Construcció d'una eina pel tractament automatitzat de fitxers OpenOffice

Jordi Feliu Prim
Enginyeria en Informàtica

Jordi Ferrer Duran
Consultor

08/01/2007

A totes les persones que m'han ajudat en aquests anys de carrera i, en especial, a la meva família i a la meva dona que han patit més directament les conseqüències.

Resum

Aquest projecte té per objectiu conèixer i manipular fitxers creats amb el paquet ofimàtic de software lliure OpenOffice. Aquests fitxers es guarden en format anomenat OASIS, format que pren com a base l'estàndard XML i l'adapta a les característiques específiques que s'han dissenyat pels documents OpenOffice.

Per tal d'assolir aquests objectius estudiarem la documentació sobre OASIS, XML i DTD, igualment estudiarem diferents alternatives per desenvolupar aplicacions sobre documents OpenOffice.

Per poder entendre més aquest format realitzarem una petita aplicació que indexi totes les paraules d'un document en un fitxer XML i que després afegeixi aquest índex al final del mateix document inicial i guardi aquest índex en un fitxer en format XML..

Estudiem dues alternatives a l'hora de fer el desenvolupament: podem treballar amb el SDK d'OpenOffice o bé podem fer un desenvolupament en Java estàndard sense utilitzar llibreries específiques d'OpenOffice i treballant amb llibreries XML.

La memòria reflecteix aquests objectius i s'estructura en 9 capítols:

- *Capítol 1:* Es presenta el projecte indicant-ne els objectius, els mètodes seguits i els continguts.
- *Capítol 2:* S'estudien els conceptes de software lliure, formats oberts i paquets ofimàtics.
- *Capítol 3:* Es presenten els passos fets per instal·lar, configurar i provar els entorns de desenvolupament per fer aplicacions sobre OASIS utilitzant Java i SDK.
- *Capítol 4:* S'estudien els formats XML i DTD.
- *Capítol 5:* Es presenta l'estructura bàsica d'un document en format OASIS.
- *Capítol 6:* S'estudia l'estructura bàsica del SDK d'OpenOffice.
- *Capítol 7:* S'explica l'estructura del fitxer XML on guardarem les paraules indexades i del fitxer DTD associat.
- *Capítol 8:* S'expliquen les característiques principals de l'anàlisi, disseny, desenvolupament, joc de proves i deploy de l'aplicació.
- *Capítol 9:* Es presenten les conclusions finals del projecte.

ÍNDEX

Figures

Capítol 1

Introducció

En aquest capítol es situarà el punt de partença del projecte, indicant-ne les motivacions per desenvolupar-lo i els seus objectius. Es mostrarà la planificació temporal i el mètode seguit per crear-lo. Finalment s'introduirà el contingut de la resta dels capítols que integren la memòria.

1.1 Justificació del PFC

Volem conèixer en profunditat el format dels fitxers creats amb el paquet ofimàtic de software lliure OpenOffice. Cada document creat amb OpenOffice comparteix una estructura bàsica amb independència del tipus de document que representi (text, full de càlcul, presentació, gràfic, etc...). Aquest format està composat per un conjunt de fitxers XML i s'anomena OASIS.

Per tal d'entendre millor com s'estructura un document OpenOffice desenvoluparem una petita aplicació per practicar la lectura, la modificació i la creació de documents en aquest format. En concret el que farem serà indexar un document de text creat amb el programa Writer (editor de text) amb totes les paraules que formen part del text i afegirem aquest índex al final del propi document. Definirem també per aquest índex uns quants atributs de la font per practicar també amb els atributs de les fonts i estils.

1.2 Objectius del PFC

Els objectius d'aquest projecte són quatre:

- *Conèixer l'estructura d'un document OpenOffice:* per això, s'ha d'estudiar el format OASIS en que es basen aquests documents. Però també tots els conceptes de diferents tecnologies relacionades amb OASIS com XML, DTD.
- *Treballar amb documents XML i els DTDs associats:* per poder realitzar aquest objectiu haurem de, per una banda, conèixer les llibreries Java de manipulació de fitxers JAR i XML, i per altra estudiar la creació de fitxers DTD per tal de validar documents XML..
- *Desenvolupar una petita aplicació per entendre millor els conceptes adquirits i veure la seva potencialitat:* desenvoluparem una petita aplicació en Java per tal de manipular un document creat amb OpenOffice afegint elements i comprovant que són reconeguts després pel paquet ofimàtic.

- *Estudiar diferents alternatives per poder crear aplicacions pròpies que treballin amb fitxers OpenOffice:* estudiarem el mercat per esbrinar diferents formes de manipular els documents en format OpenOffice.

1.3 Enfocament i mètode seguit

Per a la realització d'aquest projecte una gran quantitat d'esforços han anat destinats al coneixement dels conceptes relacionats amb el format Oasis i XML i. una altra part dels esforços han anat dirigits a estudiar les alternatives que ofereix OpenOffice per fer desenvolupaments propis (SDK). També s'han dedicat esforços a conèixer les eines de desenvolupament del projecte (JDeveloper).

Per tant podriem dir que hem invertit la meitat del temps en comprendre els conceptes relacionats amb el projecte i l'altra meitat l'hem destinada al desenvolupament d'aquest. Així, els processos a seguir per a cada bloc, han estat:

- *Obtenir informació sobre el format OASIS i altres formats relacionats:* s'obté informació sobre aquest format de la mateixa forma que podem provar senzills exemples d'aplicació. També recollim informació sobre altres formats directament relacionats amb Oasis com ara XML i DTD.
- *Obtenir informació sobre el SDK d'OpenOffice:* s'obté informació sobre aquest entorn de desenvolupament i es realitzen alguns senzills exemples per veure algun cas de funcionament.
- *Desenvolupar una petita aplicació:* un cop tenim la suficient informació teòrica processada desenvolupem una petita aplicació per veure a la pràctica com funciona tot. Elaborem uns jocs de proves per tal de verificar el correcte funcionament de l'exercici.

1.4 Planificació del projecte

La planificació del projecte està resumida a la taula 1, on apareixen el nom dels apartats en que s'han dividit les tasques per realitzar el projecte i una petita descripció del contingut. Sobre cada apartat, apareix la data d'inici i la data de finalització de l'apartat.

Apartat	Data inici	Data fi
Pla de treball: L'objectiu d'aquest apartat és la creació de la planificació pla de treball per poder fer un seguiment de l'evolució del treball, segons el que estava previst inicialment.	21-09-06	02-10-06
Estudi del SDK d'OpenOffice: Consisteix en adquirir els coneixements necessaris per realitzar un projecte relacionat amb els SIG. Així, es buscarà informació sobre aquest camp i es recollirà per a llur anàlisi i la seva posterior inclusió en la memòria.	07-10-06	19-10-06

Format fitxers OASIS: En aquest bloc es pretén conèixer amb un cert detall l'estructura dels principals fitxers d'un document OpenOffice.	20-10-06	01-11-0
Estudi d'XML i DTD: Té com a objectiu l'estudi dels formats relacionats amb el projecte.	02-11-06	04-11-06
Anàlisi, disseny, desenvolupament de l'aplicació: Aquest bloc esdevé la part central del projecte. Posem en pràctica els cocneptes adquirits en els apartats anteriors.	07-11-06	11-12-06
Revisió de la memòria i creació de la presentació: Aquest bloc és la part en què es generarà la nova documentació, es revisarà la ja entregada i es presentarà com a resultat final del treball. Aquesta documentació ha de reflectir l'assoliment dels objectius marcats en l'inici del treball.	12-12-06	08-01-07

Fig 1. Planificació del projecte

1.5 Productes obtinguts

En aquest projecte, per a la seva definició, s'han de generar dos productes:

- *Memòria:* És aquest document on, a banda de detallar la feina feta, també es donen els conceptes necessaris per conèixer el format dels documents OpenOffice en general i les característiques de la petita aplicació realitzada.
- *Projecte:* Aquest producte inclou tant el programari creat per estudiar el format OASIS com els jocs de proves que hem utilitzat per provar l'aplicació.

1.6 Descripció dels capítols de la memòria

Els capítols que segueixen són 7, dels quals els primers permeten introduir-se en la problemàtica del treball i la instal·lació dels entorns informàtics per realitzar el projecte. Mentre que els darrers es centren en el desenvolupament d'una aplicació per posar en pràctica els coneixements adquirits:

- *Capítol 1:* Es presenta el projecte indicant-ne els objectius, els mètodes seguits i els continguts.
- *Capítol 2:* S'estudien els conceptes de software lliure, formats oberts i paquets ofimàtics.
- *Capítol 3:* Es presenten els passos fets per instal·lar, configurar i provar els entorns de desenvolupament per fer aplicacions sobre OASIS utilitzant Java i SDK.
- *Capítol 4:* S'estudien els formats XML i DTD.
- *Capítol 5:* Es presenta l'estructura bàsica d'un document en format OASIS.
- *Capítol 6:* S'estudia l'estructura bàsica del SDK d'OpenOffice.

- *Capítol 7:* S'explica l'estructura del fitxer XML on guardarem les paraules indexades i del fitxer DTD associat.
- *Capítol 8:* S'expliquen les característiques principals de l'anàlisi, disseny, desenvolupament, joc de proves i deploy de l'aplicació.
- *Capítol 9:* Es presenten les conclusions finals del projecte.

Capítol 2

Introducció al software lliure i Paquets Ofimàtics

2.1 Software Lliure

Fem una breu introducció al software lliure doncs en el nostre projecte utilitzarem programari d'aquest tipus. Aquest software es caracteritza perquè un cop finalitzat pot ser utilitzat, copiat, modificat i redistribuït lliurement. De la mateixa forma el software gratuït o freeware està acompanyat ocasionalment del codi font tot i que no es garanteixen els matexios drets que en el software lliure.
També existeix el software de domini públic, aquest no necessita cap llicència i no té drets d'autor.
El software lliure té diferents nivells de llibertat:

- Llibertat 0: execució del programa.
- Llibertat 1: estudiar i modificar el programa.
- Llibertat 2: còpia del programa.
- Llibertat 3: millorar el programa i fer públiques les millores.

També ens trobem amb diferents tipus de llicències. Destaquem:

- Llicència GNU GPL: l'autor conserva els drets i permet la redistribució i modificació sota uns determinats termes. Això implica que no es puguin crear productes amb parts no llincenciades GNU GPL. Si afegim una part que no prové de GNU GPL automàticament s'ha de llicenciar com a GNU GPL.
- Llicències estil MPL: evita el funcionament tan restrictiu del sistema anterior GPL promovent la col·laboració entre desenvolupadors.

El programari lliure ha tingut una clara utilització en situacions on l'usuari no té recursos econòmics o no vol pagar llicències. Això passa tant a nivell d'usuaris particulars com a nivell d'institucions i empreses privades. Per exemple ha estat adoptat per diversos països del tercer món i avui en dia cada cop més les administracions públiques tendeixen a utilitzar-lo substituint altres programaris de pagament.

Hi ha certa controvèrsia sobre la seguretat del software lliure a causa de ser codi obert i per tant sembla possible trobar els punts dèbils de l'aplicació mentre que el software no lliure, al no proporcionar el codi, sembla més segur. En canvi hi ha estudis sobre seguretat basats en determinar els forats de seguretat d'un producte encara no corregits. Per exemple, hi ha estudis fets sobre aquest tema on s'indica que els programaris de software lliure tenen pocs o cap error de seguretat que resti per solventar, mentre que en el mateix moment de l'estudi altres programaris propietaris amb funcionalitats equivalents tenien bastants errors de seguretat no corregits.

2.2 Formats Oberts

Es tracta d'una especificació per emmagatzemar dades digitals publicada i patrocinada normalment per una organització d'estàndards oberts. Un format obert ha de poder ser implementat per un programa propietari o lliure, per contra tenim els formats propietaris controlats i definits per interessos privats.

L'objectiu dels formats oberts és garantir a llarg termini l'accés a les dades guardades sense dependre de les decisions d'una companyia com passa amb els formats propietaris.
Un altre objectiu dels formats oberts és fomentar la competència.

Ens podem trobar qualsevol combinació entre les variables programari lliure/propietari i format obert/propietari. Per exemple tenim el llenguatge obert de format de textes d'internet HTML que és la base de programes propietaris com Internet Explorer de Microsoft o de programes lliures com Mozilla Firefox. De la mateixa forma el programa lliure d'oficina OpenOffice permet manipular formats propietaris com DOC, XLS o PPT de Microsoft i formats oberts com ODT, ODS i ODP d'OpenDocument. També tenim casos en que fabricants de software propietari fan públics els seus formats perquè d'altres desenvolupadors els utilitzin en les seves aplicacions, com ara Adobe System ha fet amb PDF o Microsoft amb RTF.

Les administracions estan començant a prendre mesures en aquest sentit demanant que els arxius amb informació pública estiguin emmagatzemats amb formats oberts. Així l'estat de Massachusetts ha decidit que a partir del 2007 els seus arxius públics han d'estar en formats OpenDocument d'OASIS.

2.3 Paquets Ofimàtics

En el nostre projecte utilitzarem un paquet ofimàtic de software lliure que utilitza tant formats oberts com formats propietaris. En el cas del projecte utilitzarem només els formats oberts. Aquest paquet és OpenOffice basat en el paquet StarOffice de Sun Microsystems. El codi font va ser alliberat l'any 2000 amb la intenció de fer la competència al gran paquet ofimàtic propietari del mercat (Microsoft Office). El codi font de l'aplicació està disponible sota la llicència LGPL.
El nom d'OpenOffice és el nom amb que es coneix popularment, però no és el nom oficial ja que aquest està registrat. El nom oficial d'aquest paquet ofimàtic és OpenOffice.org o OOo. Nosaltres utilitzarem la versió portable d'aquest paquet que funciona des d'una memòria USB.
El format propi dels documents OpenOffice.org està basat en l'estàndard de format obert XML.

Capítol 3

Instal·lació i proves de l'entorn de desenvolupament

Els primers passos per a la realització del treball pràctic són la instal·lació, configuració i proves de l'entorn de desenvolupament.

Ens hem basat en les indicacions del document "OpenOffice.org 1.1. Software Development Kit" i en coneixements propis d'eines del mercat actual.

Tenim dues línies d'investigació inicials per resoldre el treball pràctic.
Una primera tractaria de resoldre el problema utilitzant simplement les especificacions OASIS dels documents OpenOffice i utilitzar les llibreries de Java per al tractament de fitxers JAR. Això ens porta, segurament, a fer una aplicació de línies de comanda del tipus:

> AfegirIndex <FitxerEntrada>

Una segona via d'investigació consisteix en estudiar les possibilitats de les API d'OpenOffice que utilitzen objectes anomenats UNO (Universal Network Objects) per tractar documents OpenOffice. Aquesta opció ens pot permetre, per exemple, integrar la solució dins del mateix paquet OpenOffice en forma d'Add-ins.

Passem a descriure els passos per instal·lar, configurar i provar l'entorn de desenvolupament per treballar en les dues solucions, subcapítols 3.1 i 3.2.

3.1 Entorn API OpenOffice

Consideracions prèvies

El llenguatge de programació que utilitzarem serà Java. Sabem que les API d'OpenOffice suporten els llenguatges Java, C++, Basic d'OpenOffice i Objectes COM/DOM de Microsoft.

Per altra banda la tecnologia UNO està disponible en les plataformes Microsoft, Linux, Solaris, MAC OS X, Free BSD i Power PC. Nosaltres utilitzarem un entorn de desenvolupament Microsoft.

Llenguatge de Programació: Java.
Plataforma: Microsoft.
Sistema Operatiu: Windows XP Home Edition Version 2002 amb Service Pack 2.
Maquinari: Intel Celeron a 2.53GHz amb 480Mb de RAM.

