cr_childNode);
         end if;
      end if;
   end loop;

   xmldom.freeDocument(cr_doc);
   xmlparser.freeParser(cr_p);
```

```
   -- calcula el valor que hauria de tenir el preu net resultant d'aplicar el
descompte
   temp:=preu_brut*((100-dte)/100);
   -- comprova que siguin iguals
   if temp=preu_net then
      return(1); -- si és així, retorna un 1
   else
      return(0); -- en cas contrari, retorna un 0
   end if;
end linia_comanda_ok;
/

-------------------------------------------------------------------------
-- Funcions d'avaluació del sistema de qualitat a nivell de camp
-------------------------------------------------------------------------
-- Funcions:
--          es_nif
--          es_numeric
--          no_nul
--          es_data_anterior_avui
--          es_tant_per_cent
-------------------------------------------------------------------------
-- En tots els casos, es rep per paràmetre d'entrada el contingut del camp
-- a verificar, i es comprova que compleixi les condicions de la funció.
-- com a resultat, retorna 1 o 0, en funció de si les condicions es
-- compleixen o no.
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-- Funció: es_nif
-- Descripció:
--  Comprova que la lletra del nif (passat per paràmetre) sigui correcta.
-------------------------------------------------------------------------
create or replace function es_nif (nif in char) return integer is
   lleNif char(1);
   charNif varchar2(8);
   numNif INTEGER;
   caracters varchar2(24):='TRWAGMYFPDXBNJZSQVHLCKET';
   pos INTEGER:=0;
begin
      lleNif:=upper(substr(nif,9,1));
      charNif:=substr(nif,0,8);
      numNif:=to_number(charNif);
      pos:=(numNif mod 23)+1;

      if lleNif=substr(caracters,pos,1) then
         return (1);
      else
         return (0);
      end if;

end es_nif;
/

-------------------------------------------------------------------------
-- Funció: es_numeric
-- Descripció:
--  Comprova que el valor passat per paràmetre sigui de tipus numèric.
-------------------------------------------------------------------------
create or replace function es_numeric (val in char) return integer is
   id numeric;

   begin
      id:=val;
      return(1);
   exception
      when value_error then
         return(0);
end es_numeric;
/

-------------------------------------------------------------------------
-- Funció: no_nul
-- Descripció:
--  Comprova que el valor passat per paràmetre no sigui nul.
-------------------------------------------------------------------------
create or replace function no_nul (val in char) return integer is
```

```
   begin
      if length(val)>0 then
         return(1);
      else
         return(0);
      end if;
end no_nul;
/

--------------------------------------------------------------------------
-- Funció: es_data_anterior_avui
-- Descripció:
--  Comprova que la data passada per paràmetre sigui anterior a la de sistema.
--  És de suposar que la data del sistema es correcta, i correspon amb la d'avui.
--------------------------------------------------------------------------
create or replace function es_data_anterior_avui (val in char) return integer is
   nlpos int:=instr(val,chr(32));
   data date:=substr(val,1,nlpos-1);
   begin
      if data>sysdate() then
         return(0);
      else
         return(1);
      end if;
end es_data_anterior_avui;
/

--------------------------------------------------------------------------
-- Funció: es_tant_per_cent
-- Descripció:
--  Comprova que el valor passat per paràmetre, sigui superior a 0 i inferior
--  a 100.
--------------------------------------------------------------------------
create or replace function es_tant_per_cent (val in char) return integer is
   tpc float :=val;
   begin
      if tpc>=0 and tpc<100 then
         return(1);
      else
         return(0);
      end if;
end es_tant_per_cent;
/
```

Annex II: Codi Java de la implementació de la interfície

Interfície.java

```
import java.io.*;
import java.net.*;
import org.w3c.dom.*;
import org.w3c.dom.Node;
import java.sql.*;
import oracle.xml.parser.v2.*;

/**
* Classe: Interficie
*   Cos principal de la lògica de l'aplicació. Interactua amb la
*   classe menú per tal d'admentre comandes escollides per l'usuari
*   i coordina l'execució de les accions sol.licitades per aquest.
***/
public class Interficie
{
   static int MAX_ERROR=10; // Nivell màxim per que un error sigui consideratlleu
   static int FATAL_ERROR=100; // Nivell màxim d'error. Error crític
   static int NO_ERROR=0; // No existència d'error.
   private static DocXMLdb document = new DocXMLdb();

   static public void main(String[] argv)
   {

      char opcio='0';
      String nomFitxer="";
      int nError=0;

      BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

      document.usr_Pass(); // solicita l'entrada d'un usuari i password vàlids

      MenuInterficie l = new MenuInterficie(); //defineix "l" com a objecte de la
                                               //classe MenuInterficie.

      while (opcio!='4') //Mentre no s'hagi sol.licitat sortid de l'aplicació ...
         {
            opcio=l.menuInter();
            if (opcio=='1') // Si es sol.licita el processament d'un nou enviament.
               {
                  System.out.println("Indica el nom del fitxer:");
                  try {
                  nomFitxer=in.readLine(); // Sol.licita l'entrada del nom del fitxer.
                  } catch(java.io.IOException ioex) {
                    System.out.println("Error en llegir el nom del fitxer");
                  }
                  if (nomFitxer.length()!=12)
                     {
                        System.out.println("Error en el nom del fitxer");
                     }
                  else
                     {
                        //Invoca la precàrrega del fitxer
                        nError=document.preCarregarFitxer(nomFitxer);
                       // Si es realitza sense errors,
                       //... s'executa el control de qualitat.
                        if (nError==NO_ERROR)
                           {
                              try{
                              //moment en que executa el control de qualitat
                              nError=
                                document.Pqualitat(nomFitxer.substring(0,5),
                                nomFitxer.substring(5,8));
                              }
```

```
      catch(SQLException e)
      {
      System.out.println(e);
      nError=FATAL_ERROR;
      }
      if (nError==NO_ERROR) // si no hi ha hagut errors en el
                            //control de qualitat...
         {
           try {
           //S'intenta carregar l'enviament a la BBDD
           nError=document.carregarFitxer(nomFitxer);
           System.out.println("Enviament OK!");
           }
           catch(SQLException e) {
           nError=FATAL_ERROR;
           }
         }
         // si el nivell d'errors del control de qualitat és
         //lleu ...
       else if (nError<=MAX_ERROR)
         {
           int error=nError;
           try {
           // ...també s'intenta carregar l'enviament a la
           // BBDD ...
           nError=nError+document.carregarFitxer(nomFitxer);
           }
           catch(SQLException e) {
           nError=FATAL_ERROR;
           }
           if (error==nError && nError>NO_ERROR)
              {
                 try {
                 // ... i si s'ha realitzat amb exit, carrega
                 // l'enviament a la taula XMLcrgerr_tab
                 document.
                   inserta_en_crgtError_tab(nomFitxer,nError);
                 System.out.println("Enviament amb errors
                                  lleus, Carregat a la BBDD");
                 }
                 catch(SQLException e) {
                 nError=FATAL_ERROR;
                 }
                 //Es considera que no hi ha errors, donat que
                 // el processament ja ha finalitzat.
                 nError=NO_ERROR;
              }
         }
   }
// Si s'han detectat errors lleus, i no s'ha pogut carregar
// l'enviament a la BBDD ...
if (nError>NO_ERROR && nError<MAX_ERROR)
   {
      try {
      //... s'inserta l'enviament a la taula XMLerr_tab
      document.inserta_en_Error_tab(nomFitxer,nError);
      System.out.println("Enviament amb errors lleus, NO
                                 carregat a la BBDD");
      }
      catch(SQLException e) {
      System.out.println("Error insertar_en_Error_tab: "+e);
      nError=FATAL_ERROR;
      }
   }
if (nError>MAX_ERROR) // Si s'han detectat errors crítics ...
   {
     try{
     if (!document.env_inFatal_err_tab(
           nomFitxer.substring(0,5),nomFitxer.substring(5,8)))
       //... s'inserta l'enviament a la taula XMLFatal_err_tab
        document.crg_Error_Fatal(nomFitxer);
     System.out.println("Enviament amb errors Crítics. NO
                                    carregat a la BBDD");
     }
     catch(SQLException e)
     {
     System.out.println(e);
```

