Títol del Projecte Fi de Carrera

Jose Martínez Fuentes
Enginyeria en Informàtica

Jordi Ferrer Duran

11/01/2008

2 Dedicatòria i agraïments

No voldria acabar la meva trajectòria acadèmica a la UOC sense valorar molt positivament tots aquests semestres què he estat estudiant la Enginyeria en Informàtica i, sobretot, posar èmfasi en la alta qualitat dels continguts de totes les assignatures que he cursat.

Pel què fa al projecte, vull agrair especialment al meu consultor, en Jordi Duran Ferrer, per donar-me la possibilitat de realitzar aquest projecte tant interessant i per donar-me suport en tot allò què m'ha fet falta per assolir amb èxit la consecució d'aquest PFC.

3 Resum del projecte

Aquest projecte de final de carrera pertany a l'àrea de Compiladors i la seva missió fonamental és aprendre els trets fonamentals de l'arquitectura del format ODF (OpenDocumentFormat) per tal de realitzar una aplicació que modifiqui documents d'aquest format.

De tots els tipus de documents que inclou l'estàndard ODF ens centrarem en els fitxers de tipus text (l'equivalent en la suite de Microsoft Office seria el Word).

Partirem d'un fitxer extern en format XML amb una llista de parelles "conjunt de termes" – referència", a més per validar el seu format s'utilitzarà una DTD. Aquest fitxer proporcionat per l'usuari ens servirà per introduir la bibliografia al document original a través del següent procés automatitzat:

- Afegir, el PRIMER COP que aparegui un terme de la llista en el document OpenOffice original, la referència que tingui associades al peu de la pàgina.
- Afegir al final del document una pàgina amb les referències que s'han utilitzat en el document.

Resumint, caldrà construir una eina informàtica que llegeixi els arxius XML que representen el document d'OpenOffice i la llista de termes i, com a sortida, generi el document original modificat segons les dues tasques indicades anteriorment. L'eina llegeix el fitxer XML (i el valida amb la seva DTD) amb la llista de termes i busca aquests termes en el document. Si hi apareixen, afegeix una referència bibliogràfica (e.g. [1]) en la primera aparició del terme i es detalla la referència en el peu de la pàgina corresponent.

Índex de continguts

CAPÍTOL 1: INTRODUCCIÓ

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

En l'actualitat, tothom està acostumat a realitzar documents de text, fulles de càlcul o presentacions tant en l'àmbit domèstic com en el professional. Però moltes vegades per realitzar aquests tipus de documents necessitem programes propietaris amb el cost econòmic què això comporta (com per exemple la suite de Microsoft, Office).

Aquí és on recau el principal avantatge del format obert OpenDocument ja què sense cap cost podrem realitzar tot tipus de documentació (podrem utilitzar programes de lliure distribució que suportin aquest format).

Si ens centrem en aspectes més tècnics, és molt interessant conèixer aquest format estàndard per tal de poder manipular aquests documents i realitzar processos automatitzats que ens facilitin un gran nombre de tasques.

I és gràcies a la realització d'aquest PFC que tindrem la possibilitat d'endinsar-nos en el món del XML i de veure les grans avantatges que té, fins i tot, per ser la base d'un format obert i estàndard com és l'ODF. Per tant, aquest PFC serà un mitjà per aprendre les diferents tecnologies relacionades amb aquest tipus de documents i que ens servirà com a base per realitzar una aplicació que automatitzi un procés de modificació d'un document ODF.

En definitiva, gràcies a aquest treball de recerca i desenvolupament tindré, per primera vegada, la possibilitat d'estudiar una tecnologia oberta i estàndard i de veure de les grans possibilitats que pot tenir alhora de desenvolupar aplicacions que interaccionin amb aquesta tecnologia. Per últim, però no menys important, tenir la possibilitat de poder veure en primera persona com un estàndard obert no té res d'envejar a un estàndard propietari.

1.2 Objectius del PFC

Molts de nosaltres segur que hem sentit parlar de la suite de lliure distribució OpenOffice per realitzar tot tipus de documentació, però el què igual no coneix tanta gent és l'estructura interna d'un d'aquests documents.

És aquí on recau el treball principal d'aquest PFC, en l'estudi de l'arquitectura interna dels documents OpenOffice i en veure el gran potencial que ens proporciona la utilització del format XML com a base d'aquests documents.

Per això s'ha de desenvolupar una aplicació que realitzi de forma automatitzada la inserció de referències bibliogràfiques en un document OpenOffice. Aquestes referències formaran part d'un document extern que serà validat per una DTD, per tant, de forma implícita s'estudiarà el format XML per l'elaboració estructurada d'informació i el format DTD per la seva validació.

Per tant, l'aplicació a desenvolupar no és un objectiu si no un mitjà per a estudiar la tecnologia XML, DTD i l'arquitectura interna dels documents OpenOffice.

Finalment, dir que un objectiu paral·lel al explicat anteriorment, però no menys important, és fer veure al lector les grans avantatges que ens proporciona el tipus d'arquitectura utilitzada per OpenOffice, gràcies a la utilització d'un format estàndard i conegut, com el XML. I, una vegada coneguda l'arquitectura interna, veure les grans possibilitats de manipulació d'aquests documents (com a exemple tenim l'aplicació desenvolupada en aquest projecte).

1.3 Enfocament i mètode seguit

Com s'ha explicat al capítol anterior, haurem de realitzar una aplicació per modificar documents de text d'OpenOffice de forma automatitzada. Per realitzar aquesta tasca s'ha de partir d'uns coneixements previs en les tecnologies utilitzades i, per aquest motiu, hi haurà una tasca important de cerca i estudi d'informació.

Com a primer pas, haurem de realitzar un estudi del format XML i DTD i, posteriorment, veure l'arquitectura interna que utilitza OpenDocument per generar els seus documents.

Una vegada tinguem aquests conceptes clars, realitzarem la DTD que validarà l'estructura d'entrada de dades, les referències bibliogràfiques i, després, crearem el fitxer XML que compleixi la nostra DTD. En aquest punt, ja tindrem l'estructura d'entrada de dades creada què ens servirà per realitzar les proves de l'aplicació que crearem.

El segon pas serà estudiar el format i l'arquitectura interna dels documents OpenOffice.

Serà en aquest punt, una vegada s'hagin assimilat els diferents conceptes explicats, quan es començarà a dissenyar l'aplicació, què, en definitiva, consistirà en plasmar programàticament, tot l'aprés anteriorment.

El cicle de vida del desenvolupament de l'aplicació serà el clàssic model en cascada, amb les fases més característiques com són presa de requeriments, anàlisis de requeriments, disseny, implementació i proves.

1.4 Planificació del projecte

Aquest projecte té dos grans blocs de treball ben diferenciat, per una banda, l'estudi de les tecnologies (XML, DTD i OpenOffice) i, per un altre, el desenvolupament d'un producte fruit del què s'ha aprés anteriorment.

Per justificar l'estimació temporal empleada en les diferents tasques, s'han establert els següents criteris.

- Bloc d'estudi de tecnologies.
  - He fet una estimació aproximada fruit de la meva experiència com a estudiant i com a investigador de noves tecnologies a la feina posant com a variables el tipus de tecnologia i la profunditat d'estudi de les mateixes.

- Bloc d'enginyeria del programari.
  - Per aquest bloc es podria haver optat per la utilització d'un model d'estimació de costos com el COCOMO II (estudiat a l'assignatura *Metodologia i Gestió de Projectes Informàtics*) que ens facilita l'estimació de les diferents parts del projecte a partir d'una sèrie de variables (com potser si el desenvolupament és orientat a objecte, l'experiència en el llenguatge de programació…). Però he cregut convenient fer una estimació basant-me en la meva experiència com a desenvolupador de C#, sobretot, per la grandària del projecte, que és suficientment petit com per ser precís en les estimacions realitzades.
    Això sí, encara què per aquest projecte, i pels motius esmentats, no he utilitzat una eina d'estimació de costos, en condicions normals, el COCOMO II hauria estat la emprada ja què tots sabem l'important i crític que és en un projecte les valoracions temporals.

A continuació es mostra la planificació realitzada amb Microsoft Project 2003:

Nombre de tarea	Duración	Comienzo	Fin
Elaboració del Pla de Treball	12 días?	lun 24/09/07	mar 09/10/07
Fita externa 1: Entrega del Pla de Treball	0 días	mar 09/10/07	mar 09/10/07
Redacció de la motivació i resum descriptiu del PFC	3 días?	mié 10/10/07	vie 12/10/07
Estudi capítol 2 - XML	7 días?	sáb 13/10/07	sáb 20/10/07
Redacció capítol 2 - XML	6 días?	lun 22/10/07	lun 29/10/07
Estudi capítol 3 - OpenDocument (Part1)	9 días?	mar 30/10/07	vie 09/11/07
Fita externa 2: Entrega PAC2	0 días	vie 09/11/07	vie 09/11/07
Estudi capítol 3 - OpenDocument (Part2)	3 días?	sáb 10/11/07	mar 13/11/07
Redacció capítol 3 - OpenDocument	8 días?	mié 14/11/07	vie 23/11/07
Enginyeria del Software	**16 días?**	**sáb 24/11/07**	**vie 14/12/07**
Anàlisi de requeriments	3 días?	sáb 24/11/07	mar 27/11/07
Disseny	5 días?	mié 28/11/07	mar 04/12/07
Implementació	8 días?	mié 05/12/07	vie 14/12/07
Fita externa 3: Entrega PAC 3	0 días	vie 14/12/07	vie 14/12/07
Redacció capítol 4 - Enginyeria del Software	7 días?	sáb 15/12/07	sáb 22/12/07
Elaboració exemples i jocs de proves	7 días?	dom 23/12/07	dom 30/12/07
Conclusions, glossari, bibliografia, linias futures	4 días?	lun 31/12/07	jue 03/01/08
Repàs final a la memòra	1 día?	vie 04/01/08	vie 04/01/08
Elaboració de la presentació	5 días?	sáb 05/01/08	jue 10/01/08
Fita externa 4: Entrega final PFC	0 días	vie 11/01/08	vie 11/01/08

Explicació

Després de fer un anàlisi a priori del projecte, he identificat 3 parts diferenciades de desenvolupament (corresponents a cadascuna de les entregues a realitzar) que em serveixen per a dividir el projecte i poder estimar de forma més acurada el cost en temps del mateix:

Pla de treball

Aquest ha estat el punt de partida per realitzar el PFC. En el pla de treball s'han identificat les tasques més importants que s'hauran de realitzar i, a partir d'aquestes, poder fer una estimació temporal més acurada de les diferents parts que consta aquest projecte. Per tant, la planificació s'ha basat en les tasques identificades en aquest pla de treball.

Estudi del llenguatge XML

Aquest és un dels dos punts importants a estudiar i investigar en aquest projecte. Primer de tot farem una breu descripció d'aquest llenguatge, exposant les característiques més importants i la sintaxi necessari per poder elaborar un document XML. Per altra banda, estudiarem quins són els diferents tipus de validacions que podem fer a un XML centrant-nos en les DTDs.

D'aquest tipus de validació s'explicarà també la seva sintaxi i de com valida un document XML a través de la seva descripció. Per últim, es farà un recull dels avantatges i inconvenients que té aquest sistema de validació respecte a d'altres.

Estudi del format OpenDocument

Aquest és el punt central d'estudi del projecte a on tindrem la missió d'aprendre tot el necessari d'aquest format per elaborar l'aplicació que modificarà documents ODF.

Primer de tot es farà una introducció explicant què és un format lliure de document. Després s'estudiaran les característiques més importants d'OpenDocument explicant les diferents fases per on ha passat aquest format fins convertir-se en un estàndard. Posteriorment, entrarem en temes més tècnics i arquitectònics d'aquest format explicant i detallant l'estructura interna d'un document així com cadascun dels documents xml que el composen.

Per una altra banda, explicarem quines són les eines o llibreries de suport al programador que ens proporciona la comunitat oberta de desenvolupament i veurem fins a quin punt ens poden servir d'ajuda per assolir la fita del nostre projecte.

I, per últim, farem un breu repàs als diferents formats de documents oberts que existeixen al mercat i compararem l'estàndard ODF amb el nou format de documents que Microsoft ha llançat al mercat, OpenXML (utilitzat en la nova suite ofimàtica Office 2007).

Enginyeria del software

Aquest és l'altre gran bloc de desenvolupament del projecte que consistirà a realitzar tots els passos necessaris per elaborar una aplicació que compleixi les especificacions detallades a la descripció del PFC.

Aquest apartat l'hem dividit en 3 blocs que són els següents:

- Anàlisi de requeriments.
    - En aquest punt realitzarem les tasques prèvies de recollida de requeriments i anàlisi dels mateixos per tal de tenir clars quins són els objectius que haurà d'assolir aquest projecte.

- Disseny.
    - En el disseny explicarem tant el disseny de l'aplicació (disseny de classes i d'interfícies d'usuari) com el disseny del fitxer XML d'entrada juntament amb la seva DTD de validació.

- Implementació
    - La implementació es centrarà en temes específics del llenguatge de programació utilitzat per elaborar l'aplicació, així com d'exposar diferents exemples de codi de les parts més significatives del projecte.

Elaboració d'exemples i jocs de proves

En aquest punt s'estudiaran les diferents casuístiques que ens podem trobar alhora d'utilitzar l'aplicació. A partir d'aquest anàlisi s'elaboraran uns exemples d'entrada de dades que ens serveixin per validar el funcionament de la nostra aplicació.

1.5 Productes obtinguts

Els productes obtinguts del desenvolupament d'aquest projecte són:

Anàlisi del XML

L'anàlisi del llenguatge XML consta d'una descripció del format, explicació de la seva sintaxi i dels seus avantatges i inconvenients. Per una altra banda, explicació de les DTDs com a mètodes per validar documents XML.

Anàlisi de l'OpenDocument

L'anàlisi del format ens donarà una visió global del format ODF juntament amb les nocions tècniques necessàries de la seva arquitectura per tal de poder abordar amb garanties l'elaboració de l'aplicació d'aquest projecte. Per altre banda, també obtindrem una comparació molt interessant entre aquest format i l'OpenXML el nou estàndard de documents de la nova versió d'Office.

L'aplicació en C#

Aquesta aplicació serà la part visible de tot l'estudi realitzat prèviament. El programari ens permetrà donat uns fitxers d'entrada (XML amb les referències bibliogràfiques i document ODF a tractar) afegir les referències bibliogràfiques al document i guardar les modificacions realitzades.

Jocs de proves

No només és important la realització de l'aplicació sinó també dels conjunt de proves que validen el seu perfecte funcionament. Per tant, a més de l'aplicació, s'entregarà un conjunt de fitxers d'entrada, indicant en cadascun d'ells quins aspectes volen validar i les sortides resultants d'executar l'aplicació amb aquestes dades d'entrada.

1.6 Breu descripció dels altres capítols de la memòria

Tal com s'ha explicat en punts anteriors, la realització d'aquest PFC està dividida en dos parts ben diferenciades. Per una banda, en l'estudi de les diferents tecnologies necessàries per elaborar aquest projecte i, per un altre, en la realització del programari que modificarà un document de text amb format OpenDocument.

Per tant, l'estructura de capítols que es mostraran a continuació seguiran aquesta mateixa estructura. Els dos capítols següents estan dedicats a l'estudi dels formats XML, DTD i OpenDocument, explicant les principals característiques de cadascun d'ells. Després tindrem un capítol dedicat al procés d'Enginyeria de Software seguit per elaborar el programari. Posteriorment, hi haurà un capítol dedicat a les línies futures de treball que podrà seguir aquest PFC. I, per últim, una breu esmenta dels recursos utilitzats per desenvolupar aquest projecte.

Com a finalització del PFC hi haurà un capítol amb les conclusions i valoracions personals, un glossari de termes i, per últim, la bibliografia utilitzada.

CAPÍTOL 2: ESTUDI DEL FORMAT XML I DTD

2.1 Característiques d'un document XML

A nivell introductori, es podria definir XML com a un conjunt de regles per definir etiquetes semàntiques que ens organitzen un document en diferents parts.

Les característiques més importants d'aquest format són:

XML és un mètode per introduir dades estructurades en un fitxer de text

Aquesta és una de les més importants característiques ja que ens permet dotar a un fitxer pla, d'estructura (a través d'etiquetes) que sigui fàcilment llegible i sense perdre el reduït tamany dels fitxers de text pla.