Instal·lacions

1- Seguint les indicacions del capítol 2 ("Firts steps") a l'apartat 2.3 Getting started comencem per descarregar el programari OpenOffice. Ho realitzarem amb una versió portable que no caldrà instal·lar al nostre ordinador, ho farem en un llapis USB així podrem disposar d'ella en qualsevol entorn.
Anem al web http://www.portableapps.com per descarregar aquesta versió

Fig 2. Pantalla d'arrencada d'OpenOffice Portable

Un cop descomprimit en qualsevol carpeta ja podem iniciar l'aplicació d'ofimàtica executant OpenOfficePortable.exe per provar que funciona correctament. La versió que tenim és la 2.0.3.

Fig 3. Pantalla d'informació (about) d'OpenOffice.

2- Ens decarreguem des del web oficial d'OpenOffice (http://www.openoffice.org) la documentació i el software del SDK. És el fitxer OOo_1.1.0_Win32Intel_sdk.zip ja que treballarem en una plataforma windows.

Guardem els arxius descomprimits al nostre disc dur en qualsevol carpeta. Busquem dins la documentació de l'API per Java d'OpenOffice. La trobem a l'adreça relativa <SDK_OpenOffice_Path>\docs\java\ref\index.html.

Fig 4. JavaDoc del SDK d'OpenOffice.

3- Cal instal·lar-se una versió del SDK de Java. Anem al web de Sun (http://www.sun.com) i descarreguem la darrera versió. En aquest cas és la versió 1.5.0 o Java 5.

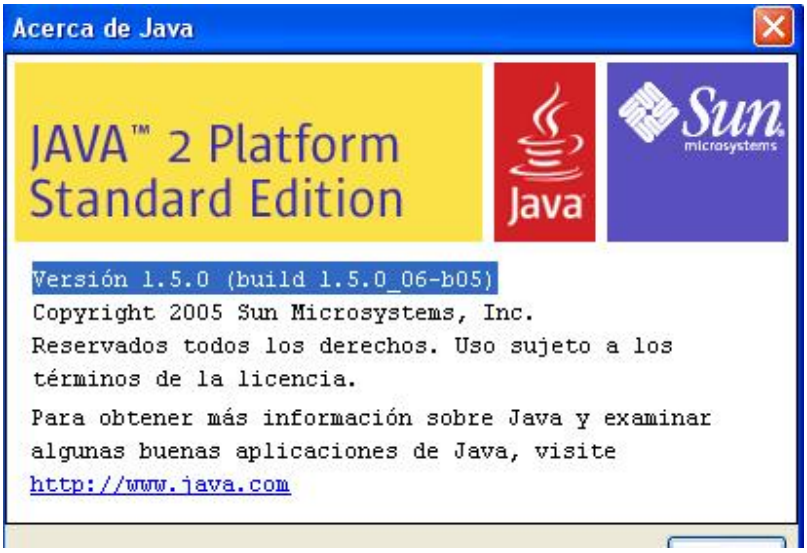

Fig 5. Pantalla d'informació (about) de SDK de Java.

Configuracions

1- Cal indicar al paquet ofimàtic OpenOffice on tenim instal·lada la màquina virtual de Java (JVM). Per tant obrim el paquet ofimàtic (OpenOfficePortable.exe) i anem a l'opció Option → Tools.
Escollim de la dreta el bloc OpenOffice.org i l'apartat Java, aleshores a la dreta escollim el JRE que volem utilitzar entre els que troba el nostre ordinador.

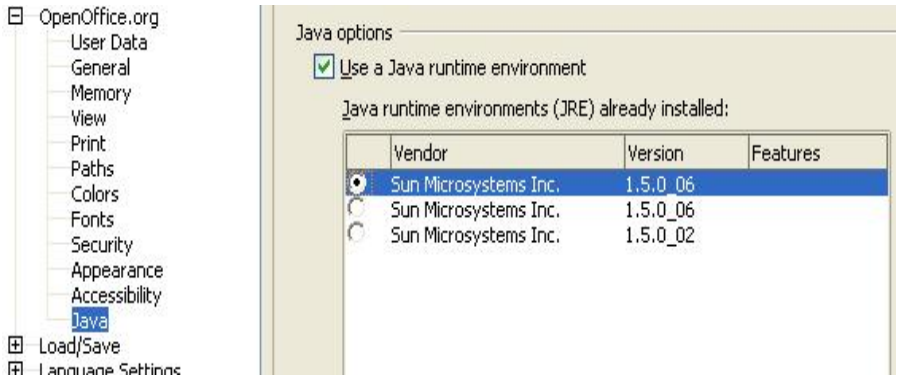

Fig 6. Configuració de Java a OpenOffice.

2- És recomanable disposar d'un entorn de desenvolupament integrat per treballar amb Java. El document d'OpenOffice recomana els entorns NetBeans (http://www.netbeans.org) i l'One Java Studio de Sun perquè ofereixen millor integració amb UNO. Inicialment començarem treballant amb el producte gratuït JDdevelopper (http://www.oracle.com/technology/products/jdev/) en la versió 10.1.3.
Aquest producte és l'adaptació que ha fet Oracle de l'eina d'IBM Eclipse.

Fig 7. Pantalla d'arrencada del JDevelopper d'Oracle.

Cal indicar al JDevelopper on tenim les classes Java d'UNO per tal de poder-les utilitzar en els nostres projectes.
Després d'haver creat un projecte, anem a Tools → Project Properties i escollim dins de la llista de l'esquerra l'apartat Libraries. Ara podem escollir amb el botó Add Jar/Directory els fitxers Class de l'API de UNO. D'aquesta forma l'IDE pot trobar les classes d'UNO en temps d'edició al fer un Import.
Les classes estan localitzades a la ruta <OpenOffice Path>/Program/classes.

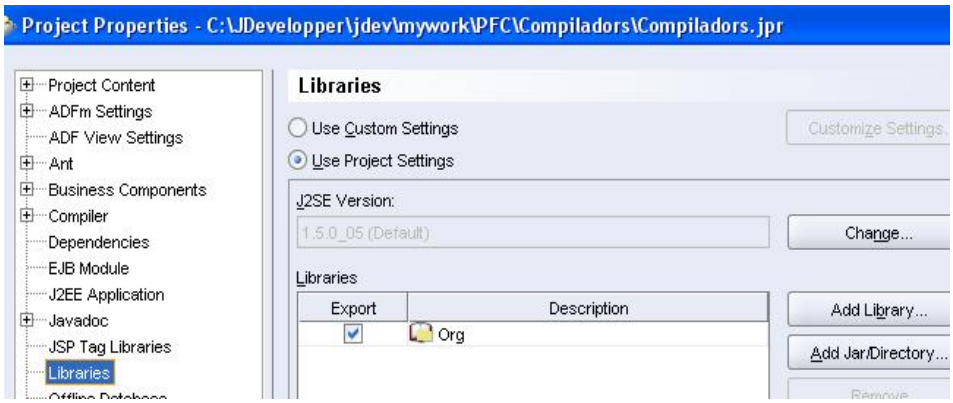

Fig 8. Configuració de llibreries al IDE JDevelopper.

Exemples

Hem provat l'exemple que apareix a la documentació del SDK *<OpenOffice Path>/examples\DevelopersGuide\FirstSteps\FirstConnection.java* per tal de provar la connexió amb OpenOffice a través del port 8100. També hem provat el programa HelloTextTableShape.java que obre diferents aplicacions OpenOffice, crea arxius i els combina en un de sol.

Documentació

Caldrà consultar la documentació de les classes natives de Java, així com les classes d' UNO (*<OpenOffice Path>\docs\java\ref\index.html*) per connectar amb documents OpenOffice.

3.2 Entorn OpenOffice amb OASIS

En el cas de treballar sense utilitzar les APIs d'OpenOffice ens calen els següents passos:

Instal·lació

1- Com abans cal descarregar una versió d'OpenOffice per, com a mínim, poder crear/veure els documents. Ja ens serveix el que hem fet anteriorment descarregant la versió portable d'OpenOffice.

Configuració

1- Ens instal·lem l'entorn de desenvolupament integrat (IDE) Jdevelopper. Ens afegirem unes llibreries de tractament de fitxers XML (per treballar amb OASIS), aquestes les trobem a http://xml.apache.org/, les copiem al nostre disc dur i li indiquem a Jdevelopper on són: Tools → Project Properties i a l'apartat Libraries fem Add Jar/Directory anant a indicar la carpeta on hem copiat aquests arxius. Amb això podem provar alguns exemples per llegir/modificar el fitxers OpenOffice que estan en format OASIS (basat en XML), així com per desenvolupar el projecte.

Exemples

Hem provat l'exemple que apareix a la documentació sobre el format OASIS (OD_Essencials.pdf) que hem descarregat del web oficial d'OpenOffice
(http://www.openoffice.com) per afegir capçaleres a un document OpenOffice. Es tracta del programa AddOutline.java del capítol 3 Text Documents Basics.

Documentació

Caldrà consultar la documentació de les classes natives de Java, així com les classes per parsejar documents en xml. També caldrà consultar la documentació relativa a OASIS per saber quins elements xml hem de modificar en el document OpenOffice.

Capítol 4

Introducció al formats XML i DTD

La raó per dedicar un apartat a estudiar XML es perquè els documents d'OpenOffice estan basats en un format derivat de XML..
El format XML (eXtensible Markup Language) és un format associat a l'estructura de dades i documents de la tecnologia internet, per exemple el format HTML també està basat en XML. XML a la seva vegada està basat en un altre llenguatge de marques anomenat SGML. De fet XML i HTML neixen d'aquest format. Va ser creat per tenir un mecanisme de transmissió d'informació entre màquines, cal destacar que és un fitxer en format textual i no binari.
Una característica destacada de XML és que l'estructura del document queda especificada en el mateix document. Aquest tipus de fitxers s'ha anat imposant en els darrers temps.
En relació a HTML, XML té molt poques restriccions sintàctiques. Gràcies a aquesta flexibilitat és molt senzill per a qualsevol desenvolupador dissenyar una estructura de dades del seu intères en XML. Així, per exemple, en relació a HTML, és el dissenyador de l'aplicació qui decidirà els noms de les etiquetes.
Així podem definir una estructura de dades en forma d'arbre mitjançant l'anidament d'etiquetes i a més incloure les dades amb l'estructura que hem creat.

Per exemple veiem una part del contingut d'un arxiu de configuració del programa PowerDVD:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<VideoDescription>
        <Item id="1">
                <name>Mostrar vídeo sin mejora.</name>
        </Item>
        <Item id="2">
                <name>Optimizar automáticamente vídeo para verlo en
                panel LCD (se recomienda).</name>
        </Item>
        <Item id="3">
                <name>Mostrar vídeo con tonos saturados optimizado para
                verlo en entorno de sala de cine oscura.</name>
        </Item>
</VideoDescription>
```

4.1 Declaració, elements i atributs

A la primera línia tenim la ***declaració XML***. Aquí indiquem quina versió d'XML utilitzarem (aquesta part és obligatòria) i quin joc de caràcters utilitza l'arxiu. En aquest cas ISO-8859-1 sinó, per defecte s'utilitza UTF-8 (unicode).
A partir de la segona línia comença la informació. En aquest punt cal distingir entre elements i atributs. Un ***element*** és un identificador col·locat entre dues branques (< , >). Per exemple <VideoDescription>. Aquest element conté tota la informació que hi ha entre la marca d'inici ("<nom Element>") i la marca final d'element identificada perquè afegeix una contrabarra a la marca d'inici després de la marca '<' ("</nom Element>"). Veiem que afegim més elements

per definir VideoDescription, en aquest cas “Item” que a la seva vegada té un subelement “name”.

Podem tenir elements sense informació. Per exemple

```
<Item id="4">
      <name> </name>
</Item>
```

En aquest cas tenim l’element name, que pertany a l’item amb id=4 que no té valor.

Veiem que item inclou un identificador dins del tag inicial. És el que anomenem un ***atribut***. Aquest atribut, contràriament al que passava amb els elements ha de ser forçosament textual, com veiem que estan delimitats per cometes simples o dobles. Normalment el sentit dels atributs és ser metadades dels elements. En aquest exemple serveix per donar un identificador únic a cada element <item>. Podem tenir més d’un atribut per element i en principi no importa en quin ordre estiguin, el que sí és important és l’ordre dels elements. D’aquesta forma cal tancar els tags en l’ordre invers en que s’han obert.

Tots els atributs han de tenir valor i ha de ser textual. Així aquests exemples són incorrectes.

```
<Item id=4> o <Item id>
```

4.2 Sintaxi, comentaris, caràcters especials, NamesSpaces

Sintaxi

Els noms dels tags són case-sensitive, els noms que comencen per ‘xml’ estan reservats i un nom ha de començar amb una lletra o un guió baix.

Tenim una descripció complerta a http://www.w3.org/TR/REC-xml#sec-common-syn

Comentaris

Podem incloure comentaris com en HTML. Aquests comencen amb “<!—“ i acaben amb els caracters “-->”.

Caràcters especials

Podem incorporar caràcters reservats d’xml utilitzant la taula d’equivalències següents:

Literal character	Entity reference
<	<
>	>
'	'
"	"
&	&

Fig 9. Taula de correspondències de caràcters especials.

XML NamesSpaces

XML ens proporciona un mecanisme per diferenciar elements de diferents arxius amb el mateix nom però diferent significat. Que fer si la nostra aplicació ha de llegir en un moment donat dos fitxers amb significat diferent però que tenen elements amb el mateix nom i crear un nou document barrejant elements amb el mateix nom?. XML ens permet afegir un prefix en forma d'únic URI a cada document XML. D'aquesta forma aquest prefix ens permet referenciar-los sense problema d'inconsistència. Per exemple:

```
<doc:document xmlns:doc="http://exemple.com/document"
xmlns:bbdd_1="http://exemple.com/dades_bbdd1"
xmlns:bbdd_2=http://exemple.com/dades_bbdd2>
<doc:linia id="1">
        <bbdd_1:linia> Valor 1 </bbdd_1:linia>
        <bbdd_2:linia> Valor 2 </bbdd_2:linia>
</doc:linia >
```

veiem com en aquest exemple hem pogut barrejar en un mateix document tres elements amb el mateix nom d'element ("***linia"***) però de diferents arxius.

4.3 DTD (Document Type Defintions)

El sentit de DTD és permetre a qui ha definit el format d'un fitxer XML especificar d'alguna forma la sintaxi que han de complir aquests fixters XML per considerar-se vàlids. Normalment el DTD és un arxiu que defineix els tipus d'elements i una llista d'atributs.

Aquest arxiu DTD el referenciem dins del document XML. D'aquesta forma en llegir el fitxer xml poden comprovar si està o no ben format segons la definició del fitxer DTD. En el cas d'un navegador el resultat d'un XML que no compleixi la especificació DTD resultarà que no mostrarà el document.
La forma d'incloure-ho dins de l'XML és amb la directiva DOCTYPE després de la definició de la versió d'xml.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE VideoDescription SYSTEM http://example.com/video.dtd>
<VideoDescription>
```

Aquesta part l'hem d'implementar en el treball pràctic per validar que la nostra aplicació ha creat un document XML ben format segons les nostres especificacions.

Les especificacions de DTD estan definides juntament amb les d'XML a l'adreça web (http://www.w3.org/TR/REC-xml).

Capítol 5

Introducció al format OASIS

5.1 Introducció

OASIS és l'especificació del format dels documents del paquet ofimàtic OpenOffice. Està basat en el format XML i vol representar una idealització del document que defineix.
En realitat un document OpenOffice està format per varis fitxers, alguns d'ells en format OASIS i tots ells guardats en format JAR per ocupar menys espai. El format JAR és en realitat un fitxer en format ZIP que afegeix el document manifest.xml on indica el contingut de l'arxiu.

Per exemple veiem el contingut de l'arxiu Hola Món.odt creat amb l'OpenOffice Writer.