```
                              }
                              catch(IOException e)
                              {
                              System.out.println(e);
                              }
                            }
                        try {
                        // Finalment, s'elimina el fitxer de la taula XMLprev_tab;
                        document.esborrarXMLPrev(nomFitxer);
                        }
                        catch(SQLException e) {
                        System.out.println("Error esborrarXMLPrev: "+e);
                        }
                     }
               }
            // Si es sol·licita la carrega de la informació de configuració
            if (opcio=='2')
               {
                  try{
                    // sol·licita el procés de càrrega de la configuració del sis. de
                    // Control de Qualitat
                    document.cfg_DB();
                  }
                  catch(SQLException e)
                     {
                        System.out.println(e);
                     }
               }
            if (opcio=='3') // Si es sol·licita un canvi d'usuari/password
               {
                   document.usr_Pass();
               }
            if (opcio=='4') // Si es sol·licita sortid de l'aplicació
               {
                   System.out.println("Adeu");
               }
         }

   }


}


/**
* Classe: MenuInterficie
*    Definida pel maneig del menú d'usuari.
***/
class MenuInterficie {

    /**
    * menuInter
    * Paràmetres: cap
    * retorna: char - opció
    *   Mostra el menú, i retorna l'opció demanada.
    **/
    public char menuInter() {
       char opcio='0';
       BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
       System.out.println("(1) Processar un nou enviament o Reprocessar un
                                                                    anterior\n"+
                          "(2) Carregar configuració de qualitat\n"+
                          "(3) Canviar User/Password\n"+
                          "(4) Sortir \n\n" +
                          "Tria una opció:");
       try {
       opcio=in.readLine().charAt(0);
       } catch(java.io.IOException ioex) {
          System.out.println("Error en triar opció");
       }

       if (opcio!='1' && opcio!='2' && opcio!='3' && opcio!='4')
          System.out.println("Opció no vàlida!!!");
        return opcio;
   }
}
```

DocXMLdb.java

```
import java.io.*;
import java.net.*;
import org.w3c.dom.*;
import org.w3c.dom.Node;
import java.sql.*;
import oracle.xml.parser.v2.*;
import oracle.xml.sql.dml.*;
import oracle.xml.sql.query.*;
import oracle.jdbc.driver.*;
import oracle.xml.parser.plsql.*;

/**
* Classe: DocXMLdb
*   Conté el seguit de mètodes i funcions, que executen les accions
*   sol·licitades per un objecte interfície.
*   - Realitza les transformacions sobre els documents XML.
*   - Es realitzen les consultes, insercions, reenviament i esborraments
*     sobre les taules.
*   - També coordina el sistema de control de qualitat.
**/
public class DocXMLdb
{
   private XMLDocument doc;
   private static int FATAL_ERROR=100; // Nivell màxim d'error. Error crític
   private static int MAX_ERROR=10; // Nivell màxim de un error considerat com a lleu
   private static int SAV_ERROR=7;  // Nivell d'error en produir-se una excepció amb
                                    // el processament d'un registre amb la BBDD
   private static int NO_ERROR=0; // No existència d'error.
   private static String TENDA="tenda"; //primers 5 caràcters utilitzats per
                                        // identificar el nom del fitxer per les
                                        // DESCARREGUES
   private String usr_pass="";

   public DocXMLdb()
      {
      }


   /**
   * DOMcarrega
   * Paràmetres:nom_fitxer - nom del fitxer que conté la informació de l'enviament
   * Retorna: int - nivell d'error
   *  Llegeix el fitxer XML, i determina si aquest és un document XML vàlid.
   *  Al seu torn, l'incorpora al sistema com un DOM Document, per que posteriorment,
   *  es pugui treballar amb aquest.
   ***/
   private int DOMcarregar(String nom_fitxer)
   {
      int error=0;
      try
      {

         DOMParser parser = new DOMParser();
         URL url = createURL(nom_fitxer);

         parser.setErrorStream(System.err);
         parser.setValidationMode(true);
         parser.showWarnings(true);

         parser.parse(url);

         doc = parser.getDocument();

      }
      catch (Exception e)
      {
         error=FATAL_ERROR;
      }
      finally{
        return (error);
      }
   }
```

```
/**
* validar_nomenviament
* Paràmetres: nomf - nom del fitxer que conté el document XML amb la informació
*                      de l'enviament
* Retorna: boolena - Cert (si la comprovació és correcta)
*                    Fals (la la comprovació resulta erronia)
*   Comprova que el nom del fitxer es correspongui amb els atributs
*    de l'element "enviament".
***/
private boolean validar_nomenviament(String nomf)
   {
      NodeList nl = doc.getElementsByTagName("enviament");
      Element e;
      NamedNodeMap nnm;
      String emisor;
      String num_enviament;
      Node n;
      String nomc;

      e=(Element)nl.item(0);
      nnm = e.getAttributes();

      n = nnm.item(0);
      emisor = n.getNodeValue();
      n = nnm.item(1);
      num_enviament = n.getNodeValue();

      nomc=emisor.concat(num_enviament).concat(".xml");
      if (nomc.equals(nomf))
         return(true);
      else
         return(false);
   }

/**
* crg_Error_Fatal
* Paràmetres: nomf - Nom del fitxer que conté la informació referent a l'enviament
* Retorna: res
*  La seva funció es insertar el contingut del fitxer (de forma literal, i sense
*  cap altre tractament), en la taula XMLfatal_err_tab, correctament identificat
*  a partir del nom del fitxer. Això és així, donat que no es pot fiar del
*  contingut del fitxer.
***/
public void crg_Error_Fatal(String nomf) throws SQLException, IOException
   {
     InputStream fitxer = new FileInputStream(nomf);
     String contingut="";
     int caracter;
     char ch;

     //Taula sobre la que guardarà la informació
     String nom_taula="XMLfatal_err_tab";
     //primera part de l'ident. de l'env. Primers 5 caràcters del nom del fitxer
     String nom_emissor=nomf.substring(0,5);
     // segona part de l'ident. de l'env. Caràcters de 6è al 8è del nom del fitxer
     String num_enviament=nomf.substring(5,8);