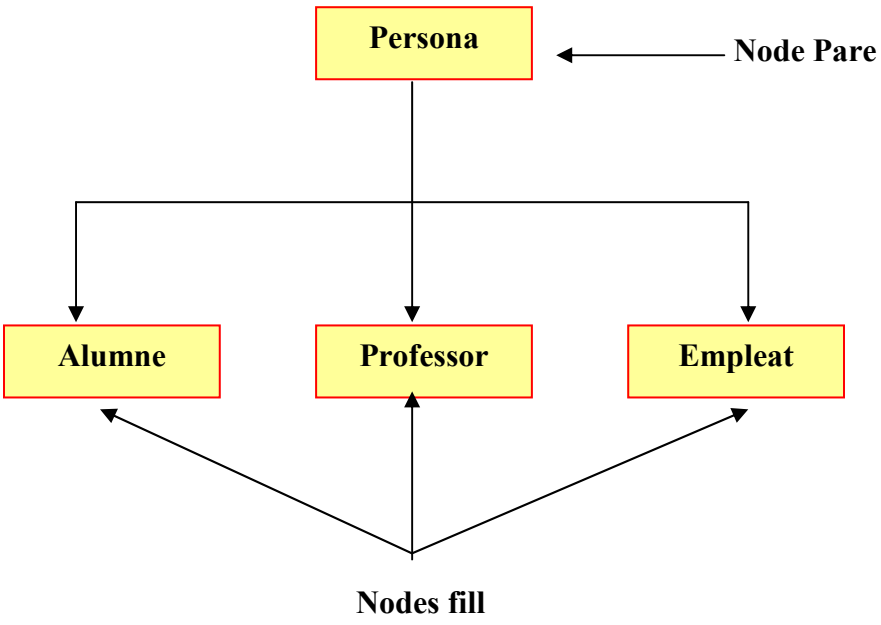
Per exemple:

<Nom> Juan </Nom>

En aquest cas estem creant una etiqueta personalitzada que serà del tipus *Nom*.

XML permet crear estructures jeràrquiques.

A més d'estructurar el text pla, ens permet crear una jerarquia per a totes les marques creades.



Com es pot observar, aquesta estructura jeràrquica és semblant a la què s'utilitza en una Base de Dades, ja què aquestes estan formades per taules, que contenen files i aquestes un conjunt de camps.

XML accepta diferents codificacions de caràcters.

Per ser un format estàndard i obert és necessari què accepti diferents codificacions ja què en funció de l'idioma, la procedència o la cultura, el document pot està representat amb una codificació o un altre.

El format XML soluciona aquest problema donant la possibilitat d'incloure en la seva capçalera la codificació amb la què ha estat escrit aquell document. D'aquesta manera quan l'analitzador que tracti el document intenti llegir-lo, detectarà el seu format , per tant, podrà interpretar-lo.

La codificació més usual què ens trobarem es la UTF-8 o UTF-16, per aquest motiu, només quan utilitzem una codificació diferent ho haurem d'indicar a la capçalera.

A continuació es mostra un exemple :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Aquesta capçalera ens està indicant que el document XML està escrit amb la codificació ISO-8859-1 (accepta caràcters amb accents).

2.1.1 Sintaxis d'un document XML

El llenguatge estàndard XML especifica unes normes bàsiques a nivell de sintaxis, a complir per tots aquells documents amb la intenció de denominar-se XML. Les característiques bàsiques exigides són:

Declaració XML com a etiqueta obligatòria

```
<?xml version="1.0" encoding="UTF-8" standardalone=”yes”?>
```

Aquesta etiqueta proporciona la següent informació:

- Informació de la versió XML (**obligatòria**). La versió més utilitzada és la 1.0.

- Codificació de caràcters utilitzada (**opcional**). Fa referència al mode en què es representen internament els caràcters, normalment UTF-8 o UTF-16.

- Declaració independent (**opcional**). Indica al processador XML si un document és independent (standardalone=”yes”) o es basa en informació de fonts externes, és a dir, si depèn de declaracions de marca externes com una DTD (standardalone = “no”), aquesta és la opció per defecte.

Un únic element arrel per document

Sempre hi haurà un element que contingui a tota la resta, sent d'aquesta manera una estructura en forma d'arbre.

Tots els elements i atributs han de tenir una sintaxis correcta

- Tots els elements han d'estar delimitats per una etiqueta inicial i un altre final amb el mateix nom. (p.ex. <*element*><*/element*>).

  <element></element>

- Els valors dels atributs XML han d'anar entre cometes simples o dobles.

  <element id="1" id2 ='2'></element>

- Els elements buits han de terminar amb "/" o afegint una etiqueta de fi.

  <element ></element> o bé **<element />**

- XML és sensible a les majúscules i minúscules i, els noms de les etiquetes poden ser alfanumèrics però sempre començats per una lletra.

  <element ></element> no és igual què **<Element></Element >**

Quan un document compleix tots aquests requisits de sintaxi es diu que és un **document XML ben format**.

2.2 Validació d'un document XML

Com hem dit en el punt anterior, els documents XML han de basar-se a la sintaxis definida en l'especificació XML per ser correctes (és el què denominem documents ben formats). Aquesta sintaxis imposa coses com la coincidència de majúscules o minúscules en els noms d'etiqueta, cometes obligatòries per els valors dels atributs, etc. Tot i això, per tenir un control més precís sobre el contingut dels documents és necessari un procés d'anàlisi més exhaustiu. Aquest procés és el què anomenem **validació.**

La validació s'encarrega de verificar:

- **La correcció de les dades.** Encara que validar un document XML no ens garantitza al 100% que les dades són correctes, ens permet detectar en gran mesura els valors incorrectes.

- **La integritat de les dades.** Al validar, es comprova que tota la informació obligatòria està present en el document.

- **L'enteniment compartit de les dades.** A través de la validació es comprova que l'emissor i el receptor perceben el document de la mateixa manera, que l'interpreten igual.

Existeixen varis mètodes per validar un document XML, entre els què destaquen DTD (*Document Type Definition*) i XML Schema.

En el capítol següent s'explicarà amb més profunditat la DTD ja què serà aquest mètode l'utilitzat en el projecte per validar els documents XML.

2.2.1 Característiques generals d'un DTD

El DTD ens permet descriure l'estructura d'un document XML mitjançant etiquetes.

A mode de resum, podem dir que una DTD descriu:

- **Elements:** indiquen quines etiquetes són permeses i el contingut de les mateixes.

- **Estructura:** indica l'ordre en què van les etiquetes en el document.

- **Anidament:** indica quines etiquetes van dintre de les altres.

D'aquesta manera, els documents que s'ajusten a la seva DTD es denominen "vàlids", concepte que no té res a veure amb el d'estar "ben format". Un document "ben format" simplement respecta l'estructura i sintaxis definida per l'especificació XML, en canvi, un document "vàlid" és ben format i, a més, compleix les regles d'una DTD

determinada. Cal afegir també, que un document XML no necessariament ha de tenir una DTD associat.

La DTD es pot especificar de dues maneres diferents:

- Contingut en el propi document XML, com a part de la seva declaració de tipus de document.
- En un fitxer extern, que dóna la possibilitat de compartir aquesta definició amb molts documents XML.

A continuació es mostra un petit exemple:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT
PERSONES(DNI,NOM,COGNOMS,EDAD)>
```

+

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE persones SYSTEM "prueba.dtd">

<persones>
        <dni>53325534</dni>
        <nom>Jose</nom>
        <cognoms>Martinez Fuentes</cognoms>
        <edad>25</edad>
</persones>
```

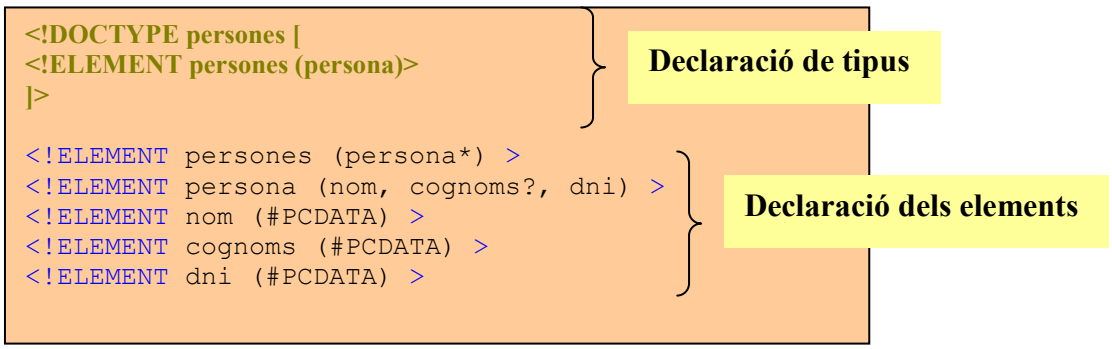
O bé

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE persones [
<!ELEMENT persones (dni,nom,cognoms,edad)>
]>

<persones>
        <dni>53325534</dni>
        <nom>Jose</nom>
        <cognoms>Martinez Fuentes</cognoms>
        <edad>25</edad>
</persones>
```

2.2.2 Sintaxi de les DTD

Per crear una DTD ben formada haurem de tenir per una banda, la declaració del tipus de document i, per un altre, els elements que el composaran.

A continuació mostrem un exemple:



Observant la DTD línia a línia extraiem la següent informació:

1. <*persones*> és un nom d'element vàlid. El símbol * indica que pot haver 0 o més elements persona.
2. <*persona*> és un nom d'element vàlid. Aquest conté obligatòriament els elements <*nom*> i <*dni*> mentre que l'element <*cognoms*> és opcional (indicat amb el símbol *"?"*).
3. <*nom*> és un nom d'element vàlid. Conté caràcters (indicat per l'expressió *(#PCDATA)*).
4. <*cognoms*> és un nom d'element vàlid. Conté caràcters (indicat per l'expressió *(#PCDATA)*).
5. <*dni*> és un nom d'element vàlid. Conté caràcters (indicat per l'expressió *(#PCDATA)*).

Podríem enumerar més elements de la sintaxi de les DTD però per l'elaboració d'aquest projecte crec que el nivell de detall explica't és suficient per a portar-ho a bon terme.

2.2.3 Avantatges i inconvenients de les DTDs

Com hem vist una DTD ens serveix per validar documents XML, però també cal dir que té força limitacions.

A continuació enumerarem les limitacions més importants:

- **No permet definir elements locals que només siguin vàlids dintre d'altres elements.** Per exemple, si volem tenir un element <*manager*> que descrigui el gestor d'una companyia o al d'una delegació, i la definició és diferent en cada cas, amb una DTD tindríem que crear un element per cada

cas, és a dir, crear l'element <*CompanyManager*> y l'element <*DelegationManager*> per evitar el conflicte.

- **Poc flexible en la definició del contingut dels elements.** En una DTD no és possible indicar el tipus de dades dels elements o atributs (si són números, dates, monedes).

A nivell de comentari, direm que per superar aquestes limitacions esmentades, tenim un altre mètode de validació més complert com *XML Schema*.

Cal dir què per el cas particular que ens trobem en aquest projecte, i que serà explica't amb detall més endavant, no necessitarem un nivell de validació més precís que el què ens pugui aportar una DTD.

CAPÍTOL 3: ESTUDI DEL FORMAT OPENDOCUMENT

3.1 Perquè un format lliure de document?

Avui dia, vivim en un món en el què cada vegada més els arxius electrònics substitueixen als documents en paper, per aquest motiu, és fonamental assegurar l'accés i funcionalitat d'aquesta documentació al llarg del temps. Del mateix mode que hi ha múltiples fabricants de paper i bolígrafs, és necessari que varis fabricants suportin i proporcionin aquests formats d'arxius i les aplicacions que creen els formats; això garantitzarà a llarg termini les dades, encara que les empreses desapareguin, canviïn la seva estratègia o pugin els seus preus.

Aquí és on pren rellevància els estàndards oberts dels documents, ja què si són oberts, cap fabricant serà propietari d'aquest format i serà l'usuari que elegirà amb quin producte interaccionarà amb aquell document. A més, si és un format estandarditzat, tenim la tranquil·litat que ha estat supervisat i aprovat per un organisme com potser la ISO (*International Standards Organization*).

També és important destacar què els formats oberts permeten a les empreses interactuar amb més facilitat amb aquests formats ja què tenen tota la documentació necessària disponible i no tenen cap barrera que podria propiciar un format privat. De forma indirecta, això provocarà què per un cost inferior les empreses desenvolupin aplicacions entorn a aquests formats i, fins i tot, software de lliure distribució.

Com es pot veure, un format lliure aporta un gran nombre de avantatges tant per les empreses com pels usuaris finals que utilitzaran els documents de format obert.

Totes aquestes característiques comentades fins aquí les compleix el format OpenDocument (ODF), és obert, és a dir, no hi ha cap empresa propietària del desenvolupament ni de la especificació d'aquest format, i és estandarditzat, tant OASIS com ISO li han donat aquest segell al format.

En el capítol següent es profunditzarà més en aquest format i s'explicaran les característiques principals del mateix.

3.2 Característiques generals d'OpenDocument

3.2.1 Història d'OpenDocument

OpenDocument (també anomenat ODF) té una llarga tradició com a format obert. Els primers esforços de desenvolupament d'aquest format d'arxius es remunta al 1999. Des dels seus inicis, ODF va ser concebut com un format d'arxiu obert i independent de qualsevol implementació.

El procés d'especificació oberta es va iniciar al 2000 amb la fundació del projecte de codi obert *OpenOffice.org* i els esforços de la comunitat en el projecte de desenvolupament XML.

Al 2002 es va crear un nivell més d'obertura del format amb la creació del Comité Tècnic OASIS (*Organization for the Advancement of Structured Information*) i va ser aprovat al maig del 2006 de forma unànime pel Joint Technical Committee 1 (JTC1) de l'organisme ISO i la Comissió Electrònica Internacional (*International Electrotechnical Commission, IEC*) com a Norma Internacional.

Al novembre del 2006, ISO/IEC va anunciar la publicació i disponibilitat de la ISO/IEC 26300:2006 a on es diu que ODF ja està disponible per la seva implementació i ús lliure de llicència, royalties o d'altres restriccions.

A continuació es mostra una taula resum de la història d'aquest format des dels seus inicials fins avui dia.

Data	Fita aconseguida
Principis del 1999	Comença a StarDivision el desenvolupament d'un format d'arxius en XML. La creació d'aquest nou format va ser motivada per les limitacions dels antics formats binaris i la necessitat de suportar Unicode.
Agost del 1999	Sun Microsystems, Inc adquireix StarDivision.
13 d'octubre del 2000	Sun Microsystems, Inc publica el codi obert d'StarOffice sota llicències obertes en el projecte OpenOffice.org recentment fundat (juliol 2000). S'estableix en OpenOffice.org el projecte comunitari XML amb l'objectiu de definir l'especificació del format d'arxius XML com un esforç de la comunitat oberta.
2002	Les definicions pel Xinés, japonès i coreà i altres representacions complexes s'afegeixen a l'especificació del format OpenOffice.org XML. S'inicia la col·laboració amb el projecte KOffice.
16 de desembre del 2002	L'OASIS convoca la seva primera conferència.
Maig del 2002	Es publica OpenOffice.org 1.0 i StarOffice 6. Tots dos utilitzen el format d'arxius OpenOffice.org
Agost del 2003	KOffice decideix utilitzar ODF com a format predeterminat.

2003-2004	Es modifica l'especificació original del format OpenOffice.org XML per reflectir els últims desenvolupaments en XML i en l'àrea de les aplicacions ofimàtiques: - Introducció dels espais de noms conforme a les regles de denominació d'OASIS. - Canvi de les DTDs de XML per Relax-NG com a llenguatge de validació. - Millores en l'esquema per suportar millor la validació de documents. - Adaptació de l'esquema per a noves versions estàndards. - Adaptacions per a noves aplicacions ofimàtiques (KOffice). - Adaptacions per a noves versions d'aplicacions ofimàtiques (OpenOffice.org 2.0). - Eliminació d'inconsistències a l'especificació. - Correcció d'errors.
Desembre del 2004	S'aprova el segon esborrany del comitè, el seu títol canvia de "*Oasis Open Office Specification*" a "*OASIS Open Document Format Office Applications*" (OpenDocument).
Gener del 2005	El comitè tècnic canvia el nom a OASIS Open Document Format for Office Applications (Open Document).
Febrer del 2005	El tercer esborrany de l'especificació del format, incloent les reaccions de l'avaluació pública, s'aprova com a esborrany del comitè.
Maig del 2005	L'OpenDocument Format (ODF) s'aprova com a estàndard d'OASIS.
Setembre del 2005	Sun Microsystems llança StarOffice 8 amb suport ODF. ODF s'envia a l'Organització Internacional per a la Normalització (ISO).
Octubre del 2005	Es llança OpenOffice 2.0 amb suport ODF.
Desembre del 2005	Softmaker llança Textmaker amb suport a ODF.
Gener del 2006	IBM llança IBM Workplace amb suport ODF.
Març del 2006	Es funda la ODF Alliance amb 35 membres inicials per promoure ODF al sector públic.
Abril del 2006	Surt al mercat KOffice 1.5 que utilitza ODF com a format predefinit.
Maig del 2006	ISO aprova ODF com ISO/IEC 26300.
Juny del 2006	La ODF Alliance ja té més de 200 membres entre els que destaquen empreses i organitzacions com IBM, Novell, Red Hat, Oracle, Software AG, Sun Microsystems.
2007	S'aprova ODF 1.2.

3.2.2 Què és i què ens aporta OpenDocument

El format de document obert (anomenat també OpenDocument o ODF) és un format d'arxiu basat en XML per aplicacions ofimàtiques destinat a la creació i edició de documents que continguin text, fulles de càlcul, base de dades, gràfics i presentacions.

Al ser un format obert, ODF permet als usuaris finals, accedir a la informació i compartir-la sense tenir que pagar cap mena de llicència per poder veure o editar informació com passa amb els formats propietaris.

Tant organitzacions com individus particulars poden utilitzar qualsevol aplicació per el processament de text, el què evita tenir que dependre tota la vida d'un producte en concret que ens permeti l'accés a aquesta informació. Per aquest motiu, és un estàndard de documentació electrònica que s'utilitza cada vegada més en l'Administració Pública.

3.3 Arquitectura d'OpenDocument

El format ODF va ser dissenyat per ser independent del fabricant i de la implementació; i per ser utilitzat pel major nombre possible d'aplicacions.

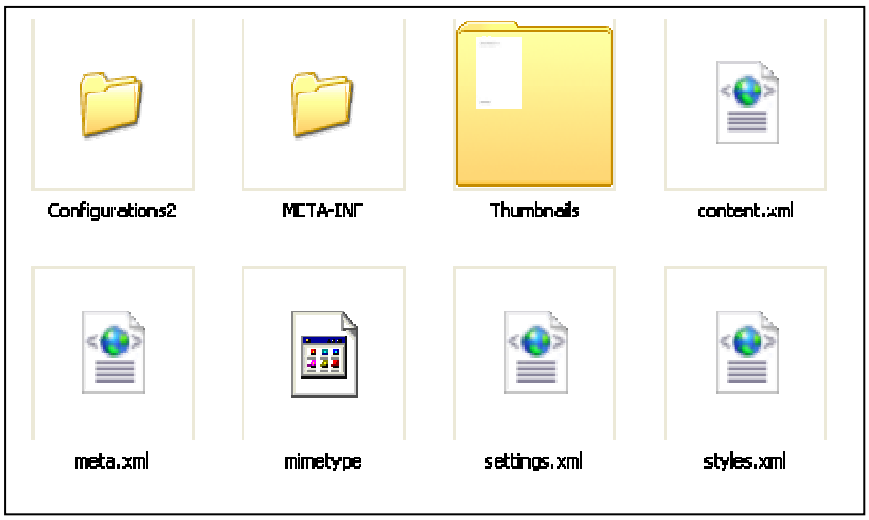
Per acomplir amb aquests requeriments i maximitzar la interoperabilitat entre aplicacions, el format reutilitza estàndards ja establerts com XML, XHTML, SVG, XSL, Xlink, Xforms, MathML.

ODF té diversos tipus de fitxers, entre els què destaquen:

Tipus de fitxer	Extensió del fitxer
Text	.odt
Full de càlcul	.ods
Presentació	.odp
Base de dades	.odb
Gràfics/Dibuixos	.odg
Fórmules matemàtiques	.odf

Tots aquests tipus comparteixen la mateixa estructura interna, on la seva característica principal és que tot document és un fitxer comprimit (en format zip) que conté un conjunt de fitxers XML que composen el document que visualitzem.

Document



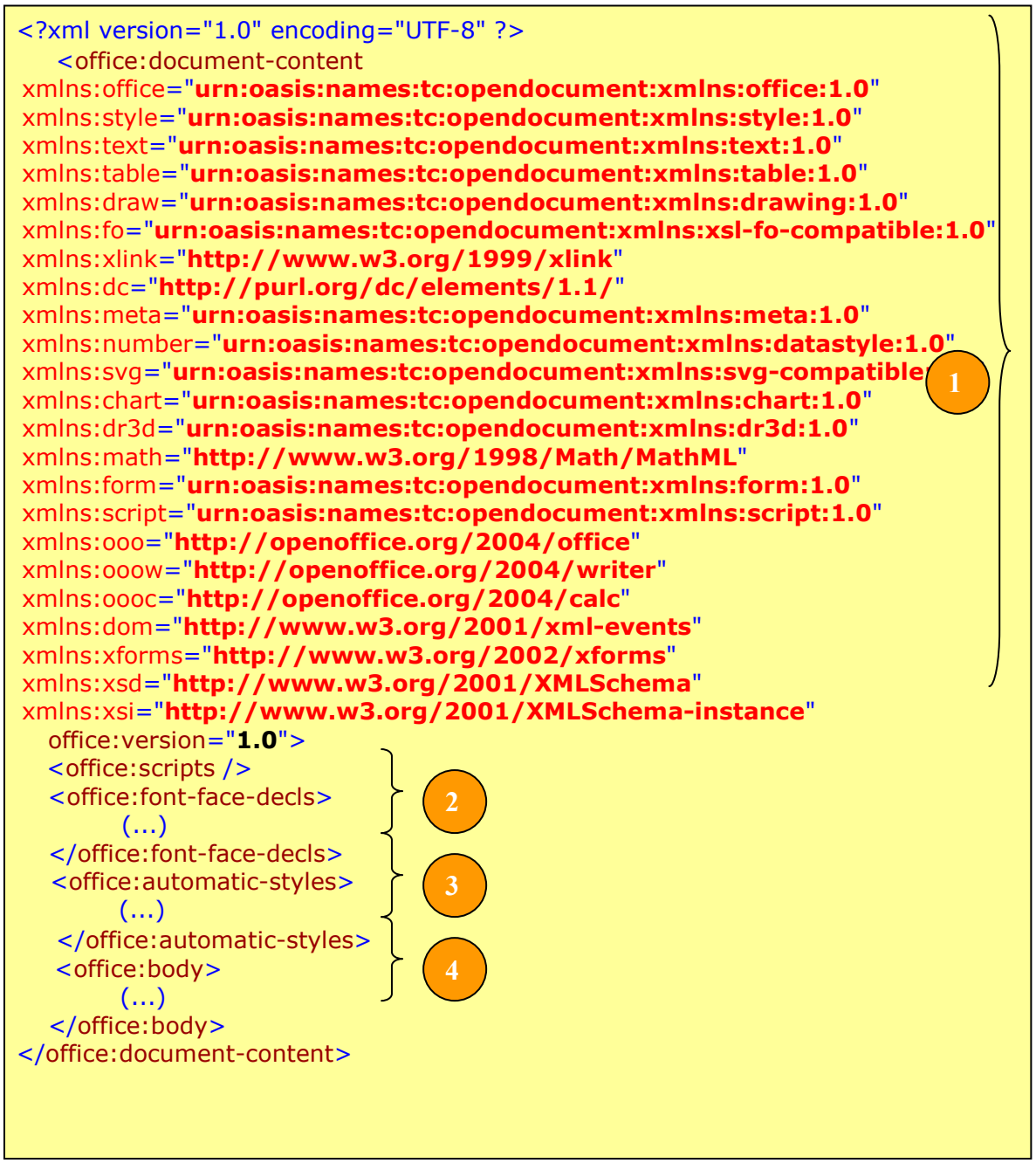
Estructura del documento descomprimit

El format OpenDocument ofereix una clara separació entre el contingut, la disposició d'aquest en el document i les metadades. Els components més notables del format s'explicaran en els punts posteriors.

3.3.1 Contingut del document. Content.xml

Aquest és el fitxer més important. Emmagatzema el contingut real del document (excepte les dades binàries, com per exemple, les imatges). El format es basa en el llenguatge de marques HTML, encara què és bastant més complex.

Aquest document està format de varies seccions que determinaran el contingut i l'estil del document. Per veure-ho més clar, mostrarem un esquema amb el seu contingut.



Esquema de contingut del fitxer content.xml

1. En aquesta secció trobem la declaració dels diferents espais de noms que s'utilitzen en el document. Per exemple, l'espai de noms xmlns:text s'utilitza per definir elements de text en el document, com potser un paràgraf (text:p).

2. Aquí trobem la declaració dels diferents tipus de font del document.

3. Aquesta secció correspon a la declaració dels estils que s'utilitzaran al document, per exemple, <style:style style:name="**P2**" style:family="**paragraph**" style:parent-style- name="**Standard**">, indica que l'element del document amb nom P2 tindrà un estil de la família "paragraph" i que hereta de l'estil amb nom "Standard". A la vegada aquests estils fan referència als definits al fitxer *styles.xml* i serà aquí on trobarem els detalls de l'estil (tipus de lletra, tamany, estil...).

4. En aquesta secció trobem el cos del document. A continuació es mostra un petit exemple de contingut:

```
<office:body>
 <office:text>
   <text:p text:style-name="Standard">Omplenem el contingut d'un
        paràgraf.</text:p>
   <text:p text:style-name="P1">Aquí tenim un altre paràgraf però amb
        negreta</text:p>
    <text:p text:style-name="P1" />
    <text:p text:style-name="P2">
        Ara posarem una nota a peu de pàgina
       <text:note text:id="ftn0" text:note-class="footnote">
          <text:note-citation>1</text:note-citation>
          <text:note-body>
              <text:p text:style-name="Footnote">Text de la nota a peu de
                   pàgina.</text:p>
          </text:note-body>
       </text:note>
    </text:p>
    <text:p text:style-name="Standard" />
    <text:p text:style-name="Standard" />
 </office:text>
</office:body>
```

Amb aquest exemple podem veure que l'estructura interna del cos del document no és senzilla ja què aquest fragment XML correspon només a 4 línies d'un document.

Per aquest motiu, he cregut convenient només explicar els elements més representatius del contingut què, d'altra banda, han estat els necessaris per elaborar el programari resultant d'aquest estudi.

A continuació es mostren les característiques principals d'aquests elements:

Paràgraf

Dintre d'un document, l'element més important és el paràgraf (<text:p>). Aquest element conté un conjunt de caràcters o d'altres elements que tenen les mateixes propietats (tipus de lletra, color…).

Taula

L'element que representa una taula és <table:table>. Les taules estan formades de files (<table:table-row>) i cada fila de cel·les (<table:table-cell>). Dintre de cada cel·la podem trobar paràgrafs que continguin text o d'altres elements.

Llista

Les llistes estan formades per les enumeracions i vinyetes. Per exemple, les enumeracions (<text:list>) estan formades per cadascú dels ítems (<text:list-item>) i dintre de cada ítem podem trobar paràgrafs o d'altres elements.

Nota a peu de pàgina

Les notes a peu de pàgina fan referència a elements del text, per tant, estaran dintre d'un paràgraf (què són els elements que contenen text). Les notes estan formades per 3 elements:

1. <text:note text:id="**ftn0**" text:note-class="**footnote**"> → aquest element indica una nota a peu de pàgina amb identificador "ftn0" i del tipus nota a peu de pàgina (footnote).
2. <text:note-citation>**1**</text:note-citation> → aquest element indica la numeració de la nota, en aquest cas 1.
3. <text:note-body> → el contingut d'aquest element serà el text de la nota a peu de plana.

3.3.1.1 Assignació d'estils als elements

Tots els elements del document tenen un estil marcat (tipus de lletra, negreta, cursiva…). Potser que sigui un estil implícit, quan l'estil que utilitza és el predeterminat, llavors simplement fa referència a aquest estil, l'estàndard.

```
<text:p text:style-name="Standard">Omplim el contingut d'un
 paràgraf.</text:p>
```

Per una altra banda, podem tenir que un element tingui un estil en concret, per exemple, amb una font més gran i amb negreta. En aquest cas, es farà referència a l'estil que implementi aquestes opcions.

```
<text:p text:style-name="P1">Aquí tenim un altre paràgraf però amb
negreta</text:p>

<style:style style:name="P1" style:family="paragraph" style:parent-style-
name="Standard">
        <style:text-properties fo:font-weight="bold" style:font-weight-
        asian="bold" style:font-weight-complex="bold" />
</style:style>
```

En aquest exemple veiem què el paràgraf fa referència a l'estil **P1** què està definit a la secció d'estils del document. A la vegada, l'estil fa referència a una família d'estils (paragraph), en canvi, aquest estarà definit en el fitxer *styles.xml* perquè és un estil genèric.

3.3.2 Altres fitxers

Styles.xml

En aquest fitxer és a on es troben definits gran part dels estils del document (encara que podem trobar-hi també en el document *content.xml*). D'aquesta manera no tindrem que repetir l'especificació de l'estil cada vegada que l'utilitzem, sinó que simplement haurem de tenir una referència a ell.

Mostrem un tros de codi XML per exemplificar l'estructura d'aquest fitxer:

```
<style:default-style style:family="paragraph">
    <style:paragraph-properties style:text-autospace="ideograph-alpha"
        style:punctuation-wrap="hanging" style:line-break="strict" style:writing-
        mode="page" />
    <style:text-properties fo:color="#000000" style:font-name="Times New Roman"
        fo:font-size="12pt" fo:language="en" fo:country="US" style:font-name-
        asian="Arial Unicode MS" style:font-size-asian="12pt" style:language-
asian="en"
        style:country-asian="US" style:font-name-complex="Tahoma" style:font-size-
        complex="12pt" style:language-complex="en" style:country-complex="US" />
</style:default-style>
```

Aquí es mostra el contingut de la definició de la família d'estils *paragraph* on es mostra el color i el tipus de font, la grandària… Aquest estil el poden heretar diferents elements del document amb l'avantatge que només està definit en un sol lloc.

Meta.xml

Aquest fitxer conté les metadades del document. Aquesta informació és la representada per l'autor del document, la identificació de l'última persona què el va modificar, la data d'última modificació, etc.

Un exemple de contingut seria el següent:

```
<office:meta>
  <meta:generator>OpenOffice.org/2.2$Win32
    OpenOffice.org_project/680m14$Build-9134</meta:generator>
  <meta:creation-date>2007-10-08T22:28:10</meta:creation-date>
  <dc:date>2007-10-12T18:14:47</dc:date>
  <dc:language>en-US</dc:language>
  <meta:editing-cycles>2</meta:editing-cycles>
  <meta:editing-duration>PT1M58S</meta:editing-duration>
  <meta:user-defined meta:name="Info 1" />
  <meta:user-defined meta:name="Info 2" />
  <meta:user-defined meta:name="Info 3" />
  <meta:user-defined meta:name="Info 4" />
  <meta:document-statistic meta:table-count="1" meta:image-count="0"
    meta:object-count="0" meta:page-count="1" meta:paragraph-count="6"
    meta:word-count="33" meta:character-count="172" />
</office:meta>
```

Els elements més representatius són:

<meta:creation-date> → Data de creació del document.

<meta:generator> → Programari què ha generat el document.

<meta:editing-duration> → Temps de durada de l'última edició.

<meta:document-statistic → Estadístiques del document (número de paràgrafs, de caràcters, de paraules, etc).

Settings.xml

Aquest fitxer inclou les propietats del document. Podem trobar el factor de zoom o la posició inicial del cursor en el moment d'obrir el document. Aquestes propietats no afecten ni al contingut ni a la disposició d'aquest en el document.

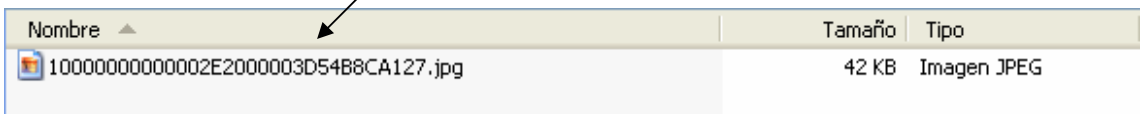
Mimetype

El fitxer *mimetype* conté una única línia que indica el tipus MIME del document.
La utilitat d'aquest fitxer és que l'extensió del fitxer no és la què determina el seu tipus sinó que s'imposa la informació del document *mimetype*. Per exemple, en un fitxer OpenDocument de text (amb extensió odt) el seu tipus mime és *application/vnd.oasis.opendocument.text*.

Carpeta Pictures

Aquesta carpeta conté totes les imatges del document. Igual que passa amb els estils, en el fitxer *content.xml* no es troba físicament el contingut de la imatge sinó que hi haurà una referència a un fitxer d'aquesta carpeta. A continuació es mostra un exemple:

```
<text:p text:style-name="Standard">
    <draw:frame draw:style-name="fr1" draw:name="graphics1" text:anchor-
      type="paragraph" svg:width="8.149cm" svg:height="10.834cm"
      draw:z-index="0">
      <draw:image
        xlink:href="Pictures/10000000000002E2000003D54B8CA127.jpg"
        xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad" />
    </draw:frame>
</text:p>
```



3.4 Llibreries de suport a format.

Hem pogut veure que l'arquitectura del format OpenDocument és força complex, que està formada per diversos fitxers relacionats i dintre de cadascun d'ells una estructura XML complicada.

Tal com s'ha explicat al principi del document, hem de desenvolupar una aplicació que tracti i modifiqui un document d'aquest format, cosa que sembla, a priori, certament complicat. Per una part, tindrem que descomprimir el document per poder tractar els fitxers que conté i, per un altre, modificar l'estructura XML del document *content.xml*

Per aquest motiu, he cregut convenient investigar si la comunitat *OpenOffice.org* proporcionava alguna llibreria de suport a aquest format. Després de realitzar la cerca dintre d'aquesta pàgina vaig trobar dos llibreries, una per a desenvolupadors de Java (odf4all) i un altre per desenvolupadors de .NET (AODL).

Per motius de disseny i implementació, que s'explicaran al tema d'Enginyeria del Software he elegit la llibreria AODL com a suport per desenvolupar aquest projecte.

A continuació s'explica l'arquitectura i les característiques més importants d'aquesta llibreria de lliure distribució.

3.4.1 AODL. An Open Document Library

A finals del 2005 un equip de desenvolupament, liderat per Lars Behrmann, van crear la llibreria de lliure distribució AODL. La seva finalitat era dotar als desenvolupadors de .NET d'una eina que els facilités la manipulació de documents ODF.

Aquesta llibreria va anar evolucionant fins que al març del 2007, aquest desenvolupament es va fusionar amb el projecte *ODF Toolkit Project* dintre de la comunitat *OpenOffice.org*. A partir d'aquest moment, es va avançar molt en el seu desenvolupament juntament amb la publicació de nombrosos exemples d'ús.

A finals del mes d'agost es va lliurar la versió 1.3 que donava la possibilitat, entre d'altres funcionalitats afegides, d'accedir a les capçaleres i peus de pàgina dels documents.

Més endavant, en l'apartat Disseny de la solució, del capítol Enginyeria del Software, s'explicarà una mica més en detall l'arquitectura interna d'aquesta llibreria i els objectes més importants i necessaris per la seva utilització.

3.5 Altres formats oberts de documents.

Encara què aquest tema es centra en l'estudi del format OpenDocument és interessant veure les diferents alternatives de formats oberts i estàndards de documents a més d'ODF.

3.5.1 Portable Document Format. PDF

El format PDF (de l'anglès Portable Document Format) va ser creat per l'empresa *Adobe Systems*. Aquest tipus de document està especialment ideat per documents susceptibles de ser impresos, ja què especifica tota la informació necessària per la presentació final del document.

Les característiques principals d'aquest format són les següents:

- **Multiplataforma.** Pot ser presentat per els diferents sistemes operatius del mercat (Windows, Unix/Linux o Mac).
- **Popularitat.** És un dels formats més estesos a Internet per l'intercanvi de documents.
- Poder **xifrar** i **firmar digitalment** els documents.
- Pot **incloure imatges, gràfics i text**.
- És l'**estàndard ISO** (ISO 19005-1:2005) per fitxers contenidors de documents electrònics en vistes a la preservació de llarga duració.

A més de les característiques esmentades, hem de destacar el fet diferencial respecte a ODF, l'especificació del format PDF és propietari de l'empresa Adobe. Aquest fet fa què encara que tothom pugui obtenir un visor gratuït per veure documents PDF, si vol editar o crear un document d'aquest format necessiti una eina de pagament d'Adobe per realitzar-ho.

3.5.2 Microsoft Office 2007. OpenXML

Fa un any aproximadament Microsoft va treure al mercat la nova versió de la seva suite ofimàtica, *Office 2007*. Aquest producte ha volgut canviar la seva filosofia respecte a les versions anteriors, ja què havia rebut nombroses crítiques per l'hermetisme en l'especificació i arquitectura dels seus documents.

L'arquitectura dels nous documents d'Office es basen en l'especificació OpenXML que es fonamenta en la ja existent ODF. Els seus documents també són paquets zip composats de fitxers XML. Fins aquest punt, podríem pensar que Microsoft, ha obert el seu format de documents per tal què ODF no sigui una veritable amenaça en quan a quota de mercat. Per desgràcia això està molt lluny de la realitat, ja què una cosa és el què ens volen fer creure i un altre la realitat de la situació. Un exemple clar és que Microsoft no ha aconseguit què la seva especificació sigui aprovada per l'organisme ISO, cosa que posa en dubte la credibilitat de Microsoft quan diu que OpenXML és un format obert.

En el punt següent esmentarem les diferències principals entre OpenDocument i OpenXML que, indirectament, ens faran veure perquè el format de Microsoft no ha aconseguit l'aprovació ISO.

3.5.3 OpenXML vs OpenDocument

Com ja hem comentat en el punt anterior sembla que OpenXML és un format què ha sorgit per fer competència a ODF per això volem esmentar les diferències més importants entre els dos i, d'aquesta manera, justificar el perquè ODF és un format obert i estandarditzat i OpenXML no.

OpenDocument	OpenXML
Especificació de la seva arquitectura i format exposat en unes 700 pàgines.	La seva especificació està documentada en més de 6000 pàgines. A més, aquesta especificació és incomplerta perquè molts punts d'aquesta fan referència a documentació propietària i privada de Microsoft.
ODF va ser aprovat com estàndard ISO 26300 el dia 1 de maig del 2006.	OpenXML va ser rebutjat com estàndard ISO 29500 a principis del mes de setembre.
Aquest format porta més de 7 anys desenvolupant-se i madurant.	El format és força nou, va aparèixer a l'any 2006.
Hi ha nombroses aplicacions com Koffice, OpenOffice, de lliure distribució que permeten crear i editar els documents amb aquest format. A més aquestes aplicacions estan disponibles en Windows, Unix/Linux i Mac.	OpenXML només funciona a través de la suite ofimàtica Office 2007, només disponible per Windows XP i Vista i, a més, és la suite ofimàtica més cara del mercat.
És un format 100% XML.	No és un format 100% XML ja què té, per exemple, sistemes de codificació binària propietàries.

Aquestes només són el recull de les diferències més significatives dels dos formats a més de ser els punts principals pel qual va ser rebutjada la candidatura del format OpenXML com a format estàndard per la ISO.

CAPÍTOL 4: ENGINYERIA DEL SOFTWARE

4.1 Anàlisi de requeriments.

4.1.1 Perfil d'usuari

És important saber quin tipus d'usuari utilitzarà el programari resultant d'aquest projecte ja què una part important de la implementació, la capa de presentació, dependrà en gran mesura d'aquest aspecte.

En el cas particular que tractem hem de dir que no hi ha cap requeriment explícit ni de quin tipus d'usuari ni, fins i tot, si ha de ser un procés batch o ha de tenir una interfície per interactuar amb l'usuari final.

Donada la poca informació en l'enunciat del PFC sobre aquest aspecte, la meva decisió ha estat crear una aplicació amb una interfície per tal de facilitar la tasca a l'usuari. D'aquesta manera ens assegurem què tant usuaris experts com principiants seran capaços d'utilitzar aquest programari. Per tant, serà una capa de presentació fàcil d'entendre, sense presentar aspectes tècnics per pantalla, en definitiva, una interfície senzilla per qualsevol tipus d'usuari.

4.1.2 Requeriments funcionals

Aquest apartat té la missió d'aconseguir extreure de l'enunciat del PFC les funcionalitats més importants què haurà de realitzar el programari a desenvolupar.

Una vegada estudiada tota la informació descrita anteriorment, queda el pas de realitzar un llistat de totes les funcionalitats principals què haurà de realitzar l'aplicació:

Realitzar esquema del XML d'entrada i la seva DTD de validació

Hem de dissenyar l'estructura del fitxer XML d'entrada de dades. Aquest XML ens permetrà subministrar a l'algoritme central de l'aplicació les dades per inserir les referències bibliogràfiques en el document.

Aquest fitxer ha de permetre que les dades d'entrada tinguin l'estructura d'una llista de parelles (conjunt de termes – referència), on cada referència tindrà les dades suficients i necessàries.

Per tal què l'esquema del XML generat sigui ben format i vàlid s'ha de realitzar una DTD que verifiqui la seva estructura jeràrquica.

Realitzar l'algoritme d'inserció de les referències en el document

Aquesta serà la part central del desenvolupament i la més laboriosa. Caldrà realitzar el següent procés automàtic en un document ODF:

- Afegir, **el primer cop** que aparegui un terme en el document OpenOffice original, la referència que tingui associada al peu de la pàgina.
- Afegir al final del document una pàgina amb les referències que s'han utilitzat en el document.

4.1.3 Requeriments no funcionals

Com a requeriments no funcionals s'indicarà la forma en què es mostra la informació a l'usuari en cada moment.

- Abans de començar el procés automàtic de modificació del document es demanarà confirmació a l'usuari.
- Mostrar al final del procés si aquest ha acabat de forma correcte o bé ha passat algun error. Si no s'ha acabat correctament tot el procés no es farà cap modificació del document.

4.1.4 Arquitectura tècnica

En aquest punt es comenta l'arquitectura tècnica que s'ha elegit per desenvolupar l'aplicació i el perquè d'aquesta decisió.

Llenguatge de programació

De tots és ben sabut què existeixen gran quantitat de llenguatges de programació cadascun amb les seves avantatges i desavantatges. La decisió de quin d'ells utilitzar és complexa i es basa en les necessitats de l'usuari final i del desenvolupador.

Com a primera opció hi havia utilitzar el Java però vaig declinar aquesta alternativa a favor d'utilitzar el C#. Els motius principals pel qual em vaig decidir per utilitzar aquest llenguatge d'implementació van ser els següents:

- La meva experiència professional en aquest llenguatge m'ha permès valorar amb més precisió l'estimació temporal de la implementació.

- És un llenguatge que es pot utilitzar sense cap cost econòmic. El desenvolupador pot utilitzar una eina gratuïta que proporciona Microsoft què és el *Microsoft C# Express*, una versió reduïda però totalment funcional de la què podem obtenir amb el *Visual Studio 2005* (aquesta si què és de pagament).

- El client que utilitzi aquest programari només tindrà que instal·lar-se el Framework 2.0 per fer-lo anar. L'arquitectura dels llenguatges de la família .NET segueix la mateixa filosofia que Java, és a dir, el codi quan el compilem es genera en un llenguatge entremig (IL) que és independent al processador en què es vol executar. Posteriorment, quan l'aplicació es vol executar en el client, el Framework (l'equivalent a la màquina virtual de Java) realitza la compilació Just-in-Time.

- Per motius professionals ja vaig donar una ullada fa temps a la llibreria AODL que s'utilitzarà com a suport a la implementació de la solució, per tant, ja he partit d'un coneixement previ molt útil per aquest projecte.

- Facilitat d'implementació de la interfície gràfica. Ja què la part principal del projecte no era la realització de la capa d'implementació era molt útil utilitzar un entorn de programació què em permetés dissenyar una interfície senzilla de forma ràpida, per poder centrar tot l'esforç en el desenvolupament de l'algoritme de tractament.

Sistema operatiu

Encara que avui dia hi ha projectes open source (Mono project) que desenvolupen un Framework per consumir aplicacions realitzades en .NET en sistemes operatius no Windows (Linux/Unix), el Framework 2.0 que es pot baixar de forma gratuïta de la plana de Microsoft només es pot executar sota un sistema operatiu Windows.

Software necessari

El Framework 2.0 es pot descarregar de la pàgina de Microsoft de la següent adreça:

http://www.microsoft.com/downloads/details.aspx?FamilyID=7abd8c8f-287e-4c7e-9a4a-a4ecff40fc8e&displaylang=es

Una vegada descarregat el fitxer simplement es fa doble click sobre el mateix i es segueixen les indicacions de la instal·lació.

Cal tenir en compte que el Framework 2.0 requereix un programari mínim en la màquina en què s'instal·larà:

- Windows Installer 3.0 o posterior (excepte per Windows 98/ME, que requereix Windows Installer 2.0 o una versió posterior).
- Internet Explorer 5.01 o versió posterior.

4.2 Disseny.

En aquest apartat explicarem el disseny de les diferents parts del projecte.
Aquesta fase es divideix en dues parts ben diferenciades, per una banda, la creació del fitxer XML d'entrada amb la seva DTD de validació i, per un altre, el disseny del programari necessari per manipular els documents ODF.

4.2.1 Disseny del fitxer XML d'entrada

Tal com s'especifica en els requeriments, s'ha de crear l'estructura d'un fitxer XML per guardar les dades d'entrada, en el nostre cas, un conjunt de parelles (termes-referència).

El primer pas, ha estat investigar sobre el format què seguia una referència bibliogràfica i en aquest procés, el primer que s'ha observat és que hi han de diferents tipus, com per exemple, les referències a llibres, articles, articles d'Internet... De tots aquest tipus de referències s'ha pogut establir una sèrie de dades comunes a totes elles i un altre tipus de dades més específiques de cada format, d'aquesta manera es podran plasmar en el document d'entrada diferents tipus de referències.

Arribat a aquest punt ja només queda mostrar el format de referència empleat en aquest treball:

Autor/s. Títol. Edició. Data de publicació. [Lloc de publicació]. [pàgines]. [pàgines consultades]. [ISBN]. [informació d'una referència web].

Aquesta seria l'estructura d'una referència, on les dades opcionals han estat marcades entre claudàtors. Per entendre millor tota la informació que pot contenir una referència, es detallen cadascuna de les dades:

- **Autor**: Serà l'autor de l'obra i aquest atribut estarà composat de nom y cognoms. També indicar que una obra pot tenir 1 o n autors.

- **Títol**: Títol de l'obra o article.

- **Edició**: Edició de l'obra o article.

- **Data de publicació**: Data en què es va realitzar la publicació.

- **Lloc de publicació:** Localitat, Ciutat o País de publicació.

- **Pàgines**: Número de pàgines que té l'obra o l'article (atribut opcional).

- **ISBN**: International Stàndard Book Number o també dit com l'estàndard internacional de numeració de llibres (atribut opcional).

- **Informació d'una referència web:** Aquestes dades es divideixen en:

    - **Url:** Uniform Resource Location o dit col·loquialment, la direcció web a on es troba el recurs al què fem referència.

- **Data de cerca:** Data en què es va buscar aquesta referència a Internet.

NOTA: És opcional indicar les dades d'una referència web però si s'indica és obligatori tant la url com la data de cerca.

Ara què ja tenim una visió més precisa de les dades que podrem incloure en una referència ens queda passar aquesta estructura en format de document XML.

El disseny proposat és el següent:

```
<bibliography>
    <reference>
        <terms>
            <term /></term>
            <term /></term>
            <term /></term>
        </terms>
        <info>
            <authors>
                <author>
                    <firstName />
                    <lastNames />
                </author>
            </authors>
            <title />
            <edition />
            <datePublish />
            <placePublish />
            <pages />
            <location />
            <ISBN />
            <online >
                <url />
                <dateSearch />
            </online >
        </info>
    </reference>
</bibliography>
```

L'element arrel és *bibliography* i serà el què contindrà totes les referències disponibles en el fitxer d'entrada. Dintre d'aquest element tenim *reference* què per una part estarà format per la llista de termes que estaran relacionats amb aquella referència (*terms*) i, per un altre, la informació de la referència.

A continuació es mostra una taula on es detalla la descripció de cadascun del elements:

Element	Descripció
<bibliography>	Llista de parelles (termes-referència).
<reference>	Parella de termes-referència.
<terms>	Llista de termes associats a una referència
<term>	Terme associat a una referència
<info>	Dades d'una referència
<authors>	Llista d'autors
<author>	Autor
<firstName>	Nom de l'autor

<lastNames>	Cognoms de l'autor
<title>	Títol
<edition>	Edició de l'obra
<datePublish>	Data de publicació
<placePublish>	Lloc de publicació
<pages>	Pàgines totals de l'obra
<location>	Conjunt de pàgines a les què es fan referència
<ISBN>	International Stàndard Book Number
<online>	Dades d'una referència web
<url>	Adreça web
<dateSeach>	Data en què es va cercar la referència web

Una vegada tenim l'estructura del fitxer XML, per tant, un document ben format, ens queda validar-ho a través d'una DTD.

Tal com hem explicat al capítol 2 hi han dues maneres d'indicar una DTD en un fitxer XML, de manera interna o externa. En el cas què tractem, s'ha decidit externalitzar la DTD, d'aquesta manera l'usuari podrà crear tants fitxers d'entrada com vulgui i tots faran referència a la mateixa DTD. Gràcies a aquesta elecció no es tindrà que arrossegar la DTD de validació en tots els fitxers d'entrada que es creïn.

L'única indicació que haurà de tenir en compte l'usuari alhora de realitzar el fitxer d'entrada és recordar-se d'indicar que el seu fitxer XML es validarà amb una DTD. Per realitzar-ho, simplement haurà d'indicar aquesta línia a la capçalera del fitxer XML d'entrada:

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

Ara ja només queda indicar el disseny de la DTD que valida el document XML de referències bibliogràfiques:

```
<!ELEMENT bibliography (reference+)>
<!ELEMENT reference (terms,info)>
<!ELEMENT terms (term+)>
<!ELEMENT term (#PCDATA)>
<!ELEMENT info
(authors,title,edition,datePublish,placePublish,pages?, location?,
ISBN?,online?)>
<!ELEMENT authors (author+)>
<!ELEMENT author (firstName,lastNames)>
<!ELEMENT firstName (#PCDATA)>
<!ELEMENT lastNames (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT datePublish  (#PCDATA)>
<!ELEMENT placePublish (#PCDATA)>
<!ELEMENT pages (#PCDATA)>
<!ELEMENT location (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT online (url,dateSearch)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT dateSearch (#PCDATA)>
```

A continuació es detalla l'especificació d'aquesta DTD:

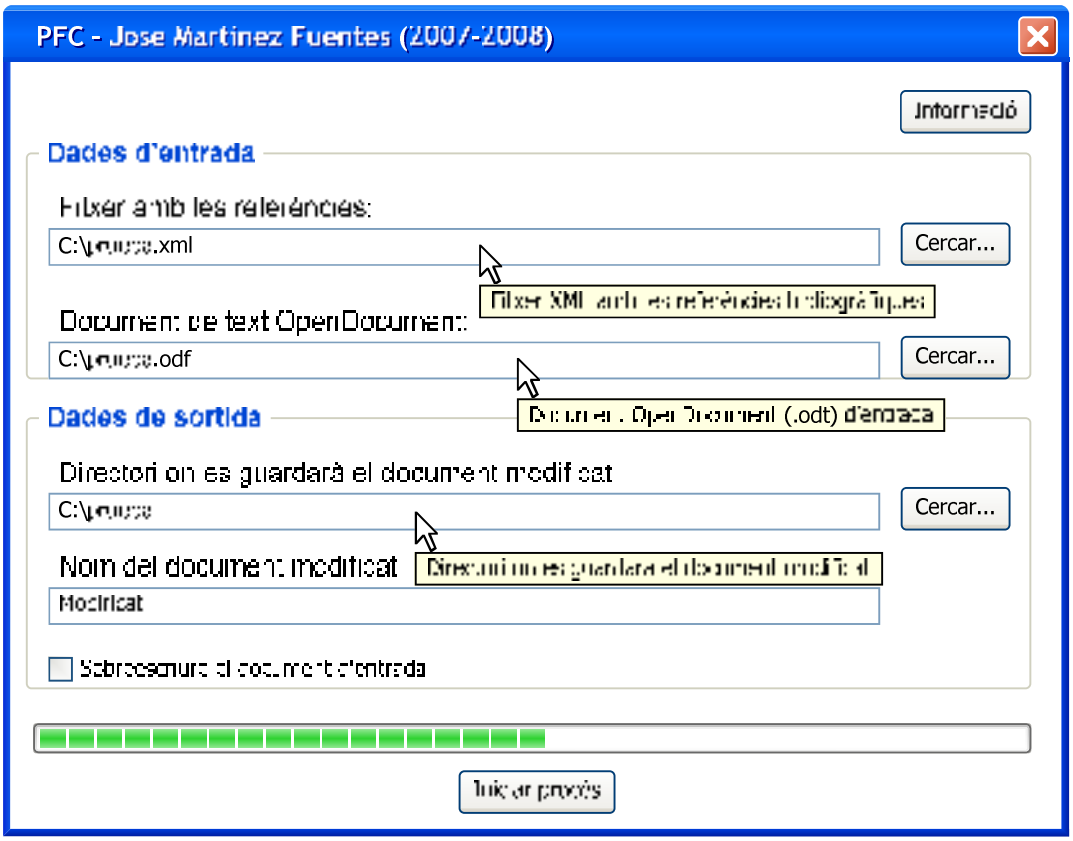
Element	Descripció
<!ELEMENT bibliography (reference+)>	L'element *bibliography* pot contenir *n* elements reference.
<!ELEMENT reference (terms,info)>	L'element *reference* conté l'element *terms* i l'element *info*.
<!ELEMENT terms (term+)>	L'element *terms* pot contenir *n* elements *term*.
<!ELEMENT term (#PCDATA)>	El contingut de l'element *term* serà text.
<!ELEMENT info (authors,title,edition,datePublish , placePublish, pages?, ISBN?, location?,online?)>	L'element *info* contindrà els elements *authors, title, edition, datePublish* i *placePublish*. I opcionalment podrà tenir l'element *ISBN,location, pages* o *online*.
<!ELEMENT authors (author+)>	L'element *authors* pot contenir *n* elements *author*.
<!ELEMENT author (firstName,lastNames)>	L'element *author* conté els elements *firstName* i *lastNames*.
<!ELEMENT firstName (#PCDATA)>	El contingut de l'element *firstName* serà text.
<!ELEMENT lastNames (#PCDATA)>	El contingut de l'element *lastNames* serà text.
<!ELEMENT title (#PCDATA)>	El contingut de l'element *title* serà text.
<!ELEMENT edition (#PCDATA)>	El contingut de l'element *edition* serà text.
<!ELEMENT datePublish (#PCDATA)>	El contingut de l'element *datePublish* serà text.
<!ELEMENT placePublish (#PCDATA)>	El contingut de l'element *placePublish* serà text.
<!ELEMENT pages (#PCDATA)>	El contingut de l'element *pages* serà text.
<!ELEMENT location (#PCDATA)>	El contingut de l'element *location* serà text.
<!ELEMENT ISBN (#PCDATA)>	El contingut de l'element *ISBN* serà text.
<!ELEMENT online (url,dateSearch)>	L'element *online* conté els elements *url* i *dateSearch*.
<!ELEMENT url (#PCDATA)>	El contingut de l'element *url* serà text.
<!ELEMENT dateSearch (#PCDATA)>	El contingut de l'element *dateSearch* serà text.

4.2.2 Disseny de la interfície de l'aplicació

Com ja s'ha esmentat en els requeriments, és important dissenyar una interfície senzilla d'utilitzar ja què, en principi, ha d'estar adaptada per usuaris de tot tipus.

La capa de presentació estarà formada per un únic formulari. Aquest, demanarà les dades d'entrada necessàries per executar l'aplicació (XML amb les referències i ubicació del document ODF). Per una altra banda, tindrà un botó d'execució què serà l'encarregat de posar en marxa el procés automatitzat de modificació del document en funció de les dades d'entrada.

El disseny proposat per aquest formulari és el següent:



Com es pot veure és una interfície molt senzilla per l'usuari tot i així s'hauran de seguir una sèrie de passos per tal de poder començar el procés de modificació del document:

1. Seleccionar el document XML d'entrada amb les referències bibliogràfiques.
2. Seleccionar el document de text en format OpenDocument.
3. Elegir si es vol sobreescriure el document d'entrada:
    a. El sobreescriurem si fem click sobre la check que ho indica i llavors es deshabilitarà el camp per inserir el nom del document modificat i el de seleccionar la carpeta del document seleccionat.
    b. La check de sobreescriure el document d'entrada estarà desactiva i llavors haurem de seleccionar la carpeta i el nom del nou document.
4. Una vegada estiguin tots els paràmetres d'entrada podrem polsar el botó d'iniciar el procés.

També important destacar els elements de d'interfície dissenyats principalment per fer augmentar la usabilitat i la facilitat d'ús:

- Descripció de la informació que es mostra per pantalla.
- Missatges emergents sobre de cada caixa de text per indicar a l'usuari el tipus de fitxer què ha de buscar.
- Botons de cerca de documents, que permetran de forma fàcil trobar els fitxers d'entrada necessaris.
- Barra de progrés que anirà mostrant el temps que queda per finalitzar el procés de modificació.

- Botó d'informació on s'obrirà una petita finestra indicant les dades identificatives d'aquest programari (autor, consultor, descripció, versió).

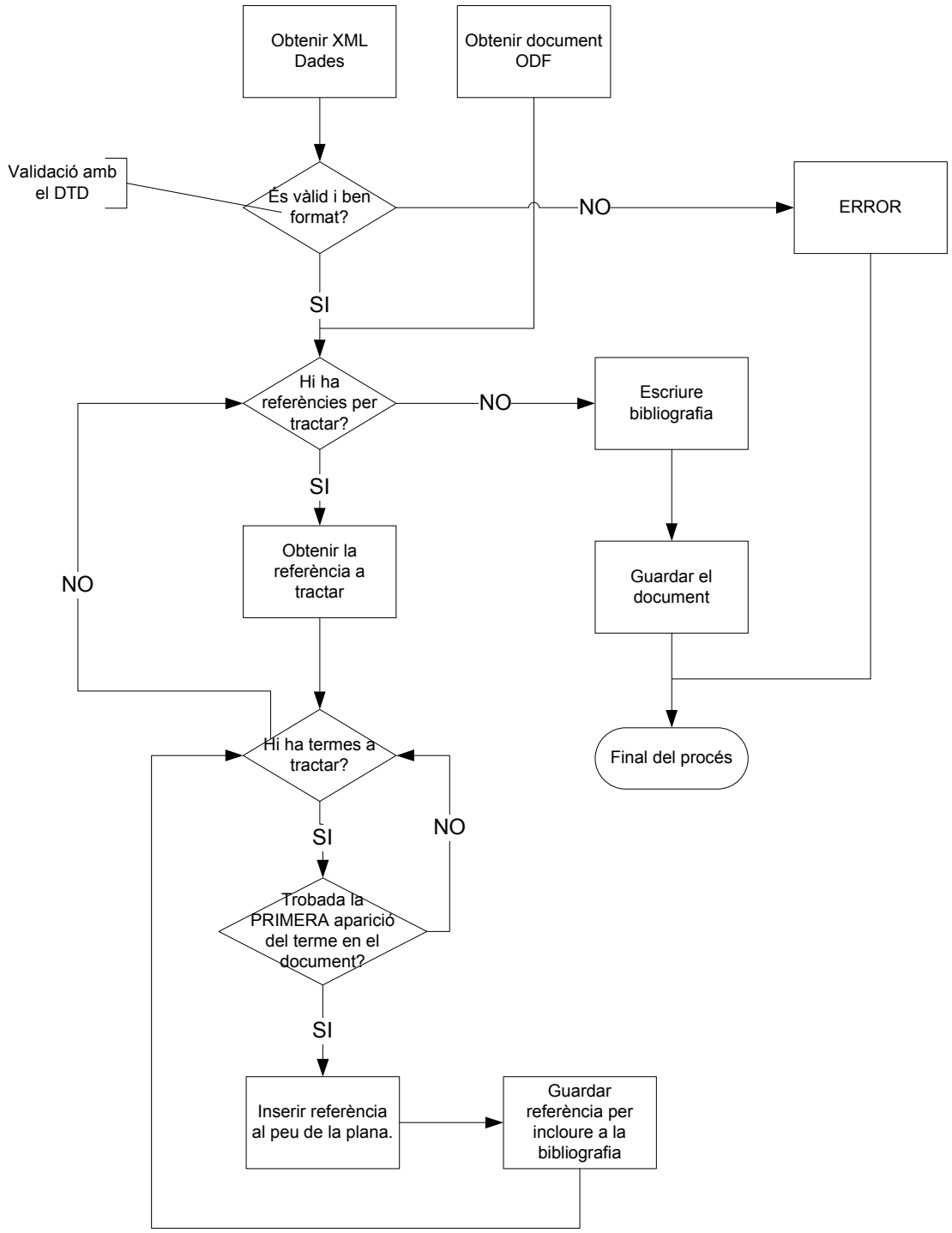
A més si l'aplicació funciona correctament es mostrarà un missatge indicant que el procés ha acabat amb èxit, en canvi, si el procés no ha acabat de forma correcte, es mostrarà un missatge amb la descripció de l'error. És important indicar que només es modificarà el document d'entrada o es crearà el document resultant en el cas que el procés hagi finalitzat correctament.

Per últim indicar que a la interfície no es demana el fitxer DTD per validar el XML d'entrada. Això és així perquè el DTD es distribuirà amb l'aplicació, és a dir, estarà en el mateix directori que el fitxer executable. En el disseny de l'algorisme de tractament s'explica de forma més detallada aquest últim punt.

4.2.3 Disseny de l'algorisme de tractament

Aquesta és la part principal del disseny del programari, el disseny de l'algorisme de tractament. Aquest algorisme haurà de recollir els fitxers d'entrada i comprovar la seva validesa. Després haurem de copiar el fitxer XML des de la ubicació indicada per l'usuari fins al directori de l'aplicació (serà un fitxer temporal que s'esborrarà al final del procés). I, per últim, realitzar la transformació del document de text OpenDocument per incloure-hi les referències bibliogràfiques.

A continuació mostrarem el diagrama de flux que segueix l'algorisme de tractament:



Com es pot veure, no és un algorisme senzill i, per aquest motiu, intentaré fer una explicació detallada per tal què quedi el més clar possible.

Primerament, realitzarem una copia del fitxer XML d'entrada en la carpeta de l'aplicació. A més de copiar-ho li canviarem el nom per un GUID per tal que sigui únic. Serà amb aquesta còpia amb la què treballarem durant tot el procés.

Una vegada tenim el fitxer temporal, el validarem amb la DTD i si és correcte carregarem l'altre fitxer d'entrada, el document de text a tractar.

En aquest punt es recorren totes les referències del fitxer XML d'entrada i, per cada una d'elles, tots els termes què la composen. Cadascú d'aquests termes es busquen dintre del document ODF i, quan trobem la primera aparició, afegim una nota a peu de plana amb la referència. Aquesta referència la guardem per tal què al final del procés es pugui mostrar una bibliografia amb totes elles.

Quan hem acabat de tractar totes les referències, guardem els canvis del document, eliminem el fitxer temporal d'entrada i finalitzem el procés.

Per últim, és important destacar què només es guardarà el contingut del document ODF modificat si tot el procés ha finalitzat correctament, en qualsevol altre cas, el document quedarà intacte o no es generarà un nou document de sortida (en funció de la opció triada per l'usuari).

4.2.4 Disseny i arquitectura de la llibreria AODL

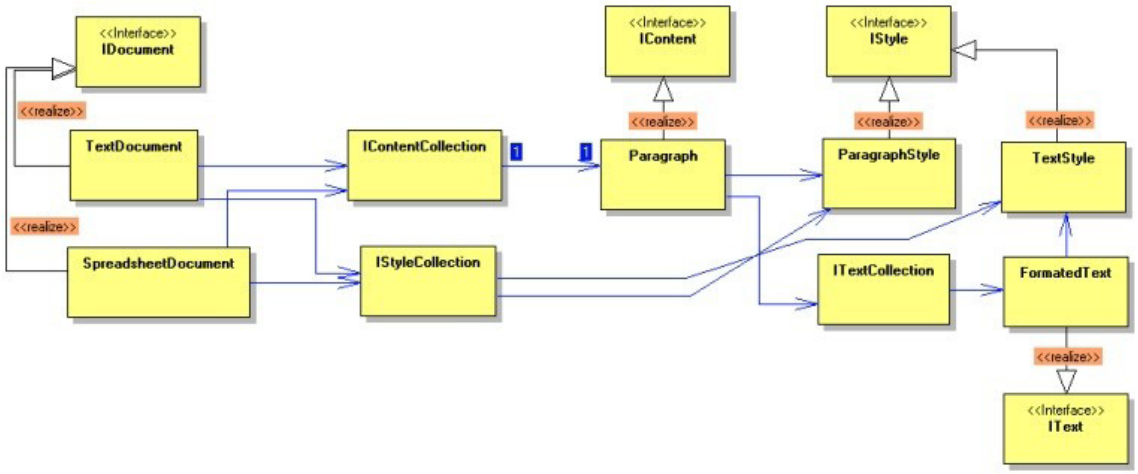
Dintre dels desenvolupaments relacionats amb ODF existeix un projecte open source anomenat *ODF Toolkit Project* que s'ha encarregat de desenvolupar una llibreria en C# per manipular documents OpenDocuement.

Les característiques més importants d'aquesta llibreria són les següents:

- Està implementat al 100% en C#.
- És compatible tant amb el Framework 1.1 com amb el Framework 2.0 (el que utilitzarem nosaltres).
- Suporta el tractament de documents de presentació o de text (el tipus de document que tractarem).
- Redueix el codi necessari per manipular documents.
- La majoria dels tags o elements XML que identifiquen objectes reals als documents ODF (paràgrafs, taules, vinyetes…) estan representats en aquesta llibreria amb objectes.
- Permet obrir, modificar i guardar documents.
- Ofereix la possibilitat d'exportar a d'altres formats (actualment s'està treballant perquè un document ODF pugui ser exportat a PDF).

De totes les característiques descrites una de les més interessants des del punt de vista del disseny de l'aplicació és que no tenim que treballar directament amb el XML del contingut del document, sinó que disposem d'una estructura de classes que "emula" l'estructura interna del document (p.ex. paràgrafs, taules, vinyetes...). D'aquesta manera, serà més eficient i més fàcil pel programador realitzar les tasques de manipulació d'un document.

A continuació es mostra l'esquema general de classes de la llibreria AODL:



Com es pot veure, els documents de text estan formats per una col·lecció d'elements (l'objecte que implementi la interfaz *IcontentCollection*). Cada element de la col·lecció serà un element físic del document, és a dir, un paràgraf, una taula, una llista… Tots aquests objectes implementaran la interfaz *Icontent* i a la vegada tindran un sèrie de propietats com pot ser el seu estil o el seu format.

Ara només ens queda determinar quins són els objectes típics d'un document ODF a on podrem trobar text, per tal de determinar en quins objectes de la llibreria AODL haurem de realitzar la cerca de termes:

- **En un paràgraf**. El terme formarà part d'un paràgraf de text.

- **En una taula**. Més concretament, a la cel·la d'una fila de la taula. Dintre d'aquesta cel·la hi haurà un paràgraf que contindrà el terme a tractar.

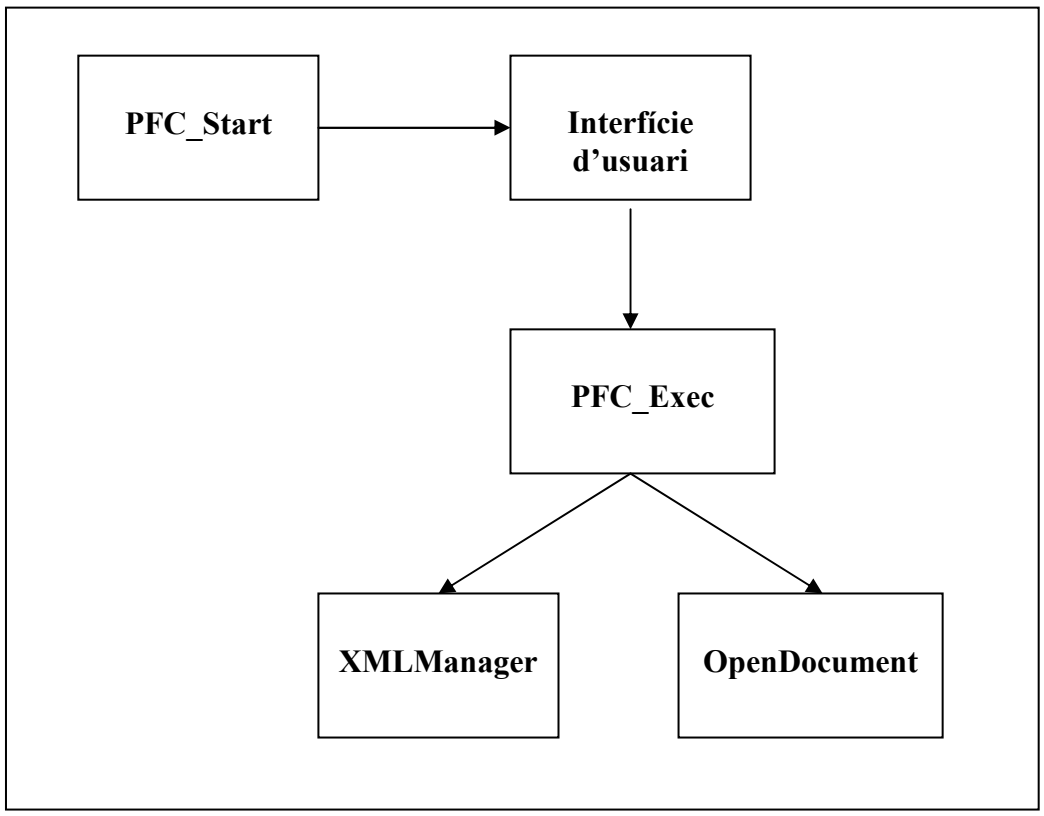
- **En una llista o vinyeta**. El terme estarà dintre d'una llista o vinyeta, a la vegada la llista o vinyeta tindrà un paràgraf que contindrà el terme a tractar.

Tractant aquestes estructures del document es contempla la gran majoria de casos d'un document estàndard a on podrem trobar text i, per tant, a on cercarem els termes.

4.2.5 Disseny de classes

En aquest apartat mostrarem el disseny de les classes principals que s'encarregaran de realitzar la lògica de l'aplicació.

A continuació es mostra un esquema d'interacció entre aquestes classes:



- **XMLManager.** Aquesta classe s'encarregarà de comprovar que el XML d'entrada estigui ben format i que, a més, sigui vàlid, és a dir, que compleixi amb les especificacions de la seva DTD.

- **OpenDocument.** Aquesta classe s'encarregarà de realitzar tot el tractament del document ODF. De fet, aquesta classe servirà de pont entre la llibreria AODL i el document ODF. D'aquesta manera, la classe exposarà una sèrie de mètodes totalment transparents a aquesta llibreria i al format intern ODF.

- **PFC_Start.** Aquesta classe serà el punt d'entrada en l'aplicació, la què obrirà la interfície del programari.

- **PFC_Exec.** Serà la classe encarregada de cridar als mètodes necessaris de *XMLManager* i *OpenDocument* per realitzar tot el procés.

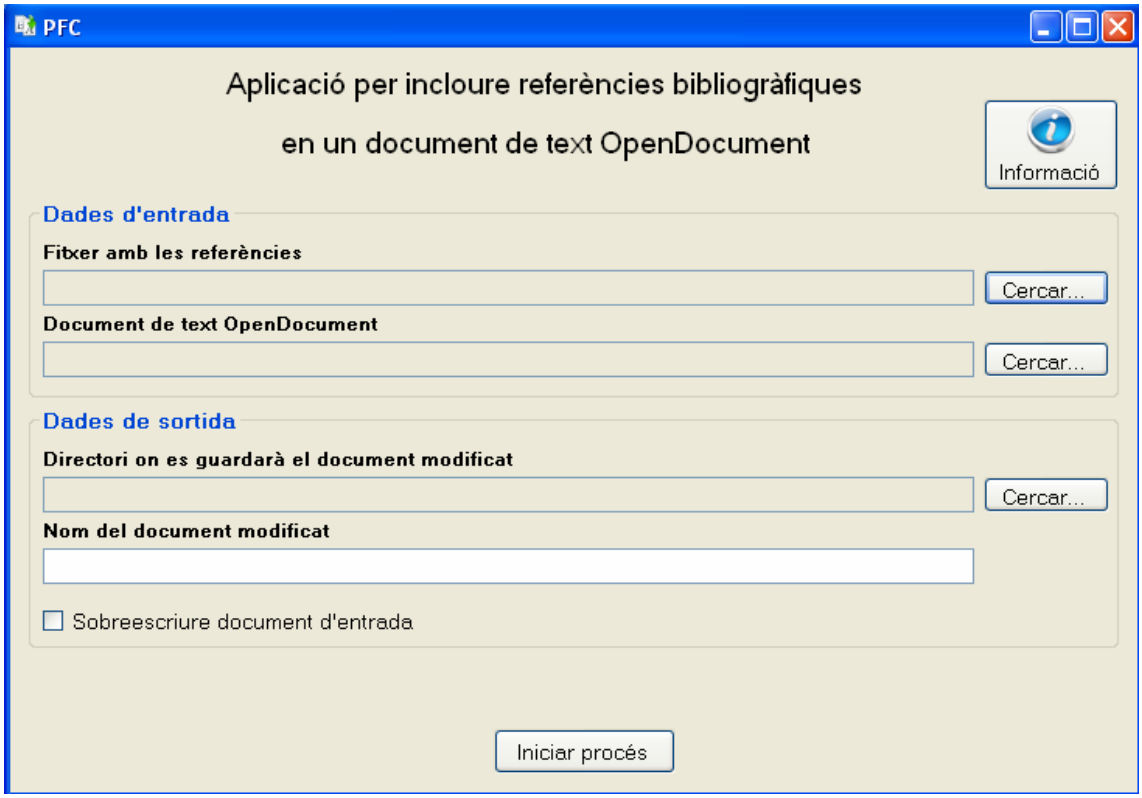
4.3 Implementació

Com ja hem comentat en punts anteriors, l'aplicació s'ha implementat sobre C# i, en concret, s'ha desenvolupat en el programari Microsoft *Visual C# 2005 Express*, la versió gratuïta per desenvolupar en .NET sobre aquest llenguatge.

En aquest punt es pretén explicar la metodologia què s'ha seguit per programar el sistema, així com mostrar petits exemples de codi.

4.3.1 Capa de presentació

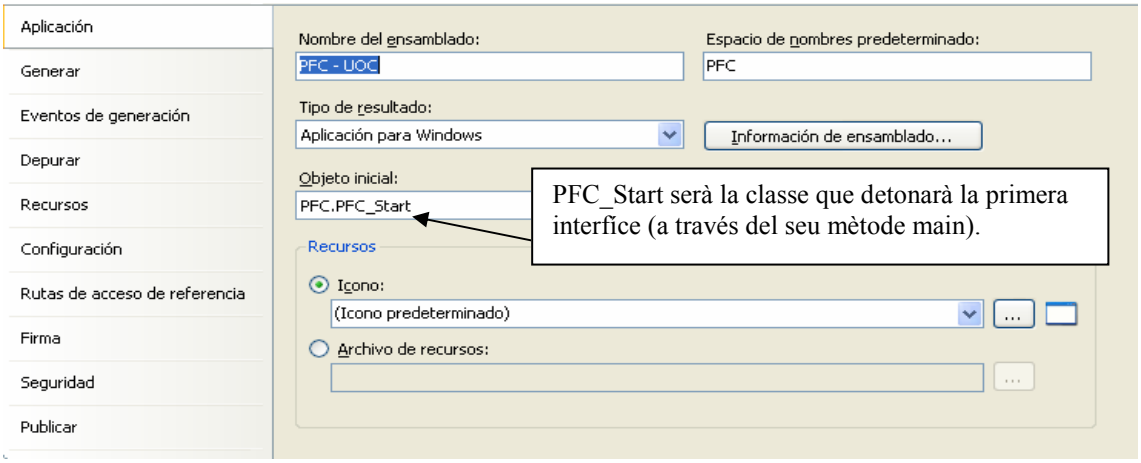
Tindrem un únic formulari que serà l'encarregat de recollir el fitxers d'entrada i de fer el tractament del document ODF.



En els projectes C# no podem indicar directament el nom del formulari inicial de l'aplicació, sinó què hem de definir una classe que serà la què s'encarregarà d'obrir la primera interfície que l'usuari es trobarà (en el nostre cas, la primera i única).

A través de la opció del Visual C# 2005 Express *Proyecto > Propiedades del proyecto* podrem diferents característiques del projecte:

- **Nom de l'ensamblat**. Nom que tindrà el fitxer executable.
- **Objecte inicial**. Classe encarregada d'iniciar l'aplicació.
- **Icona**. Icona que es mostrarà tant a la finestra principal de l'aplicació com en el fitxer executable.



4.3.1.1 Gestió d'esdeveniments

Llavors, una vegada visualitzem la interfície, ens queda controlar totes les possibles accions que pot realitzar l'usuari i no deixar realitzar el tractament del document ODF fins que no hagi informat de les dades d'entrada necessàries.

Aquestes accions que l'usuari podrà realitzar les haurem de controlar a través de la gestió d'esdeveniments. En particular, en el nostre cas haurem de gestionar els esdeveniments relacionats amb els botons de la interfície. Per una banda, tindrem els botons per seleccionar els fitxers d'entrada i, per un altre, el botó que inicialitza el tractament del document ODF.

També serà important la resposta que se li dóna a l'usuari a les seves accions. Nosaltres només tindrem que donar resposta al botó que detona el tractament del document OpenDocument. A continuació es mostren els diferents tipus de missatge que podrà donar l'aplicació:

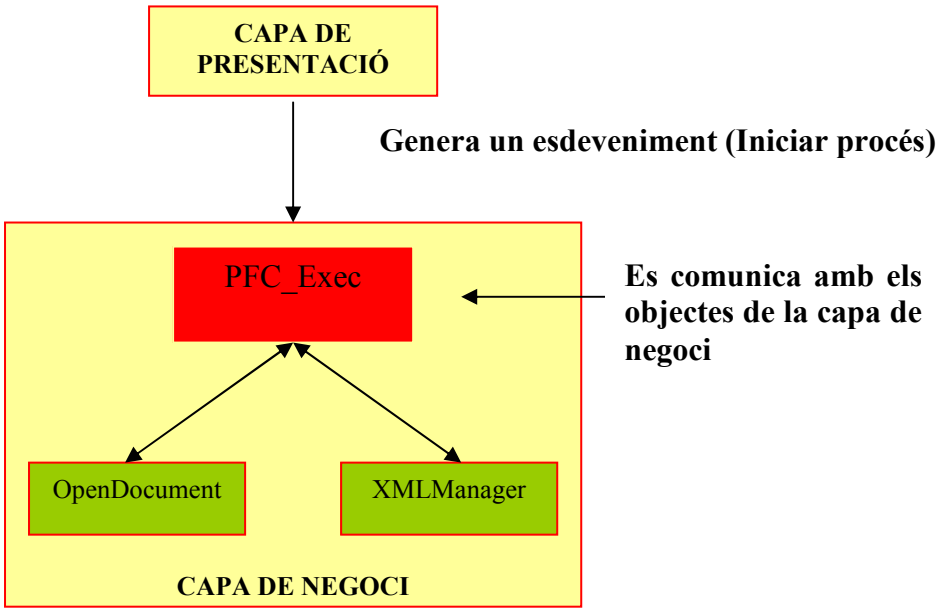
- **Missatges de falta de dades.**
  - Quan l'usuari no hagi indicat la ubicació del document XML de dades.
  - Quan l'usuari no hagi indicat la ubicació del document ODF.

- **Missatges de dades errònies o incorrectes.**
  - Quan el document XML no estigui ben format o no compleixi els requisits del seu DTD.
  - Quan el document ODF no sigui vàlid.

- **Missatges de finalització**
  - Finalització del procés de forma correcta.
  - Finalització del procés perquè s'ha llançat una excepció.

4.3.2 Capa de negoci

Per la realització d'aquesta capa s'ha realitzat la implementació de les classes indicades en el procés de disseny.



Aquest esquema exemplifica l'esquema d'interacció entre la capa de presentació i la capa de negoci. Aquesta interacció seguirà els següents passos:

1. L'usuari polsa el botó d'iniciar el procés i genera un esdeveniment que es capturat per la capa de presentació.

2. La capa de presentació crida a la classe que es comunicarà amb els objectes de negoci per realitzar tot el procés.

3. El gestor de la capa de negoci serà l'encarregat d'interactuar amb els objectes per tal de dur a terme el procés.

A continuació mostrem un exemple d'aquesta interacció:

```
private void btnExec_Click(object sender, EventArgs e)
{
      PFC_Exec exec;

            try
            {

                  //codi omitit per claredat
                  // (. . .)

                  exec.init();

                  MessageBox.Show("Procés finalitzat
                  correctament.");
            }

            catch (Exception ex)
            {

                  MessageBox.Show(ex.Message.ToString(),"Error",
                  MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

Creem una instancia del gestor de la capa de negoci

Li diem al gestor que inici el procés

```
public void init()
        {
            try
            {
                OpenDocument od;
                XmlManager oXMLManager;

                //validem el XML d'entrada amb el seu DTD
                oXMLManager = new XmlManager(m_sPathCopyXML);
                oXMLManager.ValidateWithDTD();

                //codi que no mostrem per simplicitat
                // ( . . .)

                //carreguem el document ODF a memòria
                od = new OpenDocument(m_sPathODT);

                //codi que no mostrem per simplicitat
                // ( . . .)

                //guardamos el documento con los cambios realizados
                od.Save(m_sPathODTFinal);
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
```

Creem instàncies dels objectes de la capa de negoci

Interactuem amb els objectes de negoci per realitza les funcions necessàries

4.3.2.1 Interacció amb la llibreria AODL

Aquest és el punt més interessant i costos de la implementació. Per tal d'interactuar amb la llibreria AODL s'ha creat una classe (*OpenDocument*)que només exposarà els mètodes necessaris pel tractament, d'aquesta manera és totalment transparent l'arquitectura dels documents ODF pels objectes que interactuïn amb aquesta classe.

A continuació mostrarem diferents exemples d'utilització d'aquesta llibreria:

Obrir un document

```
m_oDoc = new AODL.Document.TextDocuments.TextDocument();

//sPath és la ruta física del document
m_oDoc.Load(sPath);
```

Guardar els canvis en un document

```
m_oDoc = new AODL.Document.TextDocuments.TextDocument();

//sPath és la ruta física a on es guardarà el documentdel
document. Si en aquesta ruta ja hi havia un document ODF el
sobreescriurà.
m_oDoc.Save(sPath);
```

Inserir una marca a peu de plana

```
//es crea una marca a peu de plana
note = new AODL.Document.Content.Text.Footnote(m_oDoc,
sReference, "fnote" + m_iIdFootnote.ToString(),
AODL.Document.Content.Text.FootnoteType.footnode);

//s'inserta la marca a la paraula del text
oParagraph.TextContent.Insert(indexText + 1, note);

//s'incrementa el contador que servirà per identificar
internament una nota a peu de plana
m_iIdFootnote++;
```

Cerca de termes dintre del document

```
//Obtenim la col·lecció d'elements dels document
enumDoc = m_oDoc.Content.GetEnumerator();

//Fem un recorregut per tots els elements del document
while (enumDoc.MoveNext())
{
    //Identifiquem un paràgraf
    if (enumDoc.Current is AODL.Document.Content.Text.Paragraph)
    {
      oParagraph = enumDoc.Current as
      AODL.Document.Content.Text.Paragraph;

      if (SearchInParagraph(ref oParagraph, sTerm, sReference))
      {
        //( . . .)
      }
    }
    //Identifiquem una taula
    else if (enumDoc.Current is AODL.Document.Content.Tables.Table)
    {
      oTable = enumDoc.Current as
      AODL.Document.Content.Tables.Table;

      if (SearchInTable(ref oTable, sTerm, sReference))
      {
         //( . . .)
      }
    }
    //Identifiquem una llista o vinyeta
    else if (enumDoc.Current is AODL.Document.Content.Text.List)
    {
      oList = enumDoc.Current as AODL.Document.Content.Text.List;
      if (SearchInList(ref oList, sTerm, sReference))
      {
         //( . . .)
      }
    }
}
```

Cerquem el terme dintre dels paràgrafs del document

Crear un salt de pàgina (per separar el document de la bibliografia)

```
//creem un paràgraf que contindrà el salt de pàgina
oParagraph =
AODL.Document.Content.Text.ParagraphBuilder.CreateParagraphWithCust
omStyle(m_oDoc, ID_STYLE_BREAK_PAGE);

//creamos un element dintre del paràgraf que indicarà que és un
paràgraf especial (un salt de pàgina)
oElementXML = m_oDoc.CreateNode("paragraph-properties", "style");
oAttXML = m_oDoc.CreateAttribute("break-before", "fo");
oAttXML.Value = "page";
oElementXML.Attributes.Append(oAttXML);

//aquest element creat formarà part de l'estil del paràgraf
oPropertyStyle =  new
AODL.Document.Styles.Properties.ParagraphProperties(oParagraph.Styl
e, oElementXML);
```

Crear un títol amb negreta (el títol per la bibliografia)

```
//creem un paràgraf
oParagraph =
AODL.Document.Content.Text.ParagraphBuilder.CreateStandardTextParag
raph(m_oDoc);

//creem l'estil del text del paràgraf
oFormatedText = new AODL.Document.Content.Text.FormatedText(m_oDoc,
"R" + iCont, "Bibliografia");

oFormatedText.TextStyle.TextProperties.Bold = "bold";
oFormatedText.TextStyle.TextProperties.FontSize = "20";
oParagraph.TextContent.Add(oFormatedText);

//afegim el paràgraf al document
m_oDoc.Content.Add(oParagraph);
```

4.4 Proves i exemples de funcionament

Una vegada dissenyades i implementades totes les parts que composen aquest projecte, es mostraran una sèrie d'exemples per tal de verificar el seu correcte funcionament.