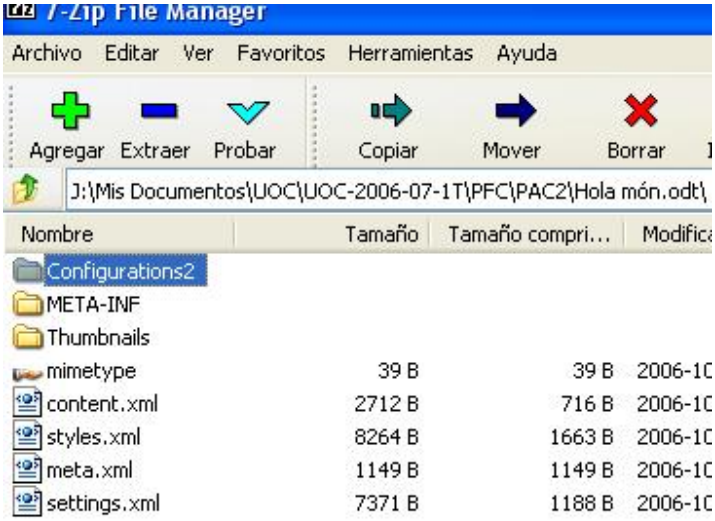

Fig 10. Contingut d'un fitxer en format OASIS.

El contingut dels arxius/carpetes més importants són:

META-INF/manifest.xml:
Té una llista de tots els arxius que hi ha dins del JAR. Són metadades del document i és imprescindible que es trobi dins del fitxer perquè OpenOffice el pugui llegir.

Si ens fixem en el contingut del fitxer manifest.xml veiem que defineix els elements <manifest:file-entry> per indicar amb l'atribut manifest:full-path el fitxer dins del JAR i, a l'atribut manifest:media-type, indica quin typus mime té el fitxer.

mimetype:
Té una línia de text que indica el tipus MIME del document

content.xml:
Contingut del document que hem editat.

styles.xml:
Informació dels estils utilitzats al document.

meta.xml:
Metainformació del document com per exemple autor, data de revisió, etc...

settings.xml:
Informació específica de l'aplicació a la que pertany el document (editor de text, full de càlcul, presentació, etc...).

Hem vist que al fitxer manifest.xml s'utilitza el namespace "manifest". OpenOffice té bastants namespaces predefinits com aquest que trobarem als fitxers contant.xml, styles.xml i settings.xml

Els mínims documents que necessita un fitxer OpenOffice per ser llegit són el META-INF/manifest.xml i content.xml.

5.2 Arxius bàsics d'un document OpenOffice

Veiem amb més detall els arxius amb que haurem de treballar:
Podem dir que el fitxer bàsic del document és content.xml i té al seu voltant tres arxius auxiliars que són meta.xml, styles.xml i contents.xml.

setting.xml

Com ja hem dit inclou informació que utilitzarà exclusivament l'aplicació que ha creat l'arxiu, com ara mides de la pantalla, posició, nom de la impressora, etc...
No hem d'utilitzar aquest fitxer en el nostre treball, per tant no aprofundirem en més detalls.

meta.xml

Inclou informació sobre el propi document. La major part dels elements d'aquest fitxer són accessibles a través del paquet ofimàtic OpenOffice mitjançant el menú (File → Properties).

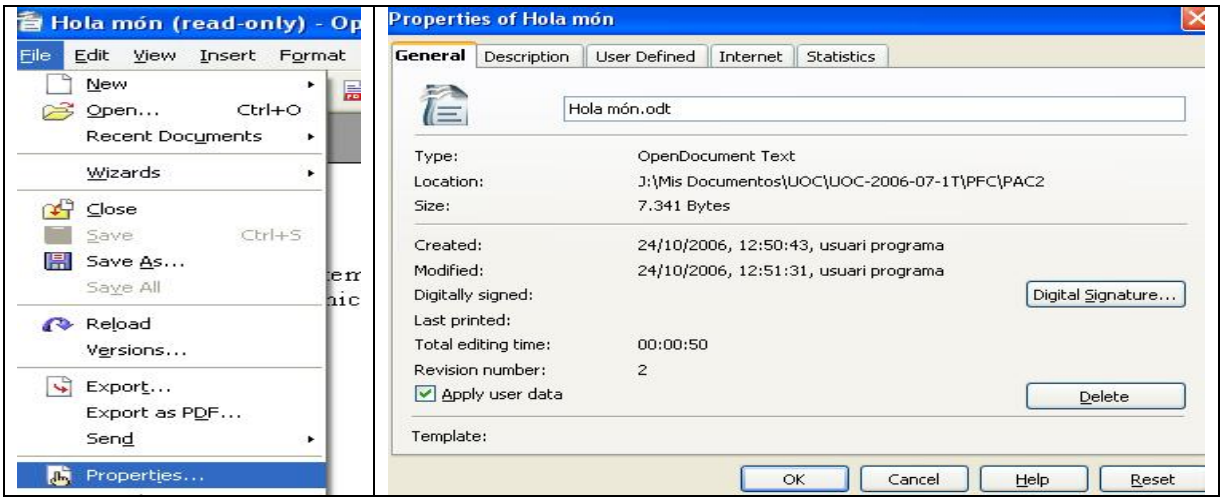

Fig 11. Accés a la pantalla de propietats d'un document OpenOffice.

Als apartats “General Document Properties”, “Document Description”, “User-Defined Information”, “Document Statistics” alguns elements d’aquest fitxer tenen NameSpaces heredats de Dublin Core (dc) i d’altres del mateix namespace que defineix aquest arxiu (meta).
No és obligatori modificar aquest document en el nostre projecte, però hi ha alguns camps que sí podriem modificar en generar l’índex com ara la data de modificació del fitxer, número de revisió, afegir un comentari i després la informació estadística com ara el número de pàgines, número de paraules o número de caràcters. Veiem quins elements hauriem de modificar.

En el nostre exemple “Hola Mon.Odt” veiem el fitxer meta.xml.

```
<?xml version="1.0" encoding="UTF-8" ?>
<office:document-meta
xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0"
xmlns:ooo="http://openoffice.org/2004/office" office:version="1.0">
<office:meta>
      <meta:generator>OpenOffice.org/2.0$Win32
      OpenOffice.org_project/680m7$Build-9044</meta:generator>
      <meta:initial-creator>usuari programa</meta:initial-creator>
      <meta:creation-date>2006-10-24T12:50:43</meta:creation-date>
      <dc:creator>usuari programa</dc:creator>
      <dc:date>2006-10-24T12:51:31</dc:date>
      <dc:language>es-ES</dc:language>
      <meta:editing-cycles>2</meta:editing-cycles>
      <meta:editing-duration>PT50S</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="0" meta:image-count="0"
      meta:object-count="0" meta:page-count="1" meta:paragraph-
      count="2" meta:word-count="8" meta:character-count="47" />
</office:meta>
</office:document-meta>
```

Tots els elements a modificar estan dins d’ <office:document-meta> <office:meta>
Data de modificació: correspon a l’element **dc:date.**

```
<dc:date>2006-10-24T12:51:31</dc:date>
```

La data està en format ISO 8601, o sigui 4 dígits per l’any, dos dígits pel mes, dos dígits pel dia separats per guions, la “T” separa la data de l’hora. Aquesta està en format hh:mm:ss.
Cal actualitzar aquesta data amb la data del sistema en el moment de la modificació.

Número de revisions: correspon a l’element **editing:cycles.**

```
<meta:editing-cycles>2</meta:editing-cycles>
```

Cal agafar el número i incrementar-lo en 1.

Estadístiques: correspon a l'element **document:statics.**

```
<meta:document-statistic meta:table-count="0" meta:image-count="0"
meta:object-count="0" meta:page-count="1" meta:paragraph-count="2"
meta:word-count="8" meta:character-count="47" />
```

Modificarem diferents atributs:
meta:page-count pel número de pàgines.
meta:paragraph-count pel número de paràgrafs.
meta:word-count pel número de paraules.
meta:character-count pel número de caracters.

Cal que la nostra aplicació compti tots aquests valors un cop modificat el document i els anoti dins dels atributs.

styles.xml

Té la informació sobre els estils utilitzats al document, part d'aquesta informació està duplicada a content.xml.
Els apartats més importants són la declaració de fonts, declaració d'estils, declaració d'estils automàtics i els estils màster.

Declaració de fonts: està dins de l'element <**office:font-face-decls**> i correspon a l'element **style:font-face.**

```
<office:font-face-decls>
   <style:font-face style:name="Tahoma1" svg:font-family="Tahoma" />
</office:font-face-decls>
```

Té els següents atributs:
style:name per al nom de la font (obligatori).
svg:font-family per a la família de la font (opcional).
style:font-family-generic per a la família genèrica d'aquesta font (opcional).
style:font-pitch indica si la font és de mida fixa o variable (opcional).
style:font-charset indica la codificació d'aquesta font (opcional).

Dins de l'element office-styles trobem un contenidor d'estils per defecte amb el seu nom. En el cas de fulls de càlcul trobem informació sobre l'estil dels números, monedes, etc... Els subelements més importants que trobem són style:default-style i style:style, els dos tenen un atribut anomenat style:family que indica a quin nivell s'aplica l'estil (caràcter, paràgraf, taula, secció,etc..)

Declaració d'estils: està dins de l'element <**office:styles**>, correspon a l'element **style:default-style.**

```
<style:default-style style:family="paragraph">
        <style:paragraph-properties fo:hyphenation-ladder-count="no-limit"
        style:text-autospace="ideograph-alpha" style:punctuation-
        wrap="hanging" style:line-break="strict" style:tab-stop-
        distance="1.251cm" style:writing-mode="page" />
        <style:text-properties style:use-window-font-color="true" style:font-
        name="Times New Roman" fo:font-size="12pt" fo:language="es"
        fo:country="ES" 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"
        fo:hyphenate="false" fo:hyphenation-remain-char-count="2"
        fo:hyphenation-push-char-count="2" />
</style:default-style>
```

En l'atribut definim el nom de la família d'estil (style:family) que utilitzarem a l'atribut família de l'atribut style:style.

Veiem que té diferents subelements com style:paragraph-properties i style:text-properties. Ens interessem per aquest darrer que haurem d'utilitzar dins del projecte.
Aquest element té molts atributs que defineixen aquesta familia com el nom de la font, mida de la font.

Declaració d'estils: està dins de l'element <**office:styles**>, correspon a l'element **style: style.**

```
<style:style style:name="Text_20_body" style:display-name="Text body"
style:family="paragraph" style:parent-style-name="Standard"
style:class="text">
        <style:paragraph-properties fo:margin-top="0cm" fo:margin-
        bottom="0.212cm" />
</style:style>
```

En aquest exemple veiem alguns dels atributs d'aquest estil:
style:name és el nom de l'estil.
style:display-name és el nom que apareix a OpenOffice.

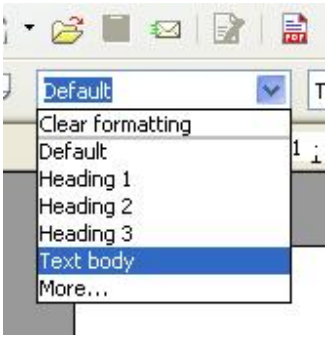

Fig 12. Detall de la selecció d'estils a OpenOffice.

style:family indica la família de l'estil que correspon a un element d'style-default-style
style:class indica quin és l'àmbit d'aplicació. En aquest cas és a nivell de caràcter.

Veiem que inclou el subelement <style:paragaph-properties> que especifica en els atrributs les coordenades on comença el paràgraf.

content.xml

Té un format variable segons el tipus de document, però hi ha una sèrie d'elements que sempre trobarem.

En primer lloc l'element pare <office:document-content> amb els namespaces que utilitzarem i l'atribut office:version que indica la versió d'OpenOffice.

Té els següents subelements:
<office:scripts> que indica els scripts que tenim al document (opcional)
<office:font-face-decl> indica les fonts que utilitzem al document i duplica la informació de styles.xml. Té dos atributs indicant el nom (style:name) i la família (font:family) de la font (opcional).
<office:body> és la part més important del document i és obligatori. El primer subelement que trobem indica el tipus de document, així podem tenir <office:text>, <office:drawing>, <office:presentation>, etc... En el nostre cas trobarem <office:text>.

```
<office:body>
        <office:text>
                <office:forms form:automatic-focus="false" form:apply-design-
                mode="false" />
                <text:sequence-decls>
                <text:sequence-decl text:display-outline-level="0"
                text:name="Illustration" />
                <text:sequence-decl text:display-outline-level="0"
                text:name="Table" />
                <text:sequence-decl text:display-outline-level="0"
                text:name="Text" />
                <text:sequence-decl text:display-outline-level="0"
                text:name="Drawing" />
                </text:sequence-decls>
                <text:p text:style-name="Standard">Petit exemple de
                document</text:p>
                <text:p text:style-name="Standard">Prova inicial Hola
                Món</text:p>
        </office:text>
</office:body>
```

Capítol 6

Introducció al SDK d'OpenOffice

6.1 Introducció

OpenOffice ens proporciona unes eines per accedir a la programació del paquet ofimàtic. Si en l'apartat anterior hem vist que podem accedir al document OpenOffice sense obrir el paquet Ofimàtic gràcies a que coneixem el format dels arxius (OASIS) també podem accedir al document des del mateix paquet ofimàtic i fer, per exemple, un botó dins del paquet ofimàtic que realitzi el treball que volem.

Per tal de donar accés a les aplicacions OpenOffice a través de les API s'ha creat la tecnologia UNO (Universal Object Network). Aquesta tecnologia està disponible en moltes plataformes com ara Solaris, Linux, Windows, Power PC, etc...
Els llenguatges de programació que suporta UNO són Java, C++, OpenOffice.org Basic i mitjançant components COM/DCOM de Microsoft.

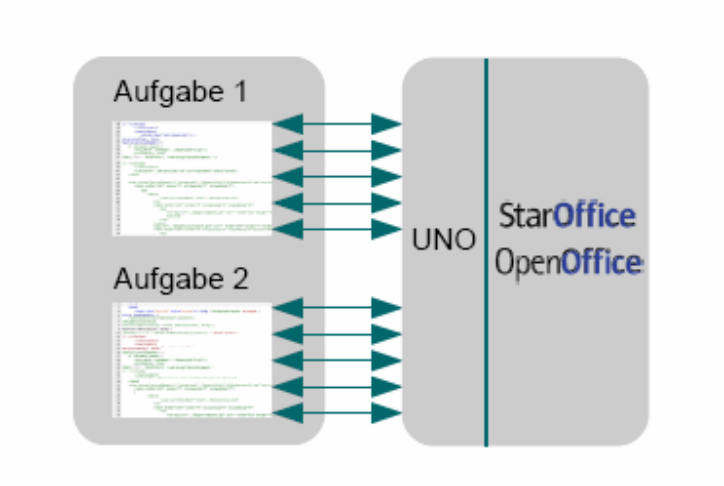

Fig 13. Esquema de connexió utilitzant UNO.

Podem connectar a instàncies locals o remotes d'OpenOffice.org a través d'aplicacions escrites amb els llenguatges de programació citats anteriorment, també mitjançant altres tecnologies com Servlets, pàgines JSP, Delphi, Visual Basic, etc...

En el nostre cas utilitzarem Java per realitzar el treball, les classes compilades Java per poder programar amb UNO es troben al directori <OfficePath>/program/classes.
La documentació JavaDoc sobre les classes està a <OfficePath>\docs\java\ref\index.html

6.2 Com fer una connexió

La comunicació entre les aplicacions escrites en Java utilitzant UNO i l'OpenOffice es fa a través del protocol TCP/IP. Per tant abans de fer qualsevol aplicació cal habilitar aquest canal de comunicació. Tenim dues formes de fer aquesta connexió:

1- **Listener sempre actiu**: Podem fer que escolti sempre que executem l'aplicació OpenOffice.