     // Lectura caràcter a caràcter del fitxer.
     try
     {
        while ((caracter=fitxer.read())!=-1)
        {
         ch=(char) caracter;
         contingut=contingut.concat(Character.toString(ch));
        }
     }
     catch(IOException e)
     {
         System.out.println(e);
     }
     finally
     {
         fitxer.close();
     }


     // Llegeix el Oracle JDBC driver
```

```
      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

      // estableix una connexió
      Connection conn =
        DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");

      // Crea un Statement
      Statement stmt = conn.createStatement ();

      // realitza la inserció a la taula XMLfatal_err_tab
      ResultSet rset = stmt.executeQuery ("insert into "+ nom_taula+" (nom_emissor,
                           num_enviament, xml) values('"+nom_emissor+"','" +
                           num_enviament + "','"+contingut+"')");
      rset.close();
      stmt.close();
      conn.commit();
      conn.close();
   }

/**
* crg_Previ
* Paràmetres: cap
* Retorna: res
* Realitza la transformació del document XML vàlid, incorporat anteriorment
* al sistema com a DOM Document, en un altre document XML, per tal
* d'emmagatzemar-lo en la taula XMLprev_tab.
***/
private void crg_Previ() throws SQLException
   {
      NodeList nl = doc.getElementsByTagName("*");
      Element e;
      NamedNodeMap nnm;
      String emissor=""; // Contindrà l'identificador de l'emissor
      String num_enviament=""; // Contindrà l'identificador del núm d'enviament
      String operacio=""; // Contindrà l'operació a realitzar
      String nom_taula=""; // Contindrà el nom de la taula contra la que s'ha de
                           // portar a terme l'operació
      String condicio=""; // Contindrà les condicions, en cas que aquestes
                          // existeixin.
      String cos_xml="<?xml version=\"1.0\"?><ROWSET><ROW num=\"1\">";
      Node n;
      int i, len;
      String taula="XMLprev_tab"; // Nom de la taula que conté els enviaments
                                  // pendents del processament de Qualitat
      int ncondicio=0;

      len = nl.getLength();
      // Bucle general de lectura dels elements i atributs del DOM Document
      for (int j=0; j < len; j++)
         {
            e = (Element)nl.item(j);
            nnm=e.getAttributes();
            // Si es localitza l'element enviament
            if (e.getTagName().equals("enviament"))
               {
                 n=nnm.item(0);
                 // Llegeix el valor de l'identificador de l'emissor
                 emissor=n.getNodeValue();
                 n=nnm.item(1);
                 // Llegeix el valor de l'identificador del núm. d'enviament
                 num_enviament=n.getNodeValue();
               }
            // Si es localitza l'element registre
            if (e.getTagName().equals("registre"))
               {
                  n=nnm.item(0);
                  // Llegeix el valor que indica l'operació a realitzar
                  operacio=n.getNodeValue();
                  n=nnm.item(1);
                  // Llegeix el valor que indica la taula contra la que s'ha de
                  // realitzar l'operació.
                  nom_taula=n.getNodeValue();
               }
            if (e.getTagName().equals("camp")) // Si es localitza l'element camp
               { //es construeix el cos d'un document xml fàcil de tractar
                n=nnm.item(0);
                cos_xml=cos_xml.concat("<").concat(n.getNodeValue()).concat(">");
```

```
                if (nnm.getLength()>1)
                   cos_xml=cos_xml.concat(nnm.item(1).getNodeValue());
                cos_xml=cos_xml.concat("</").concat(n.getNodeValue()).concat(">");
                if (nnm.getLength()==3)
                   {
                      // Avalua l'existència d'un atribut que denoti que el camp és
                      // una condició
                     if (nnm.item(2).getNodeValue().equals("SI"))
                      {
                        ncondicio++;
                        condicio=condicio.concat("|").concat(nnm.item(0).
                                                              getNodeValue());
                      }
                   }
                }
            }
         // tanca el cos del nou document XML
         cos_xml=cos_xml.concat("</ROW></ROWSET>");
         if (ncondicio>0) // Si s'han localitzat condicions
            condicio=ncondicio+condicio; // afegeix la quantitat
        // Llegeix el Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

        // Estableix una connexió
        Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");

        // Crea un Statement
        Statement stmt = conn.createStatement ();

        // Realitza la inserció a la taula XMLprev_tab;
        ResultSet rset = stmt.executeQuery ("insert into "+ taula+" (nom_emissor,
                                num_enviament, operacio,nom_taula, xml,condicio)
                                values('"+emissor+"','" + num_enviament + "','"+
                                operacio + "','"+ nom_taula +
                                "','"+cos_xml+"','"+condicio+"')");

        rset.close();
        stmt.close();
        conn.commit();
        conn.close();
      }


   /**
   * env_inErr_tab
   * Paràmetres: emissor - Identificació de l'emissor de l'enviament
   *             n_enviament - Número d'enviament d'aquest emissor
   * Retorna: Cert/Fals
   *  Determina si un enviament ja ha estat processat previament i es
   *  troba dins de la taula XMLerr_tab, cosa que significaria que hauria
   *  estat tractat anteriorment, i que s'haurien detectat errors que no haurien
   *  permés la seva incorporació a la BBDD.
   ***/
   private boolean env_inErr_tab(String emissor,String n_enviament) throws
SQLException
      {
        int n_reg=0;
        // Llegeix el Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        // estableix una connexió
        Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
        // Crea un Statement
        Statement stmt = conn.createStatement ();
        // Executa una consulta sobre la taula XMLerr_tab, que determina l'existència
        // de l'enviament en aquesta taula
        ResultSet qrset=
           stmt.executeQuery ("select count(*) as n_registres from xmlerr_tab where
                                nom_emissor='"+emissor+"' "+
                                "and num_enviament='"+n_enviament+"'");
        // Llegeix el resultat de la consulta
        while (qrset.next())
           {
              n_reg=qrset.getInt("n_registres");
           }
```

```
        qrset.close();
        stmt.close();
        conn.commit();
        conn.close();

        if (n_reg==0) // Si no hi ha registres ...
           return(false); //... retorna FALS
        else
           return(true); // ... altrament, CERT
     }

   /**
   * env_inFatal_err_tab
   * Paràmetres: emissor - Identificació de l'emissor de l'enviament
   *             n_enviament - Número d'enviament d'aquest emissor
   * Retorna: Cert/Fals
   *  Determina si un enviament ja ha estat processat previament i es
   *  troba dins de la taula XMLFatal_err_tab, cosa que significaria que hauria
   *  estat tractat anteriorment, i que s'haurien detectat errors crítics que no
   *  haurien permés la seva incorporació a la BBDD.
   ***/
   public boolean env_inFatal_err_tab(String emissor,String n_enviament) throws
SQLException
      {
        int n_reg=0;
        // Llegeix el Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        // estableix una connexió
        Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
        // Crea un Statement
        Statement stmt = conn.createStatement ();
        // Executa una consulta sobre la taula XMLFatal_err_tab, que determina
        // l'existència de l'enviament en aquesta taula
        ResultSet qrset=
           stmt.executeQuery ("select count(*) as n_registres from xmlfatal_err_tab
                               where nom_emissor='"+emissor+"' "+
                               "and num_enviament='"+n_enviament+"'");
        // Llegeix el resultat de la consulta
        while (qrset.next())
           {
              n_reg=qrset.getInt("n_registres");
           }

        qrset.close();
        stmt.close();
        conn.commit();
        conn.close();

        if (n_reg==0) // Si no hi ha registres ...
           return(false); //... retorna FALS
        else
           return(true); // ... altrament, CERT
      }