Per realitzar aquesta etapa de test s'han realitzar dos casos de prova, el primer on es proven els casos senzills d'utilització i el segon on es proven els casos més complexen i que tenen un tractament especial.

4.4.1 Cas de prova 1: Exemple senzill

Aquest exemple estarà format per un XML d'entrada amb 4 referències bibliogràfiques i on cadascuna d'elles tindrà només un terme a cercar dintre del document. Hi hauran diferents tipus de referències, per tal d'exemplificar les diferents casuístiques i combinacions de dades dintre d'una referència.

XML d'entrada

Referència 1

```
<reference>
    <terms>
        <term>conocimiento</term>
    </terms>
    <info>
        <authors>
            <author>
                <firstName>J.</firstName>
                <lastNames>Vilaseca</lastNames>
            </author>
        </authors>
        <title>La economía del conocimiento: paradigma tecnológico
        y cambio estructural. Un análisis empírico e internacional
        para la economía española
        </title>
        <edition>Working Paper Series WP02-003</edition>
        <datePublish>2002</datePublish>
        <online>
            <url>http://www.uoc.edu/in3/dt/20001/index.html
            </url>
            <searchDate>14/11/2007</searchDate>
        </online>
    </info>
</reference>
```

Referència 2

```
<reference>
  <terms>
    <term>Krieg-Brückner</term>
  </terms>
  <info>
    <authors>
      <author>
        <firstName>M.</firstName>
        <lastNames>Broy</lastNames>
      </author>
    </authors>
    <title>Derivation of invariant assertions during program
    development by transformation
    </title>
    <edition>McGraw-Hill</edition>
    <datePublish>1993</datePublish>
    <placePublish />
    <pages>725</pages>
    <ISBN>84-86251-70-2</ISBN>
  </info>
</reference>
```

Referència 3

```
<reference>
  <terms>
  <term>Abstracción</term>
  </terms>
  <info>
    <authors>
      <author>
        <firstName>R.</firstName>
        <lastNames>Peña</lastNames>
      </author>
    </authors>
    <title>Diseño de Programas: Formalismo y Abstracción</title>
    <edition>Prentice-Hall</edition>
    <datePublish>1993</datePublish>
    <pages />
    <location>134-150</location>
    <ISBN>0-13-0984450-7</ISBN>
  </info>
</reference>
```

Referència 4

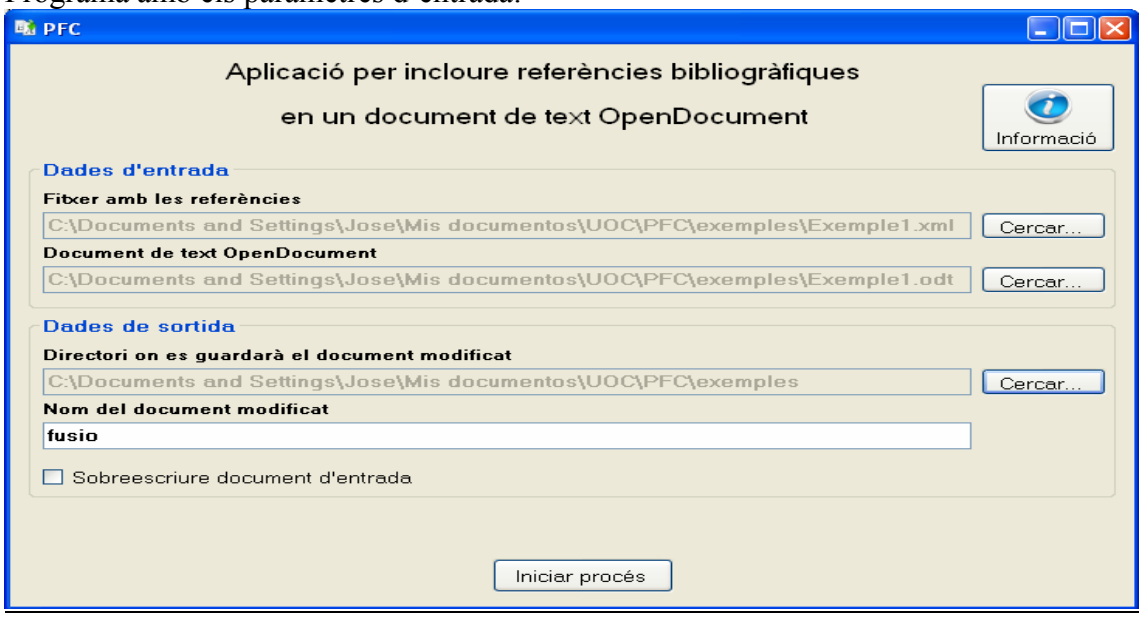
```
<reference>
   <terms>
     <term>Manber</term>
   </terms>
   <info>
     <authors>
        <author>
           <firstName>U.</firstName>
           <lastNames>Manber</lastNames>
        </author>
     </authors>
     <title>Introduction to Algorithms</title>
     <edition>Addison-Wesley</edition>
     <datePublish>1989</datePublish>
     <placePublish>London</placePublish>
     <pages>456</pages>
     <location>100-120</location>
     <ISBN>0-13-5584220-4</ISBN>
   </info>
</reference>
```

Resultat esperat

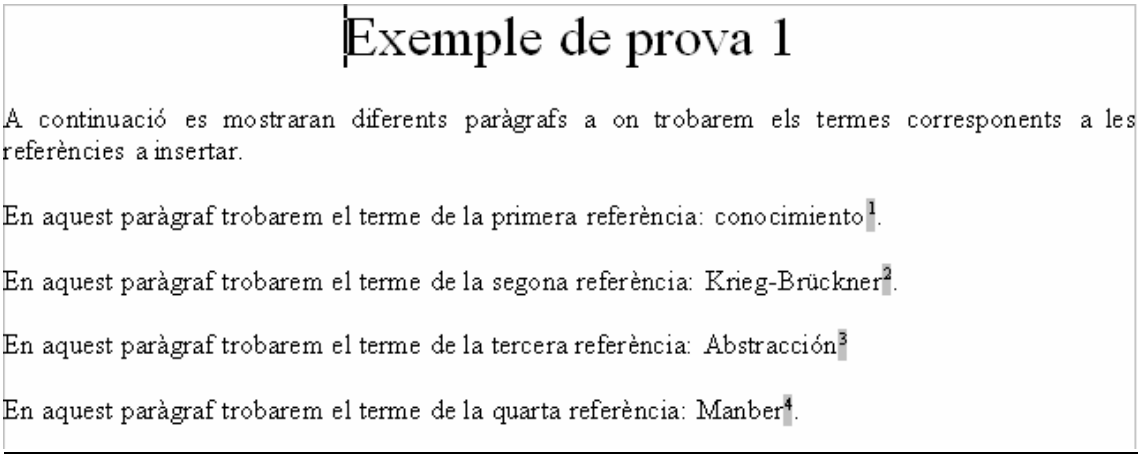
El document en el qual s'han d'inserir les referències serà molt simple i només tindrà una pàgina. Aquesta pàgina contindrà en diferents paràgrafs de text els termes de cadascuna de les referències. Per tant, el resultat esperat serà mostrar les notes a peu de plana de les diferents referències (en aquest cas 4) i al final del document mostrar una bibliografia amb totes elles.

Resultat obtingut

Programa amb els paràmetres d'entrada:



Document amb els termes que fan referència a notes a peu de pàgina.



Exemple de prova 1

A continuació es mostraran diferents paràgrafs a on trobarem els termes corresponents a les referències a insertar.

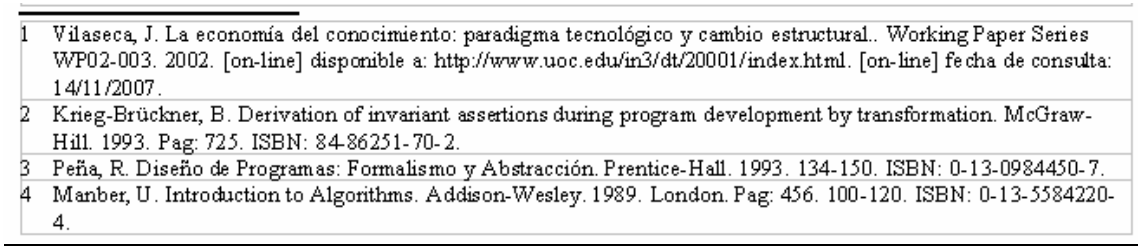
En aquest paràgraf trobarem el terme de la primera referència: conocimiento[1].

En aquest paràgraf trobarem el terme de la segona referència: Krieg-Brückner[2].

En aquest paràgraf trobarem el terme de la tercera referència: Abstracción[3]

En aquest paràgraf trobarem el terme de la quarta referència: Manber[4].

Notes a peu de pàgina corresponent a les referències bibliogràfiques.



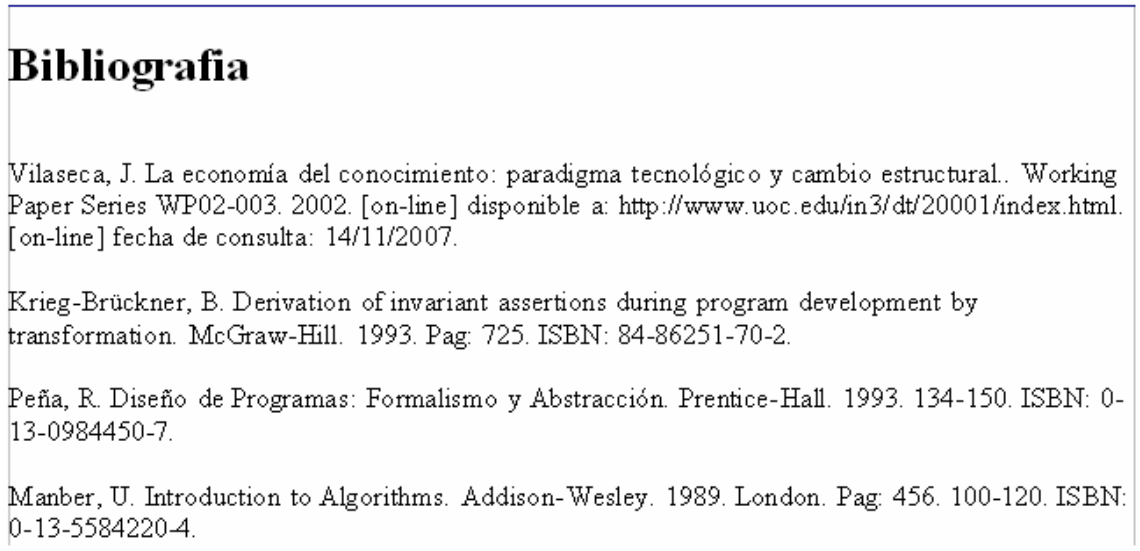
1 Vilaseca, J. La economía del conocimiento: paradigma tecnológico y cambio estructural.. Working Paper Series WP02-003. 2002. [on-line] disponible a: http://www.uoc.edu/in3/dt/20001/index.html. [on-line] fecha de consulta: 14/11/2007.

2 Krieg-Brückner, B. Derivation of invariant assertions during program development by transformation. McGraw-Hill. 1993. Pag: 725. ISBN: 84-86251-70-2.

3 Peña, R. Diseño de Programas: Formalismo y Abstracción. Prentice-Hall. 1993. 134-150. ISBN: 0-13-0984450-7.

4 Manber, U. Introduction to Algorithms. Addison-Wesley. 1989. London. Pag: 456. 100-120. ISBN: 0-13-5584220-4.

Pàgina amb totes les referències bibliogràfiques utilitzades en el document.



Bibliografia

Vilaseca, J. La economía del conocimiento: paradigma tecnológico y cambio estructural.. Working Paper Series WP02-003. 2002. [on-line] disponible a: http://www.uoc.edu/in3/dt/20001/index.html. [on-line] fecha de consulta: 14/11/2007.

Krieg-Brückner, B. Derivation of invariant assertions during program development by transformation. McGraw-Hill. 1993. Pag: 725. ISBN: 84-86251-70-2.

Peña, R. Diseño de Programas: Formalismo y Abstracción. Prentice-Hall. 1993. 134-150. ISBN: 0-13-0984450-7.

Manber, U. Introduction to Algorithms. Addison-Wesley. 1989. London. Pag: 456. 100-120. ISBN: 0-13-5584220-4.

4.4.2 Cas de prova 2: Exemple complex

En el cas anterior tenim el cas senzill que és buscar els termes en els paràgrafs del document, però també es poden trobar dintre de taules o numeracions. Per altre banda, també podem tenir el cas en què una referència s'hagi d'inserir més d'una vegada al peu de pàgina del document (quan una referència té més d'un terme) però només una en la bibliografia. També haurem de comprovar que només s'haurà d'inserir una referència a peu de plana la primera vegada que es trobi el terme i les posteriors ja no s'haurà d'indicar.

Tots aquests casos són els què es provaran en aquest exemple i d'aquesta manera s'hauran realitzat les proves necessàries que donin garanties del bon funcionament del programari.

Per realitzar l'exemple utilitzarem les mateixes referències què abans amb la diferència que per cadascuna d'elles afegirem més d'un terme i també afegirem més d'un autor a cada referència per veure què la genera correctament en el document de sortida.

XML d'entrada

Referència 1

```
<reference>
    <terms>
        <term>conocimiento</term>
        <term>economía</term>
    </terms>
    <info>
        <authors>
            <author>
                <firstName>J.</firstName>
                <lastNames>Vilaseca</lastNames>
            </author>
            <author>
                <firstName>J.</firstName>
                <lastNames>Torrent</lastNames>
            </author>
            <author>
                <firstName>Á.</firstName>
                <lastNames>Díaz</lastNames>
            </author>
        </authors>
        <title>La economía del conocimiento: paradigma tecnológico
        y cambio estructural. Un análisis empírico e internacional
        para la economía española
        </title>
        <edition>Working Paper Series WP02-003</edition>
        <datePublish>2002</datePublish>
        <online>
            <url>http://www.uoc.edu/in3/dt/20001/index.html
            </url>
            <searchDate>14/11/2007</searchDate>
        </online>
    </info>
</reference>
```

```
<reference>
  <terms>
    <term>Krieg-Brückner</term>
    <term>transformation</term>
    <term>Broy</term>
  </terms>
  <info>
    <authors>
      <author>
        <firstName>M.</firstName>
        <lastNames>Broy</lastNames>
      </author>
      <author>
        <firstName>B.</firstName>
        <lastNames> Krieg-Brückner </lastNames>
      </author>
    </authors>
    <title>Derivation of invariant assertions during program
    development by transformation
    </title>
    <edition>McGraw-Hill</edition>
    <datePublish>1993</datePublish>
    <placePublish />
    <pages>725</pages>
    <ISBN>84-86251-70-2</ISBN>
  </info>
</reference>
```

Referència 3

```
<reference>
  <terms>
  <term>Abstracción</term>
  </terms>
  <info>
    <authors>
      <author>
        <firstName>R.</firstName>
        <lastNames>Peña</lastNames>
      </author>
    </authors>
    <title>Diseño de Programas: Formalismo y Abstracción</title>
    <edition>Prentice-Hall</edition>
    <datePublish>1993</datePublish>
    <pages />
    <location>134-150</location>
    <ISBN>0-13-0984450-7</ISBN>
  </info>
</reference>
```

Referència 4

```
<reference>
   <terms>
      <term>Manber</term>
      <term>Algorithms</term>
   </terms>
   <info>
      <authors>
         <author>
            <firstName>U.</firstName>
            <lastNames>Manber</lastNames>
         </author>
      </authors>
      <title>Introduction to Algorithms</title>
      <edition>Addison-Wesley</edition>
      <datePublish>1989</datePublish>
      <placePublish>London</placePublish>
      <pages>456</pages>
      <location>100-120</location>
      <ISBN>0-13-5584220-4</ISBN>
   </info>
</reference>
```

Resultat esperat

De les 4 referències en el document d'entrada només es trobaran els termes de la 1,2 i 4, per tant, la referència 3 no sortirà al peu de plana del document i tampoc a la bibliografia.