Per això cal modificar un fitxer de configuració d'OpenOffice. Es tracta del fitxer Setup.xcu que es troba a *<OfficePath>/share/registry/data/org/openoffice/Setup.xcu*.
Busquem l'element *<node oor:name="Office"/>.* Aquest element conté elements *<prop>*.
Cal afegir aquest element.

```
<prop oor:name="ooSetupConnectionURL" oor:type="xs:string">
 <value>socket,host=localhost,port=8100;urp;</value>
</prop>
```

Per defecte utilitzem el **port 8100** però si alguna altra aplicació l'utilitza podem canviar-lo per un altre port lliure.

2- **Listener de sessió**: Permetre la connexió quan nosaltres decidim fer-ho i durant la sessió en curs.

Per fer això cridarem el programa **soffice** des de la línia de comandes

<OfficePath>/program/soffice "-accept=socket,port=8100;urp;"

Així estarà actiu durant tota la sessió. Podem comprovar que està disponible el port des de la línia de comandes (entorn windows) amb la comanda

>netstat –a

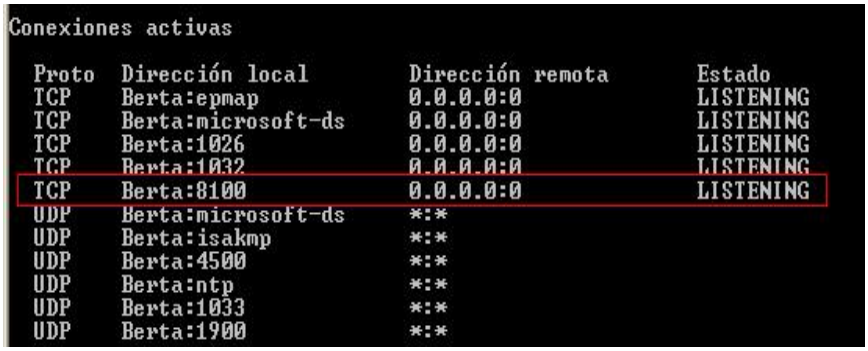

Fig 14. Detall del port 8100 amb el listener activat.

En la nostra aplicació caldrà que indiquem el port i el nom de la màquina on està el listener actiu.

6.3 Com funcionen les connexions

UNO treballa amb el concepte de ***services managers*** que són entitas que creen serveis (objectes UNO que permeten realitzar tasques concretes). Per exemple tenim els següents serveis:

- com.sun.star.frame.Desktop: informació sobre els documents carregats, permet obrir documents.

- com.sun.star.configuration.ConfigurationProvider: permet accedir a la informació de configuració d'OpenOffice (opcions del menú Tools – Options).

- com.sun.star.text.GlobalSettings: manipulació de configuracions d'impressió de documents de text.

Qualsevol servei existeix dins d'un ***component context*** format pel ***service manager*** que ha creat el servei ***més dades*** que utilitzarà el servei.

L'**arquitectura** de les aplicacions és **client – servidor**. El programa OpenOffice actua com a servidor i les aplicacions dels usuaris com a clients. El client i el servidor tenen el seu component context com el seu service manager. El service manager del client crea UNO objects al procés de client mentre que el service manager del servidor crea UNO objects en el procés del servidor.

Per poder parlar amb el servidor el client ha d'obtenir el service manager del servidor. A la pràctica això ho fem amb *com.sun.star.comp.helper.Bootstrap* que ens dóna un service manager bàsic que crea els serveis bàsics per parlar amb altres components context, un d'ells és l'objecte *UnoUrlResolver* que obté el component context i el service manager del servidor.

De fet un **servei** és una combinació d'interfaces i propietats encapsulades en un objecte. Una interface és un conjunt de mètodes que defineixen diferents aspectes del servei, per exemple l'interface *com.sun.star.view.XPrintable* permet accedir als mètodes print(), getPrinter() i setPrinter().

Una **propietat** és una part del servei que no es considera com estructural i és manipulada mitjançant mètodes genèrics del tipus getPropiertyValue() / setPropiertyValue(), en aquest cas només cal considerar la interface *com.sun.star.view.XPropiertySet*.

En el cas del nostre projecte haurem de treballar amb els serveis relacionats amb documents de text que són bàsicament:
com.sun.star.text.TextDocument i com.sun.star.document.OfficeDocument

L'origen del concepte de servei té varies raons:

- Separar especificació d'implementació: això permet en qualsevol moment redefinir el comportament dels mètodes proporcionats sense haver de modificar el codi.

- Utilitzant noms de serveis permet crear instàncies per nom d'especificació i no per nom de classe: això ens permet crear un objecte per un cert propòsit i el service manager decideix a partir del service name obtingut quina implementació retorna. Des del punt de vista de l'usuari no hi ha problema mentre ens ajustem a les interfaces i propietats del servei.

- El servei permet fer una implementació d'interfaces de serveis més precisa: això implica una major reusabilitat de les interfícies.

- El servei permet manipular una gran quantitat de propietats no estructurals: això evita tenir que utilitzar molts mètodes get i set si aquestes propietats estiguessin dins de la interface.

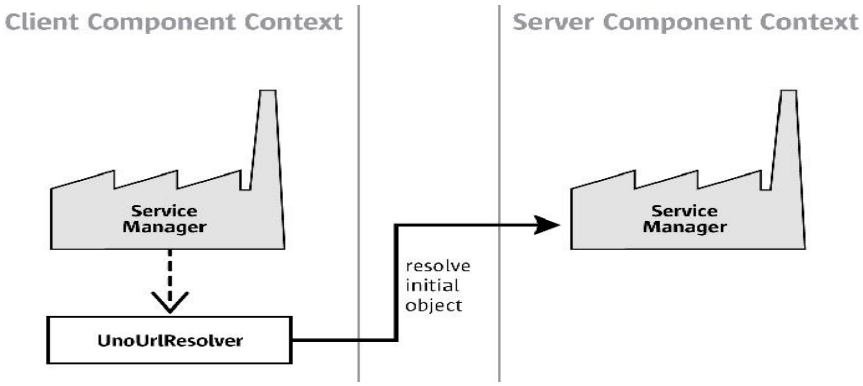

Fig 15. Detall de la petició del component context.

En cas d'error en la connexió amb el programa servidor (OpenOffice) hem de capturar l'excepció Java *com.sun.lang.DiposedException* per tal de poder controlar aquesta mena d'errors i poder reprendre la connexió.

Capítol 7

Fitxer XML de definició de l'índex

7.1 Format

Definirem l'arxiu de les paraules de l'índex en un fitxer amb un nom qualsevol i extensió 'xml'. Aquest fitxer contindrà una descripció dels mots indexats, els atributs que utilitzarem per representar-los en el document OpenOffice i una indicació sobre on apareixen en el text.

En aquesta petita aplicació només definirem uns quants atributs per mostrar el treball pràctic, en tot cas sempre serà possible millorar aquesta solució afegint més opcions.

L'objectiu final d'aquest fitxer és crear-lo a partir de les paraules trobades en el document, i guardar-lo, juntament amb les seves referències i els atributs per representació en el document OpenOffice, en format XML. Aquesta mateixa informació ens servirà per afegir un índex al final del document OpenOffice que volem analitzar.

Definim uns atributs bàsics per representar el mot a indexar:

Atribut	Descripció
id	Identificador de la paraula
mot	Paraula a indexar
font	Nom de la font
tipus	Tipus de la font (Normal, Oblique, Bold, etc..)
mida	Mida en píxels de la font
color	Color de la font
pagina	Indica la pàgina on apareix la paraula

Fig.16 Atributs definits en el fitxer xml.

Per representar aquests atributs en un document XML crearem un element pare de tots els elements que definiran els mots i l'anomenarem de forma arbitària **DescripcióIndex**.

Aquest element tindrà dos fills: un fill que definirà els atributs per representar les paraules (font, tipus, mida i color) i l'altre fill per representar les paraules indexades. Aquest dos fills s'anomenaran **Atributs** i **Paraules**.

Dins d'**Atributs** tindrem declarats (opcionalment) els atributs que defineixen els mots indexats. Aquests elements s'anomenen **font**, **tipus**, **mida** i **color** per definir respectivament el nom de la font, l'estil de la font (normal, italica, bold, etc...), els punts per caràcter de la font i el color de la font. En cas de no estar definits amb un valor concret, disposem d'uns valors per defecte per a cadascun dels atributs.

Dins de **Paraules** tindrem tants fills com paraules volguem definir. Cada paraula es representa per l'element que anomenem **index**. Aquest element tindrà un atribut obligatori que anomenem **id** i que serà un identificador únic de la paraula.

Finalment cada element **index** tindrà com a mínim dos fills: un que indicarà el mot indexat i l'altre que indicarà la pàgina on apareix. Aquests dos atributs queden representats per **mot** i per **pagina**. En tot cas han d'aparèixer tants atributs **pagina** com nombre de vegades aparegui el mot en el document.

Definirem com a obligatoris només l'atribut id per a cada element item, l'element pagina i l'element mot per a cada item definit.

Veiem una possible representació del fitxer XML.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!—Projecte final de Carrera  -->
<DescripcioIndex>
        <Atributs>
            <font>Arial</font>
            <mida>12pt</mida>
            <tipus>Bold</tipus>
            <color>vert</color>
        </Atributs>
        <Paraules>
                <index id="11">
                        <mot>Egipte</mot>
                        <pagina>1</pagina>
                        <pagina>6</pagina>
                </index>
                <index id="20">
                        <mot>Faraó</mot>
                        <pagina>4</pagina>
                 </index>
                <index id="30">
                        <mot>Tutmosis</mot>
                        <pagina>1</pagina>
                        <pagina>6</pagina>
                        <pagina>14</pagina>
                        <pagina>63</pagina>
                </index>
        </Paraules>
</DescripcioIndex>
```

Fig.17 Exemple d'arxiu xml amb atributs generals per a cada mot.

Sempre podem complicar aquesta solució i fer que els atributs siguin diferents per a cadascuna de les paraules. Amb aquesta solució desplaçaríem tota la part d'**Atributs** dins de cada element **index**. D'aquesta forma **DescripcioIndex** només tindria fills del tipus **index**. Cada element **index** tindrà els fills (optatius) per definir els **atributs**, l'element **nom** i l'element/s **pagina**.

El fitxer XML tindrà un o més elements **index**.

Veiem una possible representació d'aquesta simplificació.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!—Projecte final de Carrera  -->
<DescripcioIndex>
   <index id="11">
        <mot>Egipte</mot>
        <font>Arial</font>
```

```
        <mida>12pt</mida>
        <tipus>Bold</tipus>

        <pagina>1</pagina>
        <pagina>6</pagina>
    </index>
    <index id="20">
        <mot>Faraó</mot>
        <font>Arial</font>
        <mida>12pt</mida>
        <tipus>Bold</tipus>
         <color>vert</color>
        <pagina>4</pagina>
    </index>
    <index id="30">
        <mot>Tutmosis</mot>
        <tipus>Bold</tipus>
        <color>vert</color>
        <pagina>1</pagina>
        <pagina>6</pagina>
        <pagina>14</pagina>
        <pagina>63</pagina>
    </index>
</DescripcioIndex>
```

Fig. 18 Exemple d'arxiu xml amb atributs particulars per a tots els mots.

Definim uns valors per defecte pels elements font, tipus, mida i color.

Element	Valor per defecte
font	Arial
tipus	Normal
mida	12pt
color	Negre

Fig. 19 Valors per defecte dels atributs de representació del mot.

En el nostre cas implementarem la primera solució que proporciona definició individualitzada dels atributs de cada mot.

7.2 Definició d'un DTD

Cal definir un DTD per validar la sintaxi del fitxer xml definit anteriorment.
DTD significa Document Type Definition i permet especificar quina és la sintaxi correcte d'un document xml.
Tenim dues formes de definir un DTD:

1- Amb referència externa
   Amb aquesta modalitat indiquem el nom del fitxer que conté la definició. D'aquesta forma podem compartir la mateixa definició per a més d'un fitxer. La sintaxi d'aquesta modalitat s'obté afegint una línia a la capçalera del document del tipus.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE DescripcioIndex SYSTEM "index.dtd">
<DescripcioIndex>
```

Fig. 20 DTD amb referència externa.

On el nom del fitxer amb el dtd és index.dtd.

2- Amb definició dins del document
L'altra forma d'especificar un DTD és incloure directament la sintaxi en el mateix document xml enlloc d'indicar un nom de fitxer dtd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note [
	<!ELEMENT note (to, from, heading, body)>
	<!ELEMENT to (#PCDATA)>
	<!ELEMENT from (#PCDATA)>
	<!ELEMENT heading (#PCDATA)>
	<!ELEMENT body (#PCDATA)>
]>
<note>
	<to>Tove</to>
	<from>Jani</from>
	<heading>Reminder</heading>
	<body>Don't forget me this weekend!</body>
</note>
```

Fig. 21 DTD amb referència en línia del mateix document xml.

Dins del nostre projecte definirem el document DTD de la segona forma (dins del document). Una definició pot ser per exemple:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE DescripcioIndex [
	<!ELEMENT DescripcioIndex (Atributs, Paraules)*>
	<!ELEMENT Atributs ( font?, mida?, tipus?, color?)*>
	<!ATTLIST index id CDATA #REQUIRED>
	<!ELEMENT font (#PCDATA)>
	<!ELEMENT mida (#PCDATA)>
	<!ELEMENT tipus (#PCDATA)>
	<!ELEMENT color (#PCDATA)>
	<!ELEMENT Paraules (mot, pagina+)*>
	<!ELEMENT mot (#PCDATA)>
	<!ELEMENT pagina (#PCDATA)>
	]
```

Fig. 22 DTD pel fitxer de definició d'índexs.

Capítol 8

Anàlisi, disseny i implementació

8.1 Introducció

8.1.1 Què és DOM

DOM és una interface de programació d'aplicacions (API) per a documents HTML i XML; defineix l'estructura lògica dels documents i la manera com accedim i manipulem el document. DOM és una especificació de W3C (http://www.w3c.es) que pot utilitzar-se en qualsevol llenguatge de programació. Podem veure una especificació de DOM independent de cap llenguatge anomenada OMG IDL que veiem a http://www.omg.org/corba/corbaiiop.htm

Veiem un exemple de representació d'un fitxer HTML en un objecte tipus DOM. Si tenim el document HTML