   /**
   * err_tab2prev_tab
   * Paràmetres: emissor - Identificació de l'emissor de l'enviament
   *             n_enviament - Número d'enviament d'aquest emissor
   * Retorna: res
   *  Utilitzada per executar l'acció de reenviament. Còpia el contingut
   *  de la taula XMLerr_tab, a la taula XMLprev_tab, de manera que l'enviament passa
   *  d'estar considerat com un enviament fallit, a tornar a ser avaluat.
   ***/
   private void err_tab2prev_tab(String emissor,String n_enviament) throws
SQLException
      {
        String operacio="";
        String nom_taula="";
        String condicio="";
        String cos_xml="";

        // Llegeix el Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        // estableix una connexió
        Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
```

```
        // Crea un Statement
        Statement stmt = conn.createStatement ();
        //Executa la consulta sobre la taula XMLerr_tab, per obtenir-ne les dades de
        // l'enviament fallit
        ResultSet qrset=
           stmt.executeQuery ("select operacio,nom_taula,condicio,xml from xmlerr_tab
                               where nom_emissor='"+emissor+"' "+
                               "and num_enviament='"+n_enviament+"'");
        //Recull el resultat de la consulta
        while (qrset.next())
           {
              operacio=qrset.getString("operacio");
              nom_taula=qrset.getString("nom_taula");
              condicio=qrset.getString("condicio");
              cos_xml=qrset.getString("xml");
           }

        // realitza la inserció sobre la taula XMLPrev_tab
        ResultSet rset =
           stmt.executeQuery ("insert into xmlprev_tab "+
                               "(nom_emissor, num_enviament, operacio, nom_taula,
                                                                  xml, condicio) "+
                               "values('"+emissor+
                                      "','" + n_enviament +
                                      "','" + operacio +
                                      "','" + nom_taula +
                                      "','" + cos_xml +
                                      "','" +condicio+"')");
        rset.close();
        qrset.close();
        stmt.close();
        conn.commit();
        conn.close();
      }

   /**
   * esborrarErr_tab
   * Paràmetres: emissor - Identificació de l'emissor de l'enviament
   *             n_enviament - Número d'enviament d'aquest emissor
   * Retorna: res
   *  Elimina el contingut d'un enviament fallit de la taula XMLErr_tab
   ***/
   private void esborrarErr_tab(String emissor, String n_enviament) throws
SQLException
   {
        // Llegeix el Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        // estableix una connexió
        Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
        // Crea un Statement
        Statement stmt = conn.createStatement ();

        // Realitza l'esborrament de l'enviament fallit en la taula XMLErr_tab
        ResultSet rset =
           stmt.executeQuery ("delete from xmlerr_tab "+
                               "where nom_emissor='"+emissor+"'"+
                                " and num_enviament='"+n_enviament+"'");
        rset.close();
        stmt.close();
        conn.commit();
        conn.close();
   }