De la referència 1 trobarem els 2 termes una única vegada, per tant, s'inserirà la referència dues vegades al peu de la plana (una per cadascuna d'elles) però només una a la bibliografia.

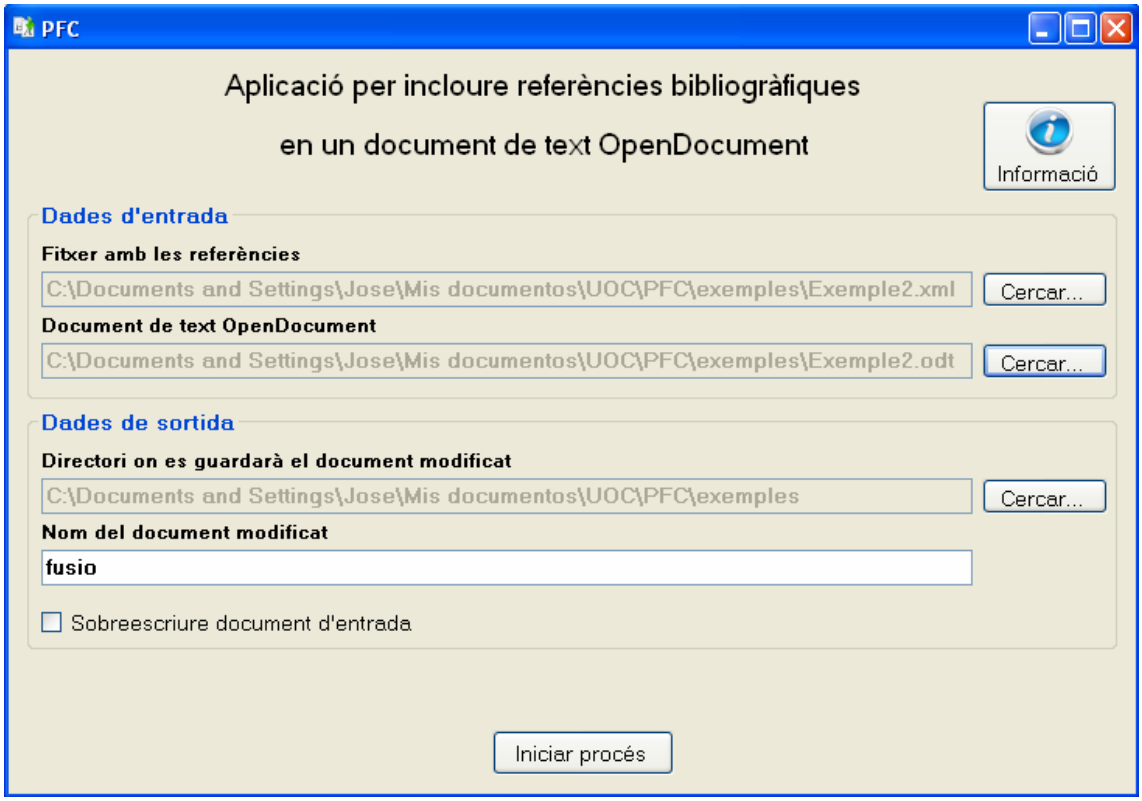
De la referència 2 trobarem el primer terme 1 vegada, el segon 2 vegades i el tercer cap vegada. Per tant, pel primer terme s'inserirà una vegada la referència, pel segon, s'inserirà una vegada i en la primera aparició del terme i pel tercer terme com no es troba no s'inserirà cap vegada.

Com ja s'ha comentat abans, per la referència 3 no s'ha trobat cap dels seus termes i, per tant, no s'inserirà a cap peu de pàgina ni a la bibliografia.

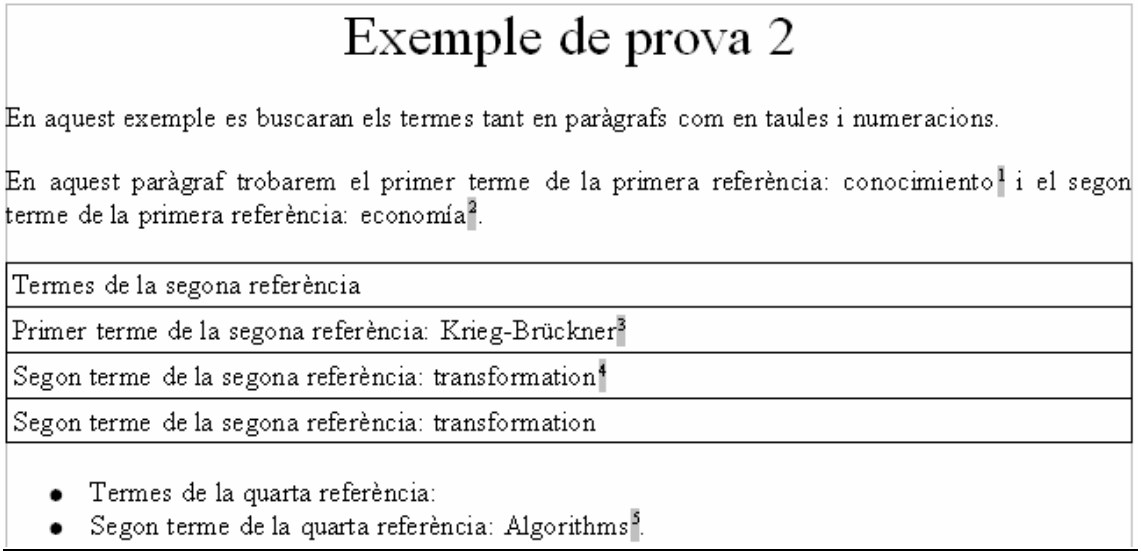
De la referència 4 només trobarem el segon terme una vegada i, per tant, només s'inserirà una vegada al peu de pàgina i una vegada a la bibliografia.

Resultat obtingut

Programa amb els paràmetres d'entrada:



Document amb els termes que fan referència a notes a peu de pàgina.



Exemple de prova 2

En aquest exemple es buscaran els termes tant en paràgrafs com en taules i numeracions.

En aquest paràgraf trobarem el primer terme de la primera referència: conocimiento[1] i el segon terme de la primera referència: economía[2].

Termes de la segona referència
Primer terme de la segona referència: Krieg-Brückner[3]
Segon terme de la segona referència: transformation[4]
Segon terme de la segona referència: transformation

- Termes de la quarta referència:
- Segon terme de la quarta referència: Algorithms[5].

Notes a peu de pàgina corresponent a les referències bibliogràfiques.

1	Vilaseca, J., Torrent, J., Díaz, Á. La economía del conocimiento: paradigma tecnológico y cambio estructural. Un análisis empírico e internacional para la economía española. Working Paper Series WP02-003. 2002. [on-line] disponible a: http//www.uoc.edu/in3/dt/20001/index.html. [on-line] fecha de consulta: 14/11/2007.
2	Vilaseca, J., Torrent, J., Díaz, Á. La economía del conocimiento: paradigma tecnológico y cambio estructural. Un análisis empírico e internacional para la economía española. Working Paper Series WP02-003. 2002. [on-line] disponible a: http//www.uoc.edu/in3/dt/20001/index.html. [on-line] fecha de consulta: 14/11/2007.
3	Broy, M., Krieg-Brückner, B. Derivation of invariant assertions during program development by transformation. McGraw-Hill. 1993. Pag: 725. ISBN: 84-86251-70-2.
4	Broy, M., Krieg-Brückner, B. Derivation of invariant assertions during program development by transformation. McGraw-Hill. 1993. Pag: 725. ISBN: 84-86251-70-2.
5	Manber, U. Introduction to Algorithms. Addison-Wesley. 1989. London. Pag: 456. 100-120. ISBN: 0-13-5584220-4.

Pàgina amb totes les referències bibliogràfiques utilitzades en el document.

Bibliografia

Vilaseca, J., Torrent, J., Díaz, Á. La economía del conocimiento: paradigma tecnológico y cambio estructural. Un análisis empírico e internacional para la economía española. Working Paper Series WP02-003. 2002. [on-line] disponible a: http://www.uoc.edu/in3/dt/20001/index.html. [on-line] fecha de consulta: 14/11/2007.

Broy, M., Krieg-Brückner, B. Derivation of invariant assertions during program development by transformation. McGraw-Hill. 1993. Pag: 725. ISBN: 84-86251-70-2.

Manber, U. Introduction to Algorithms. Addison-Wesley. 1989. London. Pag: 456. 100-120. ISBN: 0-13-5584220-4.

CAPÍTOL 5: INQUIETUDS SORGIDES DE L'ELABORACIÓ DEL PFC

5.1 Introducció

Durant la realització d'aquest projecte, m'han sorgit diferents inquietuds que m'agradaria personalment en un futur poder tractar-les i estudiar-les. Aquestes inquietuds estan relacionades tant amb la fase d'estudi de les diferents tecnologies com amb la fase de disseny i implementació de l'aplicació. Per tant, aquest capítol simplement intenta plasmar quins són els diferents temes que tractaré en un futur per complimentar els coneixements assolits durant aquests mesos.

5.2 OpenDocument

Relacionat amb el format ODF, diria que una vegada realitzat l'estudi de les seves característiques i la seva arquitectura i, sobretot, després de centrar-me en els documents de text d'aquest format, m'han sorgit les següents preguntes: Seran igual o més complexes els altres formats existents, com per exemple una fulla de càlcul o un document de presentació? Hi haurà tanta informació en xarxa dels altres formats de documents en comparació amb els de text?

Espero en els pròxims mesos poder donar resposta a aquestes preguntes i no només assolir els coneixements per modificar un document de text sinó també per poder fer-ho per una fulla de càlcul o un document de presentació.

5.3 Llibreries de suport al format OpenDocument.

Com s'ha explicat en capítols anteriors, s'ha utilitzar la llibreria de lliure distribució AODL com a suport al C# per tractar documents OpenDocument. Durant la recerca d'eines per poder interactuar amb aquest format, vaig veure que també existeix una per Java anomenada *ODF4J* (OpenDocumentFormat for Java). Llavors la pregunta que em va sorgir era saber si aquestes dues llibreries implementaven les mateixes funcionalitats, si hi havia alguna de les dues que era més eficient, ...

Per tant, és un dels temes que tinc pendents veure quina de les dos llibreries és més eficient i veure, si per una futura implementació d'una aplicació semblant, serà rellevant el llenguatge de programació utilitzat en funció de la millor llibreria disponible.

CAPÍTOL 6: LINIES FUTURES DE TREBALL

Realment, amb la consecució d'aquest projecte s'han assolit totes les fites marcades inicialment, per tant, les línies de treball en un futur estarien més dirigides a una ampliació de funcionalitats de l'aplicació elaborada.

Un tema interessant, en el què no s'ha aprofundit en aquest PFC és la forma d'implementar text en diferents formats, és a dir, com fer un text en negreta, amb lletres més grans, amb diferents colors, etc. L'assoliment d'aquests coneixements ens permetria realitzar les notes a peu de pàgina i la bibliografia amb un format determinat proporcionat per l'usuari.

Un exemple pràctic seria el següent, quan parlem d'una referència bibliogràfica web, aquesta tindrà la ruta de l'enllaç que li fa referència. D'on per aquest tipus de dada seria molt útil que es generés un vincle a la pàgina de la referència (ex. www.google.es). També seria útil definir el tipus de lletra i la grandària en les notes a peu de pàgina o en les dades de la bibliografia.

Per tant, una segona versió d'aquesta aplicació consistiria a definir quines series les propietats del text que l'usuari es podria definir i de cadascuna de les propietats quines serien les opcions disponibles.

CAPÍTOL 7: RECURSOS UTILITZATS

7.1 Programari

El programari utilitzat per desenvolupar aquest PFC ha estat el següent:

Microsoft Windows XP Professional.
Microsoft Word 2003.
Microsoft Project 2003.
OpenOffice 2.2
Microsoft Visual C# 2005 Express.

7.2 Maquinari

El maquinari utilitzat per desenvolupar aquest PFC ha estat el següent:

Portàtil DELL amb processador Intel Centrino a 1,86Mhz i 1Gb de memòria Ram.

CAPÍTOL 8: CONCLUSIONS

8.1 Conclusió final

Una vegada finalitzada l'elaboració d'aquest projecte he de dir que s'han assolit els objectius marcats inicialment.

S'ha elaborat un estudi exhaustiu sobre diferents tecnologies. XML com a format de dades estructurades, DTD com a format de validació de documents XML i OpenDocument com a estàndard obert de documents.

A partir d'aquests coneixements adquirits, s'ha elaborat l'estructura d'un document XML que pogués emmagatzemar diferents tipus de referències bibliogràfiques i, a més, s'ha realitzar una DTD per tal de validar el seu contingut.

Per una altra banda, s'ha dissenyat una aplicació que permetés llegir aquest XML de dades i validés el seu contingut a través de la DTD. Aquestes dades del fitxer d'entrada s'han utilitzat per afegir referències bibliogràfiques a un document de text OpenDocument. Addicionalment, s'ha obtingut un programari senzill d'utilitzar, amb una interfície gràfica amigable i fàcil de seguir.

8.2 Valoració personal

Fa uns mesos, abans de començar l'elaboració d'aquest projecte, tenia molt poca informació sobre el format OpenDocument, s'havia que era un format lliure i gratuït de documents semblant als què ens ofereix Office però poc més.

Ara, després d'elaborar les diferents parts que inclou aquest projecte, tinc una visió força diferent a la inicial. Per una banda he pogut aprendre com és l'arquitectura interna d'un document ODF i he pogut veure el gran potencial que tenen els documents XML ja què són la base del format OpenDocument.

Per una altra banda, he vist realment que aquest format no té res a envejar a la suite ofimàtica de Microsoft i, fins i tot, arribaria més lluny, diria què la seva arquitectura és millor i més senzilla.

Un altre dels punts que m'ha sorprès gratament és que Microsoft, en la seva versió 2007 d'Office ha intentat "emular" el format ODF que des de fa molts anys és estable i robust.

En definitiva, aquest projecte m'ha donat la possibilitat d'enriquir-me personalment gràcies als coneixements adquirits i espero en un futur poder aprofitar en la meva vida laboral la majoria dels conceptes apresos durant aquests mesos.

Per últim, espero i desitjo que totes les persones interessades tant en l'aplicació com amb la documentació elaborada els sigui d'utilitat i sàpiguen valorar les hores de treball dedicades a aquest projecte.

GLOSSARI

XML: Llenguatge de marques extensible. No és un llenguatge per si mateix sinó que es podria definir com una base per a realitzar nous llenguatges. És un dels objectius d'estudi d'aquest PFC.

DTD / XML - Schema: Un document XML pot tenir associat una DTD o un XML-Schema. Aquests contenen unes regles que ha de complir l'XML sobre el número i el tipus de dades que aquest conté. Una diferència important entre una DTD i un XML-Schema és que l'últim permet definir de forma més precisa les regles de validació del XML. La DTD és un dels objectius d'estudi d'aquest PFC.

OpenDocument o ODF: És un format de fitxers obert i estàndard per l'emmagatzemament de documents ofimàtics tals com fulls de càlcul, gràfics, texts i presentacions.

AODL: És una llibreria de lliure distribució desenvolupada plenament amb C# i dissenyada per manipular documents ODF.

BIBLIOGRAFIA

Llibres de Programació

Ceballos, F J. Enciclopedia de Microsoft Visual C#. Editorial Ra-ma. 2a Edición: 07/2007.

Kingsley-Hughes, Adrian, Kingsley-Hughes, Kathie. C# 2005. Editorial Anaya Multimedia.

Pàgines web

Informació sobre OpenDocument

Revista de la Associació de Tècnics d'Informàtica
http://www.ati.es/novatica/2006/184/nv184sum.html

Wikipedia
http://es.wikipedia.org/wiki/OpenDocument

Llibre online: OASIS OpenDocument Essentials
http://books.evc-cit.info

Online Community for the OpenDocument Format OASIS Standard
http://opendocument.xml.org

OASIS – Advancing Open Standards for the Information Society
http://www.oasis-open.org

Informació sobre DTD

Tutorial sobre DTD
http://www.ulpgc.es/otros/tutoriales/xml/DTD.html

Wikipedia
http://es.wikipedia.org/wiki/DTD

Informació sobre PDF

Article sobre l'estandardització del format PDF.
http://www.theinquirer.es/2007/01/29/el_formato_pdf_sera_un_estanda.html

Wikipedia
http://es.wikipedia.org/wiki/Portable_Document_Format