```
<TABLE>
<TBODY>
        <TR>
                <TD>Shady Grove</TD>
                <TD>Aeolian</TD>
        </TR>
        <TR>
                <TD>Over the River, Charlie</TD>
                <TD>Dorian</TD>
        </TR>
</TBODY>
</TABLE>
```

Fig. 23 Exemple DOM. Fitxer HTML

La seva representació en un objecte DOM és:

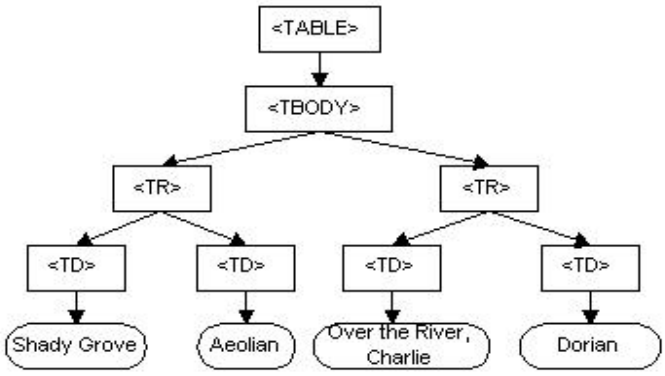

Fig. 24 Exemple DOM. Representació d'un fitxer HTML

8.1.2 Organització bàsica del fitxer content.xml

El document OpenOffice amb que treballarem serà content.xml on tenim bàsicament tot el document. Aquest fitxer té la següent estructura bàsica:
L'element pare és **office:document-content**. Aquest element té com a mínim quatre subelements. Aquests són office:scripts, office:font-face-decls, office:automatic-styles i office:body.

office-scripts no serà utilitzat i per defecte està buit.
office:font-face-decls defineix les fonts que utilitzarem en el document dins dels elements style :font-face.
office:automatic-styles defineix els estils del document dins dels elements style:style.
office:body conté entre d'altres elements el text del document i l'estil associat a cada fragment de text.

Cal tenir en compte algunes consideracions en quant al funcionament d'aquest tipus de document. Així en definir un estil a office:automatic-styles cal associar-li una font. Si aquesta font no existeix a office:font-face-decls cal afegir-la.
Una altra consideració és la nomenclatura dels estils. OpenOffice assigna a cada estil un indentificador únic que sempre serà una lletra i un número incremental començant des d'1 associat a cada lletra. Bàsicament tenim quatre lletres: la P per a paràgraf, la T per a text, la L per a llistes i Sect per a secció. D'aquesta forma els estils de paràgrafs seran P1, P2, P3, etc...
I aquesta referència és la que s'utilitza dins d'office:body per associar-la a un text.

És per això que si en el nostre programa hem d'afegir estils de paràgraf, llista i secció cal que primer analitzem el document d'entrada buscant el darrer número de cadascun d'aquests estils per així poder numerar correctament els nostres estils.

Com havíem comentat a la PAC2, d'entrada podem resoldre aquest exercici de dues formes independents. Una primera és treballant directament els fitxers OpenOffice en format OASIS i l'altra és treballar sobre els arxius OpenOffice amb les API que proporciona el paquet ofimàtic.
La resolució pràctica la farem amb el primer supòsit ja que és el que s'acosta més a les especificacions de l'enunciat que demana treballar amb el format OASIS.

8.2 Resolució

8.2.1 Anàlisi

Es tracta de modificar un fitxer de text creat amb el paquet ofimàtic OpenOffice, en concret amb l'editor Writer. Volem modificar un fitxer amb el format propi d'OpenOffice, o sigui un fitxer amb extensió odt.
La modificació que volem portar a terme és la de crear un índex amb les paraules que apareguin en el document. Les paraules del document que indexarem les guardarem en un fitxer xml que hem definit en l'apartat anterior.

La idea inicial era escriure les paraules a indexar en una llista de les pàgina/es on apareix la paraula en el text. En canvi a causa dels problemes per trobar una forma simple d'esbrinar la pàgina en què ens trobem mentre analitzem el text, de moment el que s'anota és el número de paraula dins del document en que apareix.

He trobat documentació sobre com obtenir el número total de pàgines, he observat l'existència d'estils que inclouen un salt de pàgina però no he trobat cap indicació clara sobre com saber en quina pàgina està una determinada paraula. L'únic plantejament que he trobat consisteix en comptabilitzar el número de pàgines a partir de les especificacions de les mides de la pàgina del document i anar comptant els diferents trossos de text. Amb l'estil de cada fragment de text sabem l'espai que ocupa gràcies als atributs d'estil, així podem saber la pàgina actual però ni tant sols he pogut plantejar-ho per manca de temps i perquè d'entrada no sembla una solució simple (ni tant sols sembla del tot clar que funcioni).

Altres problemes no resolts en el desenvolupament són:
- Cal afinar la cerca de paraules a indexar (Classe StringTokenizer) per evitar certes paraules incorrectes com aquelles que troba per la unió d'un caràcter com el parèntesi o la coma amb una paraula indexable (ex: '(hola' o ',la'). Això podem aconseguir-ho estudiant la classe StreamTokenizer que té el mètode WhitespaceChars que permet definir els rangs de caràcters que volem considerar com paraules a trobar.
- Cal afinar l'ordenació lexicogràfica que fem al guardar l'índex per ordenar els resultats obtinguts. L'ordenació la fem passant les paraules indexades a una llista i utilitzant el mètode Collections.sort, però obtenim una ordenació segons els criteris anglosaxons, d'aquesta forma, per exemple, les lletres amb accent s'ordenen per sota de totes les lletres sense accent. Podem obtenir una ordenació lexicogràfica del nostre alfabet utilitzant la classe collator que permet definir una lexicografia particular.

8.2.2 Disseny i Implementació

Ens trobem davant d'una petita aplicació que desenvoluparem amb el llenguatge de programació orientat a objectes Java. Com a resultat obtindrem un fitxer .class amb l'executable, per tant el programa s'executarà des de la línia de comandes amb l'intèrpret de java.

És una aplicació on no tenim bases de dades i que farà una transformació en un fitxer d'entrada a partir d'un arxiu d'especificacions generant un fitxer de sortida.

S'ha utilitzat el següent software:

Sistema Operatiu: Windows XP Home Edition Version 2002 Service Pack 2
IDE: Jdeveloper 10g Studio Edition Version 10.1.3.0.4
JDK: v1.5.0_05
OpenOffice: v2.0.3

Des del punt de vista del disseny treballarem pensant en objectes i d'aquesta forma definirem un objecte per representar una paraula a indexar. Anomenarem a aquesta classe Index.
Aquesta classe disposarà d'uns atributs per recollir informació posterior i relativa al procés del programa.

8.2.2.1 Classe Index

Els atributs que representen la informació del fitxer d'índex xml són:

Atribut	Descripció
String id	Identificador de la paraula
String mot	Paraula a indexar
String font	Nom de la font per representar la paraula dins de l'índex
String tipus	Tipus de la font per representar la paraula

	dins de l'índex (normal, oblique, bold, etc...)
String mida	Punts de la font
String color	Color de la font

Fig. 25 Atributs de la classe Index.

Els atributs que utilitzarem en el procés del fitxer OpenOffice són:

Atribut	Descripció
String sParagraf	Indica el nom del paràgraf que té a style:style
List<String> pagines	Llista de Pàgines on apareix el mot

Fig. 26 Més atributs de la classe Index.

Aquests atributs que afegim tenen la següent funcionalitat:

sParagraf indica quin valor de style-name té aquesta font dins del document Open-Office a automatic-styles. Aquest valor pot ja existir o haver estat afegit per nosaltres en el procés d'indexació.

Pagines: és una llista de cadenes on anotarem les referències que anem trobant d'aquesta paraula dins del text.

Els mètodes que implementarà aquesta classe són el Set i Get de les propietats més alguns mètodes per afegir i obtenir les referències d'aquesta paraula.

8.2.2.2 Classe IndexReader

Ja hem vist la classe Index que defineix un mot a indexar. Per a la nostra aplicació necessitem treballar amb una o més paraules a indexar. Per aquesta raó creem una classe anomenada IndexReader que utilitzarà la classe Index per elaborar una estructura de dades que reculli totes les paraules a indexar i els seus atributs.

Aquesta classe té diferents atributs per representar el conjunt de paraules a indexar i d'altres atributs per representar la informació del fitxer xml on definim les característiques d'un mot indexat:

Atribut	Descripció
String sRuta	Ruta del fitxer amb els mots a indexar
Int iTotalIndexs	Indica quantes paraules tenim per indexar en el fitxer de sRuta
Int iIndexs	Indica quantes paraules tenim per indexar després de llegir el fitxer (elimina duplicats)
Map<String,Index> mIndex	Mapa de Java que té tota la informació de les paraules a indexar.
String font	Nom de la font per representar la paraula dins de l'índex
String tipus	Tipus de la font per representar la paraula dins de l'índex (normal, oblique, bold, etc...)
String mida	Punts de la font
String color	Color de la font

boolean isFontValida	Indica si la font és valida. Les fonts vàlides són: Arial, Times New Roman i Comic Sans MS

Fig. 27 Atributs de la classe IndexReader.

Els mapes de Java són array bidimensionals. El primer component és la clau del registre i serveix per identificar l'element unívocament i el segon element són les dades associades a aquesta clau.

En el nostre cas utilitzarem com a clau una variable String que és la paraula a indexar mentre que, com a dades emprarem un objecte de tipus Index que representa tots els atributs d'aquesta paraula (com hem vist a l'apartat anterior).

Aquesta classe implementarà diferents mètodes. El mètode més important és el que llegeix el fitxer d'índexos amb les paraules a indexar e inicialitza l'objecte IndexReader amb els valors llegits.
Altres mètodes permeten modificar i accedir a diferents Atributs de la clau index a través de la paraula a indexar.

8.2.2.3 Classe IndexarParaules

Per altra banda ens cal una estructura de dades per manegar la informació del document OpenOffice.

El document de text creat amb OpenOffice està en format OASIS, basat en XML i es tracta, bàsicament, d'una sèrie de documents XML junts en un fitxer en format JAR i per tant és amb aquest format que haurem de treballar per manipular el document. Afortunadament podem aconseguir amb el llenguatge de programació Java diferents classes per treballar amb aquests formats estàndards.
Gràcies a aquesta facilitat que ens proporciona Java no crearem més classes, com ara una que representi el document OASIS a analitzar o una classe que representi un fitxer JAR..

Així per treballar amb fitxers JAR disposem de les classes:

java.util.jar.JarEntry;
java.util.jar.JarInputStream;
java.util.jar.JarOutputStream;
java.util.jar.Manifest;

Per treballar amb documents XML emprarem la classe Document que utilitza l' especificació DOM (Document Object Model) per representar en forma d'arbre un document en format XML. Aquesta classe ens facilitarà molt la feina ja que permet recórrer fàcilment el document a través dels seus elements.

Les classes que utilitzem per representar i accedir al document en format DOM són:

org.w3c.dom.Document;
org.w3c.dom.Element;
org.w3c.dom.NamedNodeMap;
org.w3c.dom.Node;
org.w3c.dom.NodeList;
org.w3c.dom.Text;

La classe IndexarParaules conté el programa principal de l'aplicació. Aquesta classe té diferents atributs que permeten realitzar la tasca encomanda. Aquests atributs són:

Atribut	Descripció
IndexReader indexReader	Representació de les paraules a indexar
Document document	Representació en forma d'arbre del document OpenOffice
Element documentRoot;	Punter a l'arrel del fitxer OpenOffice
NodeList headingElements	Llista de Nodes de des l'arrel del document OpenOffice
int lastParaNumber	Darrer número d'estil de paràgraf
int lastListNumber	Darrer número d'estil de llista
int lastSectionNumber	Darrer número de secció
Node lastParaNode	Apuntador al darrer node de paràgraf
Node lastListNode	Apuntador al darrer node de llista
Node lastSectionNode	Apuntador al darrer node de secció
File inFile, OutFile	Descriptors dels fitxers d'entrada i sortida
JarInputStream inJar	Canal d'entrada per llegir un fitxer JAR
JarOutputStream outJar	Canal de sortida per llegir un fitxer JAR

Fig. 28 Atributs de la classe IndexarParaules

Per tant tenim aquest diagrama de classe en la nostra aplicació.

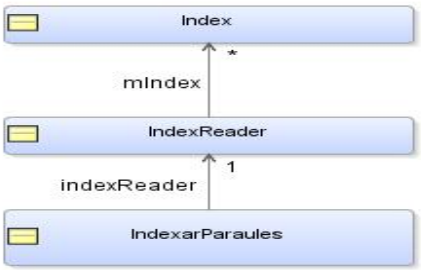

Fig. 29 Diagrama de classes

8.2.2.4 Estructura del programa

El programa bàsicament realitza les següents accions:

Analitzar el document d'entrada en format OpenOffice i generar el fitxer de sortida amb l'índex associat amb el mètode indexParaules.IndexarDocument().

Crear el fitxer amb les paraules indexades creant l'objecte IndexReader amb el mètode indexReader.guardarIndex().

En concret veiem quins passos fa el programa per genenar l'índex.

1- Copiem tots els fitxers tipus JAR que hi ha al fitxer OpenOffice cap al fitxer de sortida a excepció del fitxer content.xml on tenim la informació a tractar.

2- Obtenim el fitxer content.xml en un objecte en format DOM. Tenim el document en format d'arbre.
3- Mirem si les fonts que hem definit per les paraules a indexar existeixen a office:font-face-decls i sinó existeixen les afegim. Només permetem afegir les fonts Arial, Times New Roman i Comic Sans MS.
4- Busquem dins dels estils (office:automatic-styles) quin és el darrer número de paràgraf, llista i secció. També busquem els apuntadors al darrer element de paràgraf, llista i secció per saber on tenim que afegir-los.
5- Afegim els estils de paràgraf per a cadascuna de les paraules.
6- Afegim un estil per defecte de salt de línia per poder escriure a la darrera pàgina l'índex.
7- Afegim un estil de secció que inclourà l'índex.
8- Afegim uns estils de tipus llista.
9- Analitzem el text afegint les referències a les paraules quan les trobem en el text.
10- Afegim al text OpenOffice una llista amb les paraules indexades que s'han anat guardant a l'objecte del tipus IndexReader.
11- Guardem el fitxer JAR content.xml en el fitxer de sortida actualitzant la data de creació del fitxer JAR.
12- Guardem les paraules indexades juntament amb els atributs de representació en el document OpenOffice i amb el conjunt de les referències que s'han trobat en el text en un fitxer en format XML.

8.3 Deploy de l'aplicació

L'entorn de desenvolupament d'aquest projecte ha estat l'eina gratuïta d'Oracle anomenada Oracle Jdeveloper versió 10g. Aquest IDE de programació Java permet, entre moltes altres opcions, desenvolupar aplicacions java de consola (com és aquest cas) i permet realitzar l'execució sense tenir que executar l'intèrpret de Java des de la línia de comandes.
En l'entrega d'aquesta memòria donarem tots els fitxers que necessita Jdeveloper per compilar aquest projecte però també entregarem un fitxer Jar amb el deploy de l'aplicació.

Expliquem breument com obtenir i executar aquest deploy amb Jdevelopper. En primer lloc cal crear un arxiu de Deploy amb JDeveloper. Ho fem amb New → General → Deployment Profiles → JAR Files
Indiquem un nom.

Fig. 30 Crear un fitxer de deploy a JDdevelopper.

Configurem les propietats indicant quina és la classe principal.

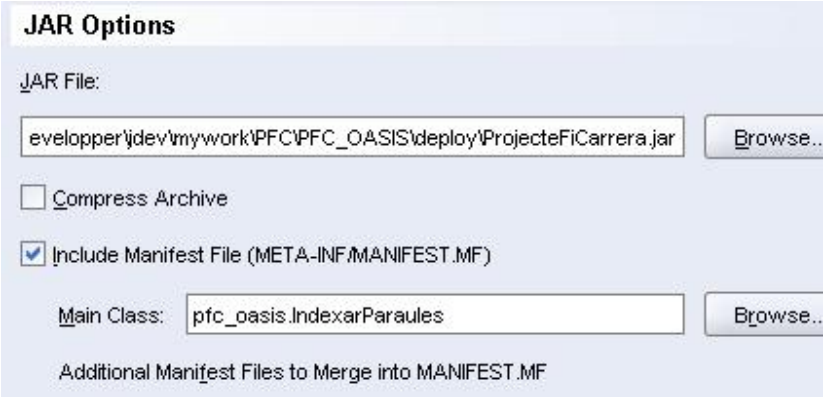

Fig. 31 Definir la classe principal de l'aplicació.

També indicarem quins fitxers .class volem exportar.

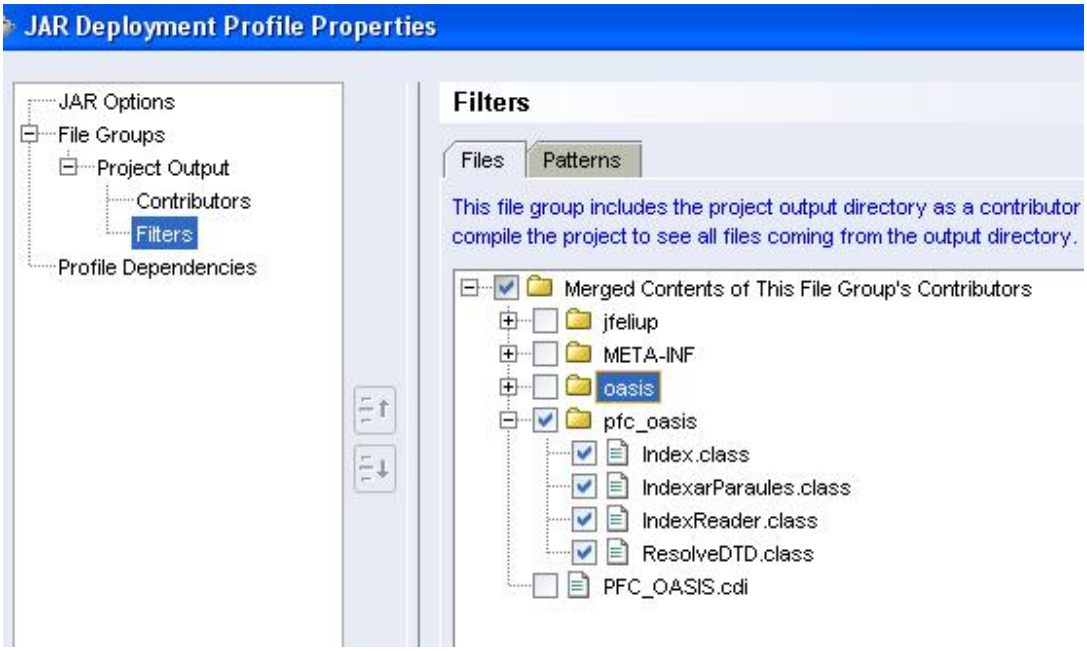

Fig. 32 Definir les classes a incloure en el fitxer JAR.

Finalment des del navegador d'aplicacions seleccionem el deploy profile que hem creat i amb el botó dret tenim l'opció de generar un fitxer JAR.

Fig. 33 Deploy de l'aplicació en un fitxer JAR.

Un cop tenim el fitxer JAR el trobem a la ruta relativa a Jdevelopper\jdev\mywork nomAplicacio\nomProjecte\deploy\deployProfileName.jar
Un cop tenim el fitxer jar només cal configurar la nostra estació de treball per poder-lo executar. Cal fer dos passos previs.

1- Configurar la variable PATH de windows.
   Anem a MiPC → Botó dret → Opciones Avanzadas → Variables de entorno
   A variables de sistema tenim la variable PATH on afegirem la ruta d'on tenim en el nostre ordinador la màquina virtual de java. En el meu cas és:

   C:\Archivos de programa\Java\jdk1.5.0_06\bin

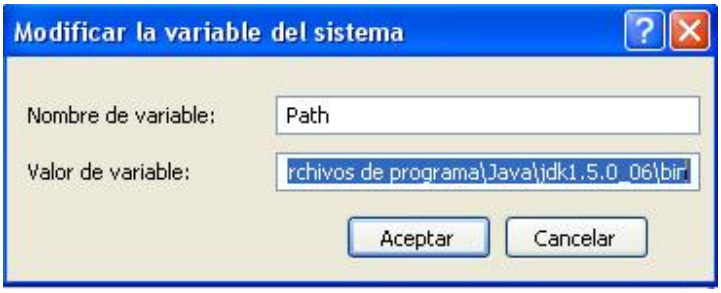

Fig. 34 Definir la variable d'entorn PATH.

2- Configurar la variable d'entorn de Java anomenada JAVA_HOME
   Anem a MiPC → Botó dret → Opciones Avanzadas → Variables de entorno
   A variables de sistema creem la variable JAVA_HOME per indicar la ruta de les classes de java. En el meu cas és:

   C:\Archivos de programa\Java\jdk1.5.0_06

Fig. 35 Definir la variable d'entorn JAVA_HOME.

En aquest cas el segon pas no serà necessari doncs el fitxer jar ja tindrà totes les classes necessàries. Aquesta variable és útil a l'hora de compilar amb javac per indicar la ruta de les llibreries. En el nostre cas aquesta feina la fem des de l'entorn Jdeveloper.

Finalment executem el programa. Suposem que el fitxer jar s'anomena Indexar. Cal obrir un intèrpret de comandes i teclejar

C:> java –jar Indexar.jar [Parametre1] [parametre2] [parametre3] –f{NomFont} –t{Tipus} –m{Mida} –c{Color}

Veiem un exemple de l'execució.

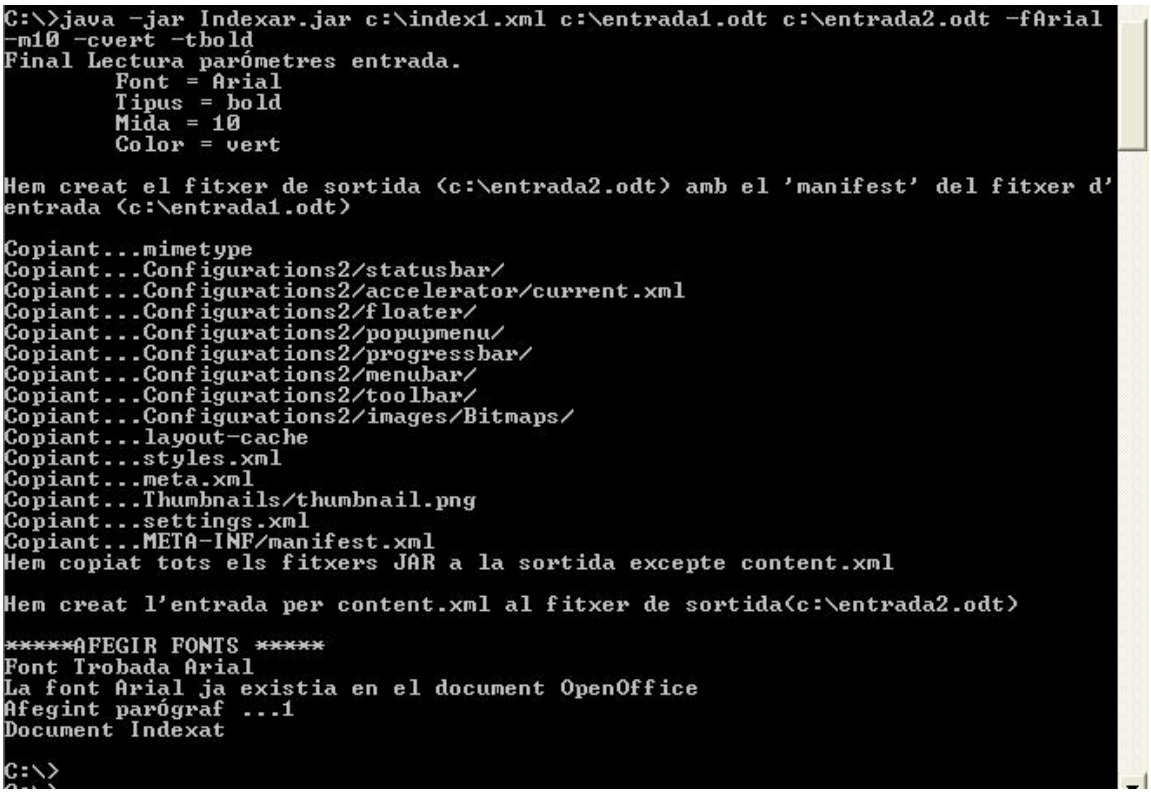

Fig. 36 Exemple d'execució del fitxer JAR.

Aquest programa es diu IndexarParaules, així per executar el programa des de la línia de comandes farem

C :>Java IndexarParaules [parametre1] [parametre2] [parametre3] –f{NomFont} –t{Tipus} –m{Mida} –c{Color}

8.4 Jocs de Proves

El programa espera com a paràmetres d'entrada un mínim de tres valors i un màxim de set valors. Els valors obligatoris són els tres primers. Aquest són:
El primer és la ruta del fitxer amb les paraules a indexar, el segon és la ruta del fitxer OpenOffice amb el document a indexar i un tercer amb la ruta del fitxer que generarem amb l'índex.
Els valors optatius són els que van de la posició quatre a la set i serveixen per indicar els atributs del nou índex que crearem. Aquest són:
-f{NomFont}: Indica el nom de la font.
-t{Tipus}: Indica el tipus de la font.
-m{Mida}: Indica la mida en punts de la font.
-c{Color}: Indica el color de la font.

La primera prova que fem és passar més paràmetres dels demanats. La sortida del JDeveloper és :

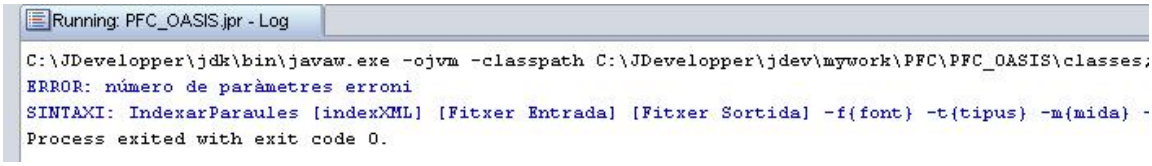

Fig. 37 Sortida del programa. Número de paràmetres erroni.

Ara per exemple passem tres paràmetres però el fitxer de sortida el tenim obert amb el OpenOffice. La sortida del JDeveloper és:

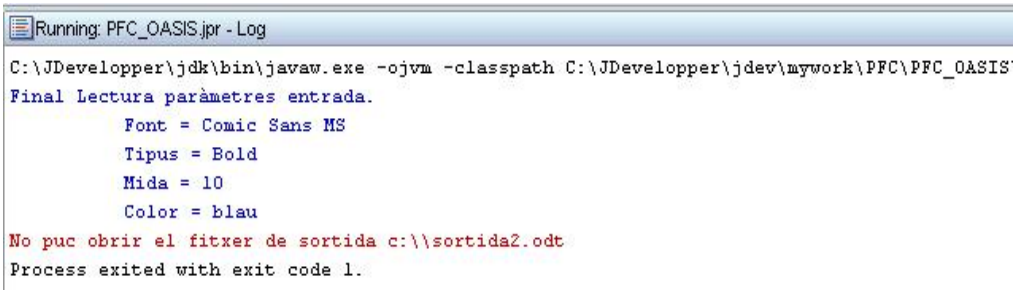

Fig. 38 Sortida del programa. No podem obrir el fitxer de sortida.

Veiem ara la sortida en el cas d'una execució normal. En aquest cas utilitzem el fitxer d'índex index.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<DescripcioIndex PUBLIC "-//w3c//DTD Reader//ES"
                        "http://IndexarParaules.dtd">
    <DescripcioIndex>
        <Atributs>
            <font>Arial</font>
            <mida>12pt</mida>
            <tipus>Bold</tipus>
            <color3>vert</color>
        </Atributs>
        <Paraules>
            <index id="11">
                <mot>Egipte</mot>
                <pagina>1</pagina>
                <pagina>6</pagina>
            </index>
            <index id="20">
                <mot>Faraó</mot>
                <pagina>4</pagina>
             </index>
            <index id="30">
                <mot>Tutmosis</mot>
                <pagina>1</pagina>
                <pagina>6</pagina>
                <pagina>14</pagina>
                <pagina>63</pagina>
            </index>
        </Paraules>
    </DescripcioIndex>
```