   /**
   * esborrarFatal_err_tab
   * Paràmetres: emissor - Identificació de l'emissor de l'enviament
   *             n_enviament - Número d'enviament d'aquest emissor
   * Retorna: res
   *  Elimina el contingut d'un enviament amb errors crítics de la taula
   *  XMLFatal_err_tab
   ***/
   private void esborrarFatal_err_tab(String emissor, String n_enviament) throws
SQLException
   {
        // Llegeix el Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

```
        // estableix una connexió
        Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
        // Crea un Statement
        Statement stmt = conn.createStatement ();

        // Realitza l'esborrament de l'enviament en la taula XMLfatal_err_tab
        ResultSet rset =
           stmt.executeQuery ("delete from xmlfatal_err_tab "+
                              "where nom_emissor='"+emissor+"'"+
                               " and num_enviament='"+n_enviament+"'");
        rset.close();
        stmt.close();
        conn.commit();
        conn.close();
   }

   /**
   * Fatal_err_tab2prev_tab
   * Paràmetres: emissor - Identificació de l'emissor de l'enviament
   *             n_enviament - Número d'enviament d'aquest emissor
   * Retorna: res
   *  Utilitzada per executar una acció de reenviament. Consulta i fa el tractament
   *  del contingut de la taula XMLFatal_err_tab, valida el nom del fitxer amb les
   *  dades contingudes en l'enviament tracta la informació continguda i transforma el
   *  document XML, i finalment còpia les dades ja tractades a la taula XMLprev_tab,
   *  de manera que l'enviament passa d'estar considerat com un enviament amb errors
   *  crítics a tornar a ser avaluat.
   ***/
   private int Fatal_err_tab2prev_tab(String emissor,String n_enviament) throws
SQLException
      {
        int error=NO_ERROR;
        String cos_xml="";
        String nomf=emissor+n_enviament+".xml"; // composa el nom del fitxer que
                                                //contenia l'enviament original
        // Llegeix el Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        // estableix una connexió
        Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
        // Crea un Statement
        Statement stmt = conn.createStatement ();
        //Executa la consulta sobre la taula XMLFatal_err_tab, per obtenir-ne les
        // dades de l'enviament fallit
        ResultSet qrset=
           stmt.executeQuery ("select xml from xmlfatal_err_tab where
                              nom_emissor='"+emissor+"' "+
                              "and num_enviament='"+n_enviament+"'");
        //Recull el que originalment era el contingut del document enviat
        while (qrset.next())
           {
              cos_xml=qrset.getString("xml");
           }
        // valida i parseja el contingut del document XML per verificar que aquest
        // sigui vàlid
        // i el carrega com un DOM document pel seu posterior tractament.
        try
        {

         DOMParser parser = new DOMParser();
         parser.setValidationMode(true);
         parser.showWarnings(true);

         StringReader reader = new StringReader(cos_xml);
         parser.parse(reader);

         doc = parser.getDocument();

        }
        catch (Exception e)
        {
         error=FATAL_ERROR;
        }

       if (error==NO_ERROR)
        {
```

```
         if (validar_nomenviament(nomf)) // Valida el nom de l'enviament
            {
               try{
                  crg_Previ(); // Realitza la carrega a la taula XMLprev_tab del
                               //document XML original
               }
               catch(SQLException e)
               {
                 System.out.println(e);
                 error=FATAL_ERROR;
               }
            }
         else
            {
               System.out.println("Enviament NO correcte "+nomf);
               error=FATAL_ERROR;
            }
       }
      qrset.close();
      stmt.close();
      conn.commit();
      conn.close();

      return(error);
     }

   /**
   * preCarregarFitxer
   * Paràmetres: nomf - nom del fitxer, del qual podrem extreure el identificador de
l'enviament
   * Retorna: int - Errors obtinguts en el processament
   *  Es l'encarregat de coordinar tots els processos de precarrega del fitxer
   *  que conté l'enviament, o de les taules d'error, en cas de tractar-se d'un
   *  enviament anterior fallit.
   *   - Comprova si es tracta d'un enviament anterior fallit, o és un nou enviament
   *   - En funció d'això:
   *      - O el carrega del fitxer al sistema com a DOM Document (invoca la funció
   *         DOMcarregar)
   *      - O el carrega de les taules que contenen els enviaments fallits
   *   - valida el nom de l'enviament (invoca la funció validar_nomenviament)
   *   - i si tot és correcte, realitza la carrega a la taula XMLprev_tab (invoca la
   *     funció crg_Previ)
   ***/
   public int preCarregarFitxer(String nomf)
      {
      int error=NO_ERROR;

      try{
      //primer avalua si es tracta d'un enviament anterior fallit (amb errors lleus)
      // per tant, s'avalua l'existència d'aquest enviament en la taula XMLErr_tab
      if (env_inErr_tab(nomf.substring(0,5),nomf.substring(5,8)))
         {
            // si és així, es còpia el seu contingut a la taula XMLPrev_tab, per
            // tornar a ser processat
            err_tab2prev_tab(nomf.substring(0,5),nomf.substring(5,8));
            // finalment es treu de la taula XMLErr_tab
            esborrarErr_tab(nomf.substring(0,5),nomf.substring(5,8));
         }
      //Si no es compleix el primer cas, s'avalua si es tracta d'un enviament anterior
      // fallit amb errors crítics.
      // per tant, s'avalua l'existència d'aquest enviament en la taula
      // XMLFatal_err_tab
      else if (env_inFatal_err_tab(nomf.substring(0,5),nomf.substring(5,8)))
         {
            //si és així, es tracta el contingut, i es copia a la taula XMLPrev_tab,
            // per tornar a ser processat
            error=Fatal_err_tab2prev_tab(nomf.substring(0,5),nomf.substring(5,8));
            // Si en el tractament del document no es produeixen errors ...
            if (error==NO_ERROR)
               // s'esborra de la taula XMLFatal_err_tab
               esborrarFatal_err_tab(nomf.substring(0,5),nomf.substring(5,8));
         }
      else //Sinó, el document xml no ha estat enviat anteriorment. S'ha de carregar
           // de fitxer.
         {
            if (DOMcarregar(nomf)==NO_ERROR) //Carrega el fitxer al sistema com a DOM
               {
```

```
               if (validar_nomenviament(nomf)) // Valida el nom de l'enviament
                  {
                     crg_Previ(); // Realitza la carrega a la taula XMLprev_tab del
                                  // document XML original
                  }
               else
                  {
                     System.out.println("Enviament NO correcte");
                     error=FATAL_ERROR;
                  }
            }
         else
            {
               System.out.println("Error al validar DTD");
               error=FATAL_ERROR;
            }
      }
   }
   catch(SQLException e)
   {
     System.out.println(e);
     error=FATAL_ERROR;
   }
   return(error);
   }

/**
* Pqualitat
* Paràmetres:
*            nome - identificador de l'emissor de l'enviament
*            n_enviament - identificador del núm. d'enviament
* Retorna: int - nivell d'error
*  Llegeix el registre en la taula XMLprev_tab, corresponent a l'enviament a
*  avaluar.
*  Seguidament n'avalua el nivell d'error, obtinguts en el processament de les
*  funcions de qualitat analitzen a nivell de registre.
*  Tot seguit, avalua el nivell d'error obtinguts en el processament de les
*  funcions de qualitat que refereixen a camps concrets de la taula.
*  Finalment, retorna el nivell d'errors total (suma dels dos anteriors)
***/
public int Pqualitat(String nome, String n_enviament) throws SQLException
   {
      int error=0;
      String emissor=nome; // conté l'identificador de l'emissor
      String num_enviament=n_enviament; // conté l'identificador del núm.
                                        // d'enviament
      String operacio=""; // contindrà la operació a realitzar
      String nom_taula=""; // contindrà la taula contra la que es desitja realitzar
                           // la operació
      String cos_xml=""; // contindrà el document XML transformat.
      DOMParser parser;
      XMLDocument localdoc;
      Element e;
      NamedNodeMap nnm;
      Node n;
      Node childNode;
      String tagName="";
      String tagValue="";
      int i, len;

      // Llegeix el Oracle JDBC driver
      DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      // estableix una connexió
      Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
      // crea un Statement
      Statement stmt = conn.createStatement ();

      // Realitza la consulta en la taula XMLprev_tab
      ResultSet rset =
           stmt.executeQuery ("select operacio,nom_taula,xml from xmlprev_tab where
                               nom_emissor='"+emissor+"' "+
                               "and num_enviament='"+num_enviament+"'");
      // Recull els valors resultants de la consulta
      while (rset.next())
         {
            operacio=rset.getString("operacio");
```

```
            nom_taula=rset.getString("nom_taula");
            cos_xml=rset.getString("xml");
         }
      // Validar a nivell de REGISTRE
      if (operacio.equals("INSERTAR") || operacio.equals("REENVIAR"))
         {
              // invoca la funció que realitzarà les operacions de la crida la
              // funcio PLSQL pertinent
             error=error+analitzarQ_taula(nom_taula,cos_xml);
         }
      // Validar a nivell d'ELEMENT (camp)
      // Parseja el document XML i n'extreu cadascun dels elements,
      // juntament amb el seu valor. Per cada un crida la funció per
      // avaluar si han de passar un control de qualitat
      try
      {
         parser = new DOMParser();
         parser.setValidationMode(false);

         StringReader reader = new StringReader(cos_xml);
         parser.parse(reader);
         localdoc = parser.getDocument();
         NodeList nl = localdoc.getElementsByTagName("*");

         len = nl.getLength();

         for (int j=0; j < len; j++)
            {
               n = nl.item(j);
               e = (Element )nl.item(j);
               tagName=e.getTagName();
               if (!tagName.equals("ROW") && !tagName.equals("ROWSET"))
                  {
                     if (n.getFirstChild()!=null)
                        {
                           childNode=n.getFirstChild();
                           tagValue=childNode.getNodeValue();
                           // invoca la funció que realitzarà les operacions de la
                           // crida la funcio PLSQL pertinent
                           error=error+
                                    analitzarQ_Element(nom_taula,tagName,tagValue);
                        }
                  }
             }

         }
         catch (Exception exp)
         {
            error=FATAL_ERROR;
         }
      //Si hi ha error, en cas que es tracti d'un reenviament o una operació
      // d'esborrar ...
      if (error>NO_ERROR && (operacio.equals("REENVIAR") ||
                             operacio.equals("ESBORRAR") ||
                             operacio.equals("DESCARREGAR")))
         // no es permet que hi hagi possibilitat de precarregar l'enviament a la
         // BBDD
         error=error+MAX_ERROR;

      rset.close();
      stmt.close();
      conn.commit();
      conn.close();
      return(error);
   }


/**
* analitzarQ_Element
* Paràmetres:
*            nom_taula - nom de la taula que conté el camp a analitzar
*            element - nom del camp a analitzar
*            valor - valor del camp a analitzar
* Retorna: int - nivell d'errors
*   Consulta sobre la taula de configuració, totes les funcions que que
*   un camp pugui tenir (cap, una o més d'una).
*   En cas de trobar-ne alguna, crida el procediment emmagatzemat PLSQL, i
```

```
   *   recull si s'ha acomplert correctament o no.
   *   En cas d'haver detectat algun error, es recull el nivell d'error
   *   establert en la taula de configuració.
   ***/
   private int analitzarQ_Element(String nom_taula, String element,String valor)
throws SQLException
     {
         String funcio="";
         int nerror=0;
         int error=0;
         CallableStatement Func_stmt=null;
         // Llegeix el Oracle JDBC driver
         DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
         //estableix la connexió
         Connection conn =
             DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
         //crea l'Statement
         Statement stmt = conn.createStatement ();

         // Realitza la consulta a la taula de configuració cfg_db
         ResultSet rset =
              stmt.executeQuery ("select funcio,nerror from cfg_db where
                                          taula='"+nom_taula+"' "+
                                          "and camp='"+element+"'");
         // Recull el resultat de la consulta
         while (rset.next())
            {
               funcio=rset.getString("funcio");
               nerror=Integer.valueOf(rset.getString("nerror")).intValue();
               if (!funcio.equals("")) // si s'ha trobat alguna funció
                  {
                     //realitza la crida al procediment emmagatzemat que ha d'avaluar
                     // si és correcte o no
                     Func_stmt = conn.prepareCall("begin ? := "+funcio+"(?); end;");
                     Func_stmt.registerOutParameter(1,OracleTypes.INTEGER);
                     Func_stmt.setString(2,valor);
                     Func_stmt.execute();
                     // si el resultat de l'execució del procediment és 0, es que hi
                     // ha errors
                     if (Func_stmt.getInt(1)==0)
                        {
                            error=error+nerror;
                        }
                  }
            }

         rset.close();
         stmt.close();
         conn.commit();
         conn.close();

         return(error);

     }

   /**
   * analitzarQ_taula
   * Paràmetres:
   *            nom_taula - nom de la taula que conté el registre a analitzar
   *            xml - cos del document XML transformat
   * Retorna: int - nivell d'errors
   *   Consulta sobre la taula de configuració, si aquesta taula té alguna funcio
   *   d'avaluació de la seva qualitat
   *   En cas de trobar-ne alguna, crida el procediment emmagatzemat PLSQL, i
   *   recull si s'ha acomplert correctament o no.
   *   En cas d'haver-se produït algun error, s'estableix valor del nivell d'error
   *   indicat en la taula de configuració.
   ***/
   private int analitzarQ_taula(String nom_taula, String xml) throws SQLException
     {
         String funcio="";
         int nerror=0;
         int error=0;
         CallableStatement Func_stmt=null;
         // Llegeix el Oracle JDBC driver
         DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
         //estableix la connexió
```

```
        Connection conn =
            DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
        //crea l'Statement
        Statement stmt = conn.createStatement ();

        // Realitza la consulta sobre la taula de configuració cfg_db
        ResultSet rset =
             stmt.executeQuery ("select funcio,nerror from cfg_db where
                                 taula='"+nom_taula+"' "+
                                 "and camp='*'");
        // Recull es resultats de la consulta
        while (rset.next())
           {
              funcio=rset.getString("funcio");
              nerror=Integer.valueOf(rset.getString("nerror")).intValue();
           }
        if (!funcio.equals("")) // si s'ha trobat alguna funció
           {
              //realitza la crida al procediment emmagatzemat que ha d'avaluar si
              // l'enviament és correcte o no
              Func_stmt = conn.prepareCall("begin ? := "+funcio+"(?); end;");
              Func_stmt.registerOutParameter(1,OracleTypes.INTEGER);
              Func_stmt.setString(2,xml);
              Func_stmt.execute();
              // si el resultat de l'execució del procediment és 0, es que hi ha
              // errors
              if (Func_stmt.getInt(1)==0)
                 {
                     error=nerror;
                 }
           }

        rset.close();
        stmt.close();
        conn.commit();
        conn.close();

        return(error);

    }

/**
* valorCamp
* Paràmetres:
*            camp - nom de l'element del qual es desitja obtenir el valor
*            cos_xml - cos del document XML
* Retorna: String - valor o contingut de l'element
*   Donat un document XML emmagatzemat en un String i donat un element,
*   contingut en aquest document(camp), retorna el seu valor o contingut
***/
private String valorCamp(String camp, String cos_xml)
   {
        XMLDocument localdoc;
        Node n;
        Node childNode;

        String tagValue="";
        try
        {
           //Parseja el document XML contingut en l'String cos_xml
           DOMParser parser = new DOMParser();
           parser.setValidationMode(false);

           StringReader reader = new StringReader(cos_xml);
           parser.parse(reader);
           localdoc = parser.getDocument();
           //Localitza l'element
           NodeList nl = localdoc.getElementsByTagName(camp);
           n = nl.item(0);
           //N'extreu el contingut
           if (n.getFirstChild()!=null)
              {
                 childNode=n.getFirstChild();
                 tagValue=childNode.getNodeValue();
              }

        }
```

```
      catch (Exception exp)
      {
         tagValue="";
      }

      return(tagValue);
   }

/**
* seguentNumDescarrega
* Paràmetres: camp
* Retorna: String - Següent identificador d'enviament (DESCARREGAR)
*  Consulta sobre la taula XMLDesc_log, per saber quin ha estat l'últim
*  enviament (ID) realitzat, incrementa el valor en una unitat (fent les
*  respectives conversions d'enter a string, i inversa), i retorna el
*  nou valor
***/
private String seguentNumDescarrega() throws SQLException
   {
     String desc_actual="";
     String desc_nova="";
     int x_nova=0;
     int desc_long=0;

     // Llegeix el Oracle JDBC driver
     DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
     // Estableix la connexió
     Connection conn =
       DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
     // crea l'Statement
     Statement stmt = conn.createStatement ();
     //Consulta l'últim valor introduït (el més gran) en l'identificador de
     // l'enviament de la taula "log" de descarregues
     ResultSet qrset=
        stmt.executeQuery ("select max(descarrega) as actual from xmldesc_log");

     //Recull el resultat de la consulta
     while (qrset.next())
        {
           desc_actual=qrset.getString("actual");
        }

     if (desc_actual!=null) //si hi ha enviaments anteriors ...
        {
         // ... realitza per obtenir el seu valor + 1
         x_nova=Integer.valueOf(desc_actual).intValue()+1;
         desc_nova="00"+x_nova;
         desc_long=desc_nova.length();
         if (desc_long>3)
            desc_nova=desc_nova.substring(desc_long-3);
        }
     else // Si es tracta del primer enviament ...
        desc_nova="001"; // directament s'assigna el valor 1

     qrset.close();
     stmt.close();
     conn.commit();
     conn.close();

     return(desc_nova);

   }

/**
* carregarFitxer
* Paràmetres: nomf - nom del fitxer. Utilitzat per obtenir l'identificador de
*               l'emissor i el núm. d'enviament
* Retorna: int - nivell d'errors en cas de produir-se
*  Llegeix el registre contingut en la taula XMLprev_tab, n'obté els valors
*  i executa l'operació indicada contra la base de dades.
*  És per tant, la funció encarregada d'executar definitivament l'operació definida
*  en l'enviament sobre la base de dades.
***/
public int carregarFitxer(String nomf) throws SQLException
{
   int error=0;
   String emissor=nomf.substring(0,5); // Conté l'identificador de l'emissor
```

```
String num_enviament=nomf.substring(5,8); // Conté l'id. de núm. d'enviament
String operacio=""; // Contindrà l'operació a realitzar
String nom_taula=""; // Contindrà el nom de la taula contra la que s'ha de
                     // realitzar l'operació
String condicio=""; // Contindrà les condicions en cas d'existir (per a
                    // KeyColumnList)
String condiciosql=""; // La mateixa condició, en cas d'existir, també es
                       // composa en SQL (per a Select)
String cos_xml=""; // Contindrà el cos del document XML
String condiciotmp="";
int i=0;
String [] keyColNames;

// Llegeix el Oracle JDBC driver
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
// estableix la connexió
Connection conn =
    DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");
//crea l'Statement
Statement stmt = conn.createStatement ();

// Realitza la consulta sobre la taula XMLprev_tab per tal d'obtenir la
// informació sobre l'enviament
ResultSet rset =
     stmt.executeQuery ("select operacio,nom_taula,condicio,xml from xmlprev_tab
                         where nom_emissor='"+emissor+"' "+
                         "and num_enviament='"+num_enviament+"'");
// Recull el resultat de la consulta
while (rset.next())
   {
      operacio=rset.getString("operacio");
      nom_taula=rset.getString("nom_taula");
      condicio=rset.getString("condicio");
      cos_xml=rset.getString("xml");
   }

OracleXMLSave sav = new OracleXMLSave(conn, nom_taula);

// Si la s'han especificat camps condicionals per l'operació, es preparen
// per poder ser utilitzats.(ja sigui per KeyColumnList com per sentències
// Select)
if (condicio!=null && !condicio.equals("null"))
   {
      String camp="";
      String valor="";
      condiciotmp=condicio;
      keyColNames = new String [(int)
                  Integer.valueOf(condicio.substring(0,condicio.indexOf('|'))).
                                                                  intValue()];
      condiciotmp=condicio.substring(condicio.indexOf('|')+1);

      while(condiciotmp.indexOf('|')>=0)
      {
         camp=condiciotmp.substring(0,condiciotmp.indexOf('|'));
         keyColNames[i]=camp;
         if (condiciosql.length()>0)
            condiciosql=condiciosql+" and ";
         condiciosql=condiciosql+camp+"='"+valorCamp(camp,cos_xml)+"' ";
         condiciotmp=condiciotmp.substring(condiciotmp.indexOf('|')+1);
      }
      camp=condiciotmp.substring(0);
      keyColNames[i]=camp;
      if (condiciosql.length()>0)
         condiciosql=condiciosql+" and ";
      condiciosql=condiciosql+camp+"='"+valorCamp(camp,cos_xml)+"' ";
      sav.setKeyColumnList(keyColNames);
   }

// En funció de quina sigui l'operació a realitzar...
if (operacio.equals("INSERTAR")) //si es tracta d'una inserció...
   {
      try {
      sav.insertXML(cos_xml); // realitza la inserció.
      }
      catch (oracle.xml.sql.OracleXMLSQLException e) {
         error=SAV_ERROR;
      }
```

```
   }
else if (operacio.equals("REENVIAR")) // si es tracta d'un reenviament ...
   {

       try {
         sav.updateXML(cos_xml); // realitza l'update
       }
       catch (oracle.xml.sql.OracleXMLSQLException e) {
         error=SAV_ERROR;
       }
   }
else if (operacio.equals("ESBORRAR")) // si es tracta d'un esborrament...
   {

       try {
         sav.deleteXML(cos_xml); // esborra el/s registres
       }
       catch (oracle.xml.sql.OracleXMLSQLException e) {
         error=SAV_ERROR;
       }
   }
else if (operacio.equals("DESCARREGAR")) // si es tracta d'una descarrega...
   {
      String next_nlog="";
      FileOutputStream fo;
      PrintStream sortida;
      // realitza la consulta sobre la taula indicada, i amb les condicions
      // especificades
      ResultSet qset = stmt.executeQuery("select * from "+nom_taula+" where
                                          "+condiciosql);

      // inicia el OracleXMLQuery per la consulta especificada
      OracleXMLQuery qry =  new OracleXMLQuery(conn,qset);

      // obté el document XML en forma d'String
      String xmlString = qry.getXMLString();

      // obté el valor del identificador, per composar i identificar
      // la descarrega
      try
      {
      next_nlog=seguentNumDescarrega();
      }
      catch (SQLException e)
      {
          next_nlog="";
          error=FATAL_ERROR;
      }
      if (next_nlog.length()>0)
       {
           xmlString=xmlString.replace('\'','"');
           // inserta el contingut de la descàrrega en la taula XMLDesc_log,
           // per guardar un registre/control de la mateixa
           ResultSet iset = stmt.executeQuery("insert into xmldesc_log
                              (descarrega,sol_licitant,nsol_licitud,xml)"+
                              " values('"+next_nlog +"','"+emissor+"','"
                              +num_enviament+"','"+xmlString+"')");
           // composa el nom del fitxer XML resultant, i guarda
           // la informació en el fitxer
           try{
           fo=new FileOutputStream(TENDA+next_nlog+".xml");
           sortida=new PrintStream(fo);
           sortida.println(xmlString);
           sortida.flush();
           sortida.close();
           }
           catch(Exception e)
           {
              System.out.println("Error guardant el fitxer: "+e);
              error=FATAL_ERROR;
           }
       }
   }

sav.close();
rset.close();
```

```
      stmt.close();
      conn.commit();
      conn.close();

      return(error);
   }

   /**
   * inserta_en_Error_tab
   * Paràmetres: nomf - conté informació sobre l'identificador de l'emissor i del núm.
   *                    d'enviament
   *             error - Conté el nivell d'error obtingut del sistema de control de
   *                     qualitat.
   * Retorna: res
   *  Llegeix la taula XMLprev_tab, i la replica a la taula XMLerr_tab, afegint-hi el