Fig. 39 Fitxer de paraules indexades.

En aquest exemple veiem com hi ha elements no reconeguts com color3, quan l'element reconegut és color. En aquests casos el programa utilitza el valor per defecte de color que és 'negre'.

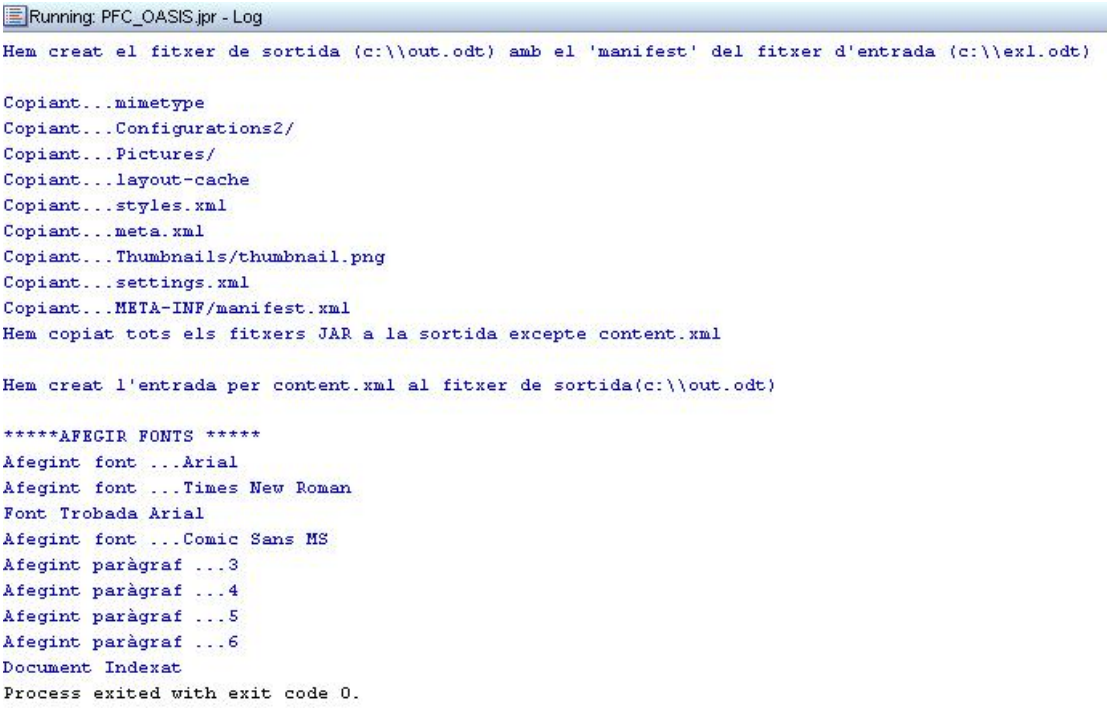

Fig. 40 Sortida del programa. Execució correcte.

Veiem diferents missatges de log que van informant sobre el procés.
El document OpenOffice generat té un apartat d'índex a la darrera pàgina.

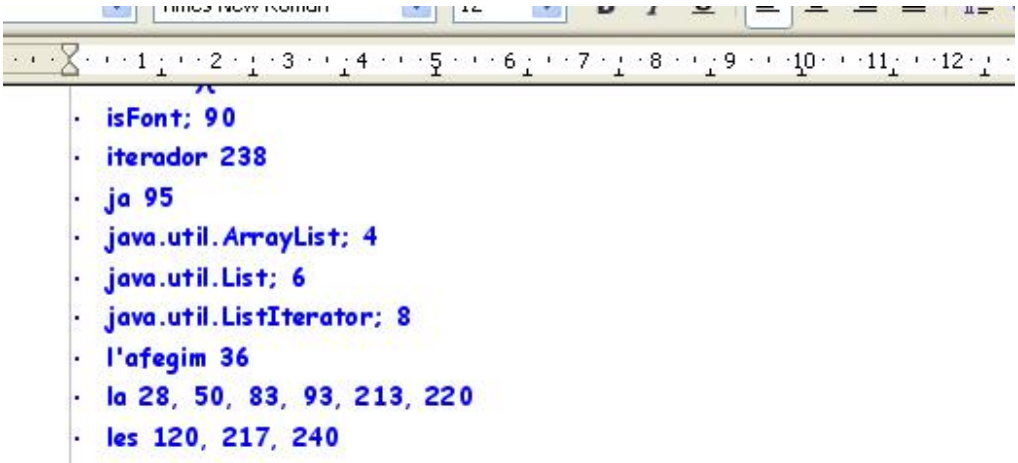

Fig. 41 Detall de l'índex en el documentOpenOffice creat.