   *  nivell d'error obtingut del sistema de control de qualitat.
   ***/
   public void inserta_en_Error_tab(String nomf, int error) throws SQLException
   {
        String emissor=nomf.substring(0,5);
        String num_enviament=nomf.substring(5,8);
        String operacio="";
        String nom_taula="";
        String condicio="";
        String cos_xml="";

        // Llegeix el Oracle JDBC driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        // estableix la connexió
        Connection conn =
          DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");

        Statement stmt = conn.createStatement ();
        //Executa la consulta sobre la taula XMLprev_tab, per obtenir-ne les dades de
        //l'enviament
        ResultSet qrset=
           stmt.executeQuery ("select operacio,nom_taula,condicio,xml from xmlprev_tab
                                       where nom_emissor='"+emissor+"' "+
                                       "and num_enviament='"+num_enviament+"'");
        //Recull el resultat de la consulta
        while (qrset.next())
           {
              operacio=qrset.getString("operacio");
              nom_taula=qrset.getString("nom_taula");
              condicio=qrset.getString("condicio");
              cos_xml=qrset.getString("xml");
           }

        // realitza la inserció sobre la taula XMLerr_tab
        ResultSet rset =
           stmt.executeQuery ("insert into xmlerr_tab "+
                              "(nom_emissor, num_enviament, operacio, nom_taula,
                                nivell_error, xml, condicio) "+
                              "values('"+emissor+
                                     "','" + num_enviament +
                                     "','" + operacio +
                                     "','" + nom_taula +
                                     "','" + error +
                                     "','" + cos_xml +
                                     "','" +condicio+"')");
        rset.close();
        qrset.close();
        stmt.close();
        conn.commit();
        conn.close();
   }

   /**
   * inserta_en_crgtError_tab
   * Paràmetres: nomf - conté informació sobre l'identificador de l'emissor i del núm.
   *                    d'enviament
   *             error - Conté el nivell d'error obtingut del sistema de control de
   *                     qualitat.
   * Retorna: res
   *  Llegeix la taula XMLprev_tab, i la replica a la taula XMLcrgerr_tab, afegint-hi
   *  el nivell d'error obtingut del sistema de control de qualitat.
   ***/
```

```
public void inserta_en_crgtError_tab(String nomf, int error) throws SQLException
{
     String emissor=nomf.substring(0,5);
     String num_enviament=nomf.substring(5,8);
     String operacio="";
     String nom_taula="";
     String condicio="";
     String cos_xml="";

     // Llegeix el Oracle JDBC driver
     DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
     // estableix la connexió
     Connection conn =
       DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");

     Statement stmt = conn.createStatement ();
     //Executa la consulta sobre la taula XMLprev_tab, per obtenir-ne les dades de
     //l'enviament
     ResultSet qrset=
        stmt.executeQuery ("select operacio,nom_taula,condicio,xml from xmlprev_tab
                              where nom_emissor='"+emissor+"' "+
                              "and num_enviament='"+num_enviament+"'");
     //Recull el resultat de la consulta
     while (qrset.next())
        {
           operacio=qrset.getString("operacio");
           nom_taula=qrset.getString("nom_taula");
           condicio=qrset.getString("condicio");
           cos_xml=qrset.getString("xml");
        }

     // realitza la inserció sobre la taula XMLcrgerr_tab
     ResultSet rset =
        stmt.executeQuery ("insert into xmlcrgerr_tab "+
                           "(nom_emissor, num_enviament, operacio, nom_taula,
                             nivell_error, xml, condicio) "+
                           "values('"+emissor+
                                  "','" + num_enviament +
                                  "','" + operacio +
                                  "','" + nom_taula +
                                  "','" + error +
                                  "','" + cos_xml +
                                  "','" +condicio+"')");
     rset.close();
     qrset.close();
     stmt.close();
     conn.commit();
     conn.close();
}

/**
* esborrarXMLPrev
* Paràmetres: nomf - conté informació sobre l'identificador de l'emissor i del núm.
*                    d'enviament
* Retorna: res
*   Esborra el contingut corresponent a l'enviament, de la taula
*   XMLprev_tab;
***/
public void esborrarXMLPrev(String nomf) throws SQLException
{
     String emissor=nomf.substring(0,5); //Conté l'identificador de l'emissor
     String num_enviament=nomf.substring(5,8); // Conté l'identificador del núm. de
                                               // l'enviament
     // Llegeix el Oracle JDBC driver
     DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
     // estableix la connexió
     Connection conn =
       DriverManager.getConnection ("jdbc:oracle:oci8:"+usr_pass+"@");

     Statement stmt = conn.createStatement ();

     // Realitza l'esborrament de l'enviament en la taula XMLprev_tab
     ResultSet rset =
        stmt.executeQuery ("delete from xmlprev_tab "+
                           "where nom_emissor='"+emissor+"'"+
                            " and num_enviament='"+num_enviament+"'");
```

```
      rset.close();
      stmt.close();
      conn.commit();
      conn.close();
   }


   /**
   * cfg_DB
   * Paràmetres: cap
   * Retorna: res
   *  Realitza la càrrega de la informació de configuració del sistema
   *  de qualitat sobre la taual cfg_tab, extraient la informació del
   *  document XML cfg_db.xml.
   ***/
   public void cfg_DB() throws SQLException
   {
    String tabName = "cfg_db"; // nom de la taula que conté la informació del sis. de
                               // C.Qualitat
    String fileName = "cfg_db.xml"; // nom del fitxer XML que conté la informació de
                                    // configuració
    // Llegeix el Oracle JDBC driver
    DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
    // estableix la connexió
    Connection conn =
      DriverManager.getConnection("jdbc:oracle:oci8:"+usr_pass+"@");

    OracleXMLSave sav = new OracleXMLSave(conn, tabName);
    // realitza la carrega del contingut del fitxer a la taula de configuració
    URL url = sav.getURL(fileName);
    int rowCount = sav.insertXML(url);

    System.out.println("Insertades "+rowCount+
               " regles, a la taula "+ tabName);

    sav.close();
    conn.close();
   }

   /**
   * usr_Pass
   * Paràmetres: cap
   * Retorna: res
   *  Demana la introducció de l'usuari i el password a la BBDD, en forma de
   *  usuari/password, i guarda el valor per utilitzar-lo en totes
   *  les operacions on sigui requerit
   ***/
   public void usr_Pass()
      {
         String up="";
         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

         // demana la introducció de l'usuari i password
         System.out.println("Indica Usuari/Password:");
         try {
         up=in.readLine();
         } catch(java.io.IOException ioex) {
         System.out.println("Error en introduir Usuari i/o Password");
         }
         // el guarda a la variable privada usr_pass, per utilitzar-lo
         // quan faci falta.
         if (up.length()>=3)
            {
               usr_pass=up;
            }
         else
            {
                System.out.println("Error en introduir Usuari i/o Password");
            }

      }

   private URL createURL(String fileName)
   {
      URL url = null;
      try
      {
```

```
            url = new URL(fileName);
         }
         catch (MalformedURLException ex)
         {
            File f = new File(fileName);
            try
            {
               String path = f.getAbsolutePath();
               String fs = System.getProperty("file.separator");
               if (fs.length() == 1)
               {
                  char sep = fs.charAt(0);
                  if (sep != '/')
                     path = path.replace(sep, '/');
                  if (path.charAt(0) != '/')
                     path = '/' + path;
               }
               path = "file://" + path;
               url = new URL(path);
            }
            catch (MalformedURLException e)
            {
               System.out.println("Cannot create url for: " + fileName);
               System.exit(0);
            }
         }
         return url;
      }
}
```