Capítol 9

Conclusions

Un cop finalitzat el treball d'aquest projecte podem extreure'n algunes conclusions:

- *OpenOffice, un paquet ofimàtic de sofware lliure amb fitxers de format obert:* Dins d'un mercat informàtic dominat per Microsoft s'ha utilitzat en aquest projecte un paquet ofimàtic de tanta qualitat com Microsoft Office però gratuït. No només això sinó que el format dels fitxers que genera també és obert i no propietari com passa amb els fitxers creats amb Microsoft Office. Aquesta darrera característica és molt interessant doncs evita la dependència dels fitxers respecte un determinat programari.
- *Independència d'un programari concret:* Com dèiem en el punt anterior, aquest format obert permet a qualsevol empresa desenvolupar un programa per tractar amb els fitxers d'aquest format. No només això sinó que el mateix paquet ofimàtic ofereix un SDK per facilitar aquesta tasca. Aquest obre el mercat a empreses que poden crear productes a mida.
- *Java i XML, estàndards del mercat:* Per tal de realitzar el treball pràctic hem utilitzat el llenguatge de programació Java, un IDE de Java anomenat Jdevelopper, tots ells igualment gratuïts. Per tant s'ha pogut realitzar tot el projecte amb programari totalment gratuït i d'alta qualitat.

  També hem utilitzat XML que en aquests darrers anys s'ha convertit en un estàndard per emmagatzemar dades. XML té un format textual i ha substituit els tradicionals fitxers de dades i/o configuració binaris gràcies a l'augment de la capactitat de processament dels computadors en els darrers anys. XML, entre d'altres avantatges, crea fitxers molt interpretables directament (utilitzant noms adients als continguts) i visualment (estructura d'arbre).
- *Flexibilitat de la programació amb Java:* En el treball pràctic hem pogut veure alguns dels beneficis del programari lliure i dels formats oberts. Per exemple a l'hora de programar aquesta eina de transformació en Java hem vist la gran quantitat de llibreries de lliure accés per realitzar tot tipus de tasques com manipular fitxers en format JAR, XML, etc...

  La tasca del programador es focalitza en molts casos en cercar la llibreria que fa la feina que volem, més que en desenvolupar la seva pròpia llibreria com passava fa anys.

Glossari de termes

XML: eXtended Markup Language és un llenguatge de marques per intercanvi d'informació entre ordinadors.

OASIS: conjunt d'especificacions del format dels documents OpenOffice.

SDK: Software Development Tool identifica el conjunt d'eines necessàries per fer desenvolupaments sobre un producte software.

ODT, ODS, ODP: extensions dels documents OpenOffice segons els tipus d'aplicació. ODT és l'extensió dels documents fets amb el programa OpenOfficeWriter (editor de textos), ODS és l'extensió dels documents fets amb l'OpenOfficeCalc (full de càlcul) i ODP és l'extensió dels documents fets amb l'OpenOfficeImpress (presentacions).

JAR: Java ARrchives identifica uns fitxers de tipus ZIP que contenen arxius compilats Java (en format class) llestos per ser utilitzats en els nostres programes.

UNO:Universal Network Object. Són els objectes que proporcionen accés a la API d'OpenOffice.

API: Application Programming Interface són el conjunt de funcions i variables que pot proporcionar un producte software per tal de programar aplicacions de tercers que interactuïn amb el programari.

DTD: Document Type Definiton permet definir els elements correctes i l'estructura d'un document XML mitjançant unes simples regles sintàctiques.

DOM : Document Object Model és una interface de programació d'aplicacions (API) per documents HTML i XML. Defineix l'estructura lògica dels documents i la forma d'accedir i manipular els documents.

Bibliografia i referències

- **Programari OpenOffice**: http://www.openoffice.org
- **Especificació XML**: http://www.w3.org/TR/REC-xml
- **Programari lliure en català**: http://www.softcatala.com
- **OASIS Standards**: http://www.oasis-open.org/specs/index.php
  http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office
- **Entorn IDE d'Oracle per Java** http://www.oracle.com/technology/products/jdev
- **SDK de Java** http://www.sun.com
- **Wikipedia** http://es.wikipedia.org
- **SDK OpenOffice** http://development.openoffice.org
- **API OpenOffice** http://api.openoffice.org
- **Aplicacions portables** http://portableapps.com
- **Software lliure** http://www.gnu.org/philosophy/free-sw.es.html
- **GNU GPL** http://www.gnu.org/licenses/licenses.es.html
- **Documentació sobre JAR** http://java.sun.com/j2se/1.3/docs/guide/jar/jar.html
- **Tutorial sobre Jar** http://www.programacion.com/java/tutorial/jar

Apèndix. Arxius adjunts

Juntament amb la memòria de la pràctica i la presentació també adjunto un arxiu anomenat PFC_jfeliup_memòria_ArxiusAdjunts.zip. En aquest document s'inclouen diferents informacions:

1- La carpeta PFC_OASIS (PFC_OASIS.zip) inclou els arxius generats en la realització de l'exercici amb l'entorn JDevelopper. Per tal de poder-los provar en una altra màquina cal copiar aquesta carpeta en la següent ruta:

[Ruta JDeveloper] / jdev / mywork /

2- Arxiu Indexar.jar amb l'aplicació. Per executar-la cal escriure des de la línia de comandes.

c> java –jar Indexar.jar [Parametre1] [parametre2] [parametre3] "-f{NomFont}" -t{tipus} –m{mida} –c{color}

3- La carpeta JavaDoc amb els javadocs que hem creat a partir dels fitxers fonts i la utilitat javadoc (JavaDoc.zip).

4- Fitxers d'exemple entregats extrets del conjunt de joc de proves (Joc de Proves.zip):

Prova1 :
Index1.xml → Fitxer amb les paraules indexades.
Entrada1.odt → Fitxer d'entrada 1.
Sortida1.odt –> Fitxer de sortida 1.

Prova2 :
Index2.xml → Fitxer amb les paraules indexades.
Entrada2.odt → Fitxer d'entrada 2.
Sortida2.odt –> Fitxer de sortida 2.

5- El fitxer "IndexarParaules.dtd" per validar els fitxers XML.

6- El fitxer Fonts_Java.zip amb els fitxers fonts de java que també estan dins del fitxer PFC_OASIS.zip.

Apèndix. Codi

En aquest apartat adjuntem el codi de la pràctica. En concret dedicarem un apartat a cadascuna de les classes de l'exercici.

Classe Index

```
package pfc_oasis;

import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

/*****************************************************************************
 * Classe que representa una paraula a indexar en un document OpenOffice
 * Definim els atributs que tindrà la paraula dins del document OpenOffice quan
 * l'afegim al document
 * ***************************************************************************/
public class Index {
    private String id;  //Identificador de la paraula
    private String mot; //Paraula a indexar
    private String sParagraf; //Indica quin paràgraf té a style:style
    private List<String> pagines = new ArrayList(); //LLista de Pàgines on apareix el mot
            //Llista d'aparicions de la paraula dins del document.
    //private boolean isFont; //Indicarà si la font ja està en el document OpenOffice

    //Constructor. Definim els valors per defecte
    public Index(){
        //isFont = false;
        //isFontValida = false;
    }

    //getters i setters a les propietats.
    public void setId(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public void setMot(String mot) {
        this.mot = mot;
    }

    public String getMot() {
        return mot;
    }

    public void setPagines(List<String> pagines) {
        this.pagines = pagines;
    }
```

```
    public List<String> getPagines() {
        return pagines;
    }

    //Assigna el nom style-name de paràgraf que té dins del document aquesta paraula
    public void setSParagraf(String sParagraf) {
        this.sParagraf = sParagraf;
    }

    //Retorna el nom style-name de paràgraf que té dins del document aquesta paraula
    public String getSParagraf() {
        return sParagraf;
    }

    //afegeix la referència 'iPosicio' a les referències de la pàgina
    public void afegirParaulaTrobada (long iPosicio){
        String sPosicio = "";

        sPosicio = sPosicio.valueOf(iPosicio);
        pagines.add(sPosicio);
    }

    //retorna un iterador de les referències trobades
    public ListIterator<String> getReferencies(){
        return pagines.listIterator();
    }

    //retorna quantes refèrencies té aquesta paraula
    public int getTotalParaules (){
        return pagines.size();
    }
}
```

Classe IndexReader

```
package pfc_oasis;

import com.sun.org.apache.xml.internal.serialize.OutputFormat;
import com.sun.org.apache.xml.internal.serialize.XMLSerializer;

import com.sun.org.apache.xml.internal.serializer.WriterToUTF8;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;

import java.util.ListIterator;
import java.util.Map;

import java.util.Set;

import java.util.Vector;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.DOMImplementation;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

import org.w3c.dom.Text;

/*****************************************************************************
 * Classe que defineix l'estructura de dades per indexar les paraules dins del
 * document OpenOffice. Utilitza com a base la classe Index.
 * Creem diferents atributs
 *      - sRtuta Guardem la ruta del fitxer de paraules a indexar
 *      - iTotalIndexs indica quantes paraules tenim per indexar
 *      - iIndexs indica quantes paraules hem indexat dins la classe
 *      - Map <String,Index> Tenim un mapa amb la paraula clau a indexar i com a
 *            valor la llista anterior amb els atributs de l'objecte tipus Index
 * ***************************************************************************/
public class IndexReader{

    private String sRuta;
    private long iTotalIndexs;
    private long iIndexs;
    private Map<String,Index> mIndex;
```

```
    private Index indexTmp;
    private String font;    //nom de la font per representar la paraula dins de l'índex
    private String tipus;    //tipus de la font per representar la paraula dins de l'índex (normal,
oblique, bold, etc...)
    private String mida;    //punts de la font
    private String color;   //color de la font
    private boolean isFontValida; //Indica si la font és valida. Aquestes són:
            //Arial, Times New Roman i Comic Sans MS
    //private boolean fontValida;
    private String paragraf;
    private Long paraulesIndexades;

    //Constructor de la classe. Inicialitzem els valors per defecte.
    //Busca els atributs de les paraules a indexar a la línia de comandes.
    public IndexReader(String argv[]) {//sRutaXML) {
        sRuta = argv[0]; //sRutaXML;
        isFontValida = false;
        paraulesIndexades = new Long(0);
        mIndex = mIndex = new HashMap();
        //Lectura dels atributs del fitxer d'índexs si s'han passat com a paràmetre
        int i = argv.length - 1;
        //Valors per defecte
        font = "Arial";
        tipus = "normal";
        mida ="12";
        color = "negre";
        String sTmp, sTipus;
        while (i>2){    //Els paràmetres dels atributs estan en les posicions 3 a 6
            sTmp = argv[i].substring(0,2);
            if (sTmp.compareToIgnoreCase("-f")==0){
                sTmp = argv[i].substring(2,argv[i].length());
                font = sTmp;
            }
            else if (sTmp.compareToIgnoreCase("-t")==0){
                sTmp = argv[i].substring(2,argv[i].length());
                tipus = sTmp;
            }
            else if (sTmp.compareToIgnoreCase("-m")==0){
                sTmp = argv[i].substring(2,argv[i].length());
                mida = sTmp;
            }
            else if (sTmp.compareToIgnoreCase("-c")==0){
                sTmp = argv[i].substring(2,argv[i].length());
                color = sTmp;
            }
            i--;
        }
    }

    //Imprimeix per la sortida estàndard els atributs de les paraules a indexar
    public void imprimirAtributs (){
        System.out.println("\tFont = " + font);
        System.out.println("\tTipus = " + tipus);
        System.out.println("\tMida = " + mida);
        System.out.println("\tColor = " + color);
    }
```

```
//Indica que la font de l'índex és vàlida
public void SetFontValida (){
   isFontValida = true;
}

//Indica si la font de l'atribut 'font' és una font considerada vàlida
public boolean isFontValida(){
   return isFontValida;
}

//Guardar la taula d'índex dels mots trobats en un fitxer XML segons
//un format definit per nosaltres
public void guardarIndex(){
   Document dom;

   //obtenim una instància del document builder factory
   DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
   try {
      //Obtenim una instància del document builder
      DocumentBuilder db = dbf.newDocumentBuilder();

      //creem una instància de DOM
      dom = db.newDocument();
      //Creem el pare
      Element rootEle = dom.createElement("DescripcioIndex");
      dom.appendChild(rootEle);

      //Creem el primer fill amb els atributs de l'índex
      Element atributEle = dom.createElement("Atributs");
      //Font
      Element fontEle = dom.createElement("font");
      Text fontText = dom.createTextNode(this.font);
      //Mida
      Element midaEle = dom.createElement("mida");
      Text midaText = dom.createTextNode(this.mida);
      //Tipus
      Element tipusEle = dom.createElement("tipus");
      Text tipusText = dom.createTextNode(this.tipus);
      //Color
      Element colorEle = dom.createElement("color");
      Text colorText = dom.createTextNode(this.color);

      fontEle.appendChild(fontText);
      midaEle.appendChild(midaText);
      tipusEle.appendChild(tipusText);
      colorEle.appendChild(colorText);
      atributEle.appendChild(fontEle);
      atributEle.appendChild(midaEle);
      atributEle.appendChild(tipusEle);
      atributEle.appendChild(colorEle);

      //Creem el segon fill amb els mots indexats
      Element paraulaEle = dom.createElement("Paraules");

      Set <String> sParaulesIndex = this.ObtenirParaulesIndex();
```

```
            Vector v = new Vector (sParaulesIndex);
            Collections.sort(v, new String ().CASE_INSENSITIVE_ORDER);
            Iterator <String> it = v.iterator();
            while (it.hasNext()){
               String clau = it.next();
               Element indexEle = dom.createElement("index");
               indexEle.setAttribute("id", this.mIndex.get(clau).getId());

               Element motEle = dom.createElement("mot");
               Text motText = dom.createTextNode(clau);
               //Escrivim el mot
               motEle.appendChild(motText);
               indexEle.appendChild(motEle);
               //Escrivim les referències
                Index inx = mIndex.get(clau);
                List<String> pagina= inx.getPagines();
                for (String pag: pagina){
                   Element paginaEle = dom.createElement("pagina");
                   Text paginaText = dom.createTextNode(pag);
                   paginaEle.appendChild(paginaText);
                   indexEle.appendChild(paginaEle);
                }
               paraulaEle.appendChild(indexEle);
            }
            rootEle.appendChild(atributEle);
            rootEle.appendChild(paraulaEle);

            OutputFormat format = new OutputFormat(dom);
            format.setIndenting(true);
            //format.DTD.XHTMLPublicId("Hola.DTD");
            format.setEncoding("Windows-1252");
            format.setDoctype("-//w3c//DTD Reader//ES","http://IndexarParaules.dtd");
            XMLSerializer serializer = new XMLSerializer(
                  new FileOutputStream(new File(sRuta)), format);
            serializer.serialize(dom);

         }catch(ParserConfigurationException pce) {
               System.out.println("Error en crear el document XML amb els índex en instanciar
DocumentBuilder " + pce);
               System.exit(1);
         }
         catch(IOException ie) {
            ie.printStackTrace();
         }
      }

      // Mètode de DEBUG per imprimir el contingut del mapa d'índexos
      public void ImprimirMapaIndexos (){
         Set<String> ids = mIndex.keySet();
         System.out.println("Long Index " + mIndex.size());
         for (String id: ids) {
            Index inx = mIndex.get(id);
            System.out.println("id-" + inx.getId() + "\tmot "+ id);
            //System.out.println("FOntOO? " + inx.isIsFontValida() + " *** " + "Font? " +
inx.isIsFont() + "***" + inx.getColor()+ " *** "+inx.getFont()+" *** "+inx.getTipus()+" ***
"+inx.getMida()+" *** "+inx.getTotalParaules()+" *** "+inx.getSParagraf());
```

```
        List<String> pagina= inx.getPagines();
        for (String pag: pagina){
            System.out.print(pag + " - ");
        }
        System.out.println("");
    }
}

//Setters i Getters als atributs de la classe
public String getSRuta() {
    return sRuta;
}

public long getITotalIndexs() {
    return iTotalIndexs;
}

public long getIIndexs() {
    return iIndexs;
}

public Index getIndexTmp() {
    return indexTmp;
}

//Obtenim un conjunt de les paraules a indexar
public Set ObtenirParaulesIndex(){
    return this.mIndex.keySet();
}

//Obtenim el nom de la font de la paraula a indexar 'clauIndex'
public String getFont(String clauIndex){
    //String sFont = this.mIndex.get(clauIndex).getFont();
    String sFont = this.font;
    if (sFont==null) sFont="";
    return sFont;
}

//Obtenim el color de la font de la paraula a indexar 'clauIndex'
public String getColor(String clauIndex){
    //String sColor = this.mIndex.get(clauIndex).getColor();
    String sColor = this.color;
    if (sColor==null) sColor="";
    return sColor;
}

//Obtenim l'estil de la font de la paraula a indexar 'clauIndex' (normal, italic, bold)
public String getPuntsTipus(String clauIndex){
    //String sTipus = this.mIndex.get(clauIndex).getTipus();
    String sTipus = this.tipus;
    if (sTipus.length()==0) sTipus="normal";
    return sTipus;
}

//Obtenim el número de punts de la font de la paraula a indexar 'clauIndex'
public String getPuntsFonts(String clauIndex){
```

```
        //String sPuntsFont = this.mIndex.get(clauIndex).getMida();
        String sPuntsFont = this.mida;
        if (sPuntsFont==null) sPuntsFont="12pt";
        return sPuntsFont;
    }

    //Afegeix una referència a la paraula a indexar 'clauIndex' dins la llista de referències
    public void ParaulaTrobada(String clauIndex, long iContadorParaules){
        Index indexTmp;

        indexTmp = this.mIndex.get(clauIndex);
        indexTmp.afegirParaulaTrobada(iContadorParaules);
        this.mIndex.put(clauIndex, indexTmp);
    }

    //Afegeix un nou mot a l'índex.
    public void AfegirParaulaIndex (String clauIndex, long iContadorParaules){
        paraulesIndexades++;

        indexTmp = new Index();
        indexTmp.setId(paraulesIndexades.toString());
        indexTmp.setMot(clauIndex);
        indexTmp.afegirParaulaTrobada(iContadorParaules);
        this.mIndex.put(clauIndex,indexTmp);
    }

    //Obtenim un iterador de tipus String de les referències de la paraula a indexar 'clauIndex'
    public ListIterator<String> ObtenirReferencies (String clauIndex){
        return this.mIndex.get(clauIndex).getReferencies();
    }

    //Obtenim el nom d'estil de la paraula a indexar 'clauIndex' creada al document OpenOffice
    public String ObtenirParagraf(String clauIndex){
        return this.mIndex.get(clauIndex).getSParagraf();
    }

    //Guardem el nom d'estil de la paraula a indexar 'clauIndex' creada al document
//OpenOffice
    public void GuardarParagraf(String clauIndex, String sParagraf){
        Index indexTmp;

        indexTmp = this.mIndex.get(clauIndex);
        indexTmp.setSParagraf(sParagraf);
        this.mIndex.put(clauIndex, indexTmp);
    }

    public String get_font() {
        return font;
    }

    public void setFont(String font) {
        this.font = font;
    }

    public String get_color() {
        return color;
```

```
    }

    public String getTipus() {
        return tipus;
    }

    public void setParagraf(String paragraf) {
        this.paragraf = paragraf;
    }

    public String getParagraf() {
        return paragraf;
    }

    public void setITotalIndexs(long iTotalIndexs) {
        this.iTotalIndexs = iTotalIndexs;
    }

    //Indiquem que la font de la paraula a indexar 'clauIndex' és una de les fonts vàlides
    //Permetem les fonts Arial, Times New Roman i Comic Sans MS
    /*public void NovaFontOpenOffice(String clauIndex){
        Index indexTmp;

        indexTmp = this.mIndex.get(clauIndex);
        indexTmp.setIsFontValida(true);
        this.mIndex.put(clauIndex, indexTmp);
    }*/

    //Indica que la font de la paraula a indexar 'clauIndex' existeix al document OpenOffice
    /*public void FontExisteix (String clauIndex){
        Index indexTmp;

        indexTmp = this.mIndex.get(clauIndex);
        indexTmp.setIsFont(true);
        this.mIndex.put(clauIndex, indexTmp);
    }*/

    //Mira si la paraula a indexar 'clauIndex' és una font vàlida
    //Permetem les fonts Arial, Times New Roman i Comic Sans MS
    /*public boolean FontAfegida (String clauIndex){
        boolean bFontAfegida = false;
        Index indexTmp;

        indexTmp = this.mIndex.get(clauIndex);
         if ((indexTmp.isIsFontValida()==true))
            bFontAfegida = true;
        return (bFontAfegida);
    }*/

    public String getMida() {
        return mida;
    }
}
```

Classe ResolveDTD

```
package pfc_oasis;

import java.io.StringReader;

import org.xml.sax.EntityResolver;
import org.xml.sax.InputSource;


public class ResolveDTD implements EntityResolver{

public InputSource resolveEntity (String publicID, String systemID){
    if (systemID.endsWith(".dtd")){
        StringReader stringInput = new StringReader("");
        return new InputSource (stringInput);
    }
    else {
        return null;
    }
}
}
```

Classe IndexarParaula

```
package pfc_oasis;

import com.sun.org.apache.xerces.internal.parsers.DOMParser;
import com.sun.org.apache.xml.internal.serialize.OutputFormat;
import com.sun.org.apache.xml.internal.serialize.XMLSerializer;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;

import java.util.Collections;
import java.util.Comparator;
import java.util.Date;
import java.util.Iterator;
import java.util.ListIterator;
import java.util.Set;
import java.util.StringTokenizer;
import java.util.Vector;
import java.util.jar.JarEntry;
import java.util.jar.JarInputStream;
import java.util.jar.JarOutputStream;
import java.util.jar.Manifest;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.Text;

import org.xml.sax.InputSource;


/*****************************************************************************
 * Classe principal per indexar un document OpenOffice. Espera tres paràmetres
 * per la línia de comandes.
 * paràmetre 1: ruta amb el fixter xml on s'indiquen les paraules a indexar
 * paràmetre 2: ruta del document OpenOffice que volem indexar
 * paràmetre 3: ruta del fitxer de sortida amb el document del paràmetre 2 indexat.
 *
 * ***************************************************************************/
public class IndexarParaules {

    private IndexReader indexReader; /*Classe que llegeix un arxiu XML amb les paraules
                a indexar i crea un array del tipus Index amb els mots llegits.*/
    protected Document document = null; /* Objecte per parsejar el document OpenOffice */
    protected Element documentRoot; /* Punter a l'arrel del document OpenOffice */
    protected NodeList headingElements; /* Llista d'elements de capçalera */
     //Darrer número de paràgraf, llista i secció en el document en curs
    int lastParaNumber = 0;
```

```
    int lastListNumber = 0;
    int lastSectionNumber = 0;
    // apuntador al darrer Node de paràgraf, llista i secció en el document en curs
    Node lastParaNode = null;
    Node lastListNode = null;
    Node lastSectionNode = null;
    /* Descriptors de fitxer */
    File inFile;
    File outFile;
    /* Streams (bytes) per llegir i escriure als fitxers JAR */
    JarInputStream inJar;
    JarOutputStream outJar;

    String sParagrafBreak;  //Indica el nom d'estil per fer el salt de paràgraf

    public static void main(String[] argv) {

        IndexarParaules indexParaules = new IndexarParaules();

//DEBUG   System.out.println(argv.length);
        if (argv.length < 3) {
            System.out.println("ERROR: número de paràmetres erroni");
            System.out.println("SINTAXI:  IndexarParaules  [indexXML]  [Fitxer  Entrada]  [Fitxer
Sortida] -f{font} -t{tipus} -m{mida} -c{color}");
            System.exit(0);
        } else {
            indexParaules.IndexarOpenOffice(argv);
            indexParaules.IndexarDocument(argv);
        }
    }

/*****************************************************************************
* Creem l'objecte de la classe IndexReader i afegim la informació a indexar.
*****************************************************************************/
    void IndexarOpenOffice(String[] argv) {
        //Creem un objecte per guardar les paraules a indexar. Guardem els atributs de la línia
de comandes
        indexReader = new IndexReader(argv);
//indexReader.llegirIndexos();   Obsolet. Versió Inicial PAC3
        System.out.println("Final Lectura paràmetres entrada.");
        indexReader.imprimirAtributs();
//System.out.println ("\tParaules en el document XML = "+ indexReader.getITotalIndexs());
//System.out.println("\tParaules a indexar = "+ indexReader.getIIndexs() );
//DEBUG indexReader.ImprimirMapaIndexos();
    }

 /*****************************************************************************
 * Aquest mètode fa tota la feina de cercar els índexs en el document i afegir aquesta
 * informació al nou document.
 * Utilitza els mètodes de la classe:
 *          openInputFile
 *          processarDocument
 *          escriuDocument
 *****************************************************************************/
    public void IndexarDocument(String[] argv) {
        JarEntry inEntry;
```

```
        JarEntry outEntry;

        /* Creem accessos als descriptors dels fitxers d'entrada i sortida*/
        inFile = new File(argv[1]);
        outFile = new File(argv[2]);
        //Intentem obrir el fitxer d'entrada
        if (openInputFile()) {
           Manifest manifest = inJar.getManifest();  /* Obtenim el manifest del fitxer d'entrada
*/
           /* Obrim el fitxer de sortida i copiem el manifest del fitxer d'entrada */
           try {
              if (manifest == null) {
                 outJar = new JarOutputStream(new FileOutputStream(outFile));
              } else {
                 outJar = new JarOutputStream(new FileOutputStream(outFile), manifest);
              }
           } catch (IOException e) {
              System.err.println("No puc obrir el fitxer de sortida " + argv[2]);
              System.exit(1);
           }

           System.out.println("\nHem creat el fitxer de sortida (" + argv[2] + ") amb el 'manifest'
del fitxer d'entrada (" + argv[1] +")\n");
           try {
              byte buffer[] = new byte[16384];
              int nRead;
              //llegim totes les entrades JAR del fitxer d'entrada
              //copiem totes les entrades diferents de content.xml
              while ((inEntry = inJar.getNextJarEntry()) != null) {
//DEBUG   System.err.println(inEntry.getName());
                 if (!inEntry.getName().equals("content.xml")) {
                    System.out.println("Copiant..." + inEntry.getName());
                    outEntry = new JarEntry(inEntry);
                    outJar.putNextEntry(outEntry);
                    while ((nRead = inJar.read(buffer, 0, 16384)) != -1) {
                       outJar.write(buffer, 0, nRead);
                    }
                 }
              }
              inJar.close();
              System.out.println("Hem copiat tots els fitxers JAR a la sortida excepte
content.xml\n");

              //Treballarem amb content.xml per indexar el document
              if(openInputFile()){
                 //Busquem l'entrada content.xml dins del fitxer JAR d'entrada
                 while ((inEntry = inJar.getNextJarEntry()) != null &&
                   !(inEntry.getName().equals("content.xml"))) {
                 }
                 //Creem una entrada al fitxer de sortida JAR pel fitxer context.xml amb la data
actual
                 outEntry = new JarEntry(inEntry);
                 outEntry.setTime(new Date().getTime());
                 outJar.putNextEntry(outEntry);
                 System.out.println("Hem creat l'entrada per content.xml al fitxer de sortida(" +
argv[2] + ")\n");
```

```
                //Obtenim content.xml en una estructura DOM (Representació en arbre)
                document = readContent(); /* parse content.xml */
                processarDocument(); /* Afegir els estils i indexar el document */
                escriuDocument(); /* escriure l'índex al fitxer JAR de sortida */

                System.out.println("Document Indexat");
                outJar.close();
            }
            else{
                System.err.println("No puc Obrir el fitxer d'entrada '" + argv[1] + "'");
                System.exit(1);
            }
          } catch (IOException e) {
            System.err.println("Error creant el fitxer de sortida");
            e.printStackTrace();
          }
        }
        else{
          System.err.println("No puc Obrir el fitxer d'entrada '" + argv[1] + "'");
          System.exit(1);
        }

    }

  /*****************************************************************************
   * Funció que intenta accedir a un fitxer que es troba a la ruta indicada
   * en el descriptor de fitxer inFile (atribut de la classe).
   * RETORNA: true  ---> si ha pogut obrir-lo.
   *          false ---> no s'ha pogut obrir.
   *****************************************************************************/
    public boolean openInputFile() {
        boolean bObrir = false;
        try {
            inJar = new JarInputStream(new FileInputStream(inFile));
            bObrir = true;
        } catch (IOException e) {
            bObrir = false;
        }
        return (bObrir);
    }

    /*****************************************************************************
     *  Mètode que busca dins del document OpenOffice examinat si existeix la font
     *  definida per representar els mots a indexar. Aquesta font és un atribut
     *  (font) de l'objecte indexReader. Si no existeix cal afegir una entrada
     *  d'estil per aquella font sempre i quan sigui una de les fonts que definim
     *  com a vàlides per aquesta aplicació. Aquestes fonts són "Arial" "Times
     *  New Roman" i "Comic Sans MS". Si volem afegir més fonts vàlides hauríem
     *  d'incloure-les en les sentències if-else if amb els atributs corresponents
     *  a la nova font.
     *  Retorna =0 --> La font ja existeix
     *          =1 --> Font afegida al document
     *          =2 ---> La font no existeix però no s'ha afegit doncs no és una font
     *                  correcte (Arial, Comic Sans MS o Times New Roman.
     *  ***************************************************************************/
```

```
   public int afegirFonts (IndexReader mIndex){
      int iNovesFonts = 0;
      boolean bFontTrobada;
      NodeList children;
      Node nodeFonts, child;
      Element element;

      //Set <String> sParaulesIndex = mIndex.ObtenirParaulesIndex();
      //Iterator <String> it = sParaulesIndex.iterator();

      //while (it.hasNext()){
         //String clau = it.next();
         //String sFont =mIndex.getFont(clau);
         String sFont =mIndex.get_font();
         //Busquem per tos els valors si ja tenim la font.
         nodeFonts = findFirstChild(documentRoot, "office:font-face-decls");
         children = nodeFonts.getChildNodes(); //Obtenim els fills de font-face-decls
         int i=0;
         bFontTrobada=false;
         while ( (bFontTrobada==false) && (i < children.getLength())){
            //DEBUG      System.out.println("Clau " + clau + " BUCLE " + i + " / "+
children.getLength());
            child = children.item(i);   //Obtenim el Node fill
            if (child.getNodeType() == Node.ELEMENT_NODE){    //Si és un ELEMENT el
tractem
               element = (Element) child;
               //Mirem si és del tipus font-face
               if (element.getTagName().equals("style:font-face")){
                  String fontName = element.getAttribute("style:name");
                  //DEBUG   System.out.println("\tfont : " + fontName);
                  if(fontName.equals(sFont)){
                     System.out.println ("Font Trobada " + sFont );
                     //mIndex.FontExisteix(clau);
                     mIndex.SetFontValida();
                     bFontTrobada = true;
                     //iNovesFonts++;
                     if    ((sFont.equals("Arial"))    ||    (sFont.equals("Comic    Sans    MS"))    ||
(sFont.equals("Times New Roman"))){
                        //mIndex.NovaFontOpenOffice(clau);
                     }
                  }
               }
            }
            i++;
         }
         //Si no trobem la font cal afegir-la
         if (bFontTrobada==false){
            iNovesFonts=2;
            if     ((sFont.equals("Arial"))     ||     (sFont.equals("Comic     Sans     MS"))     ||
(sFont.equals("Times New Roman"))){
               //mIndex.NovaFontOpenOffice(clau);
               iNovesFonts=1;
               mIndex.SetFontValida();
               Element elementFont = document.createElement("style:font-face");
               elementFont.setAttribute("style:name",sFont);
               elementFont.setAttribute("style:font-pitch","variable");
```

```
                elementFont.setAttribute("svg:font-family",sFont);
                if (sFont.equals("Arial")){
                    elementFont.setAttribute("style:font-family-generic","script");
                }
                else if(sFont.equals("Comic Sans MS")){
                    elementFont.setAttribute("style:font-family-generic","script");
                }
                else if(sFont.equals("Times New Roman")){
                    elementFont.setAttribute("style:font-family-generic","roman");
                }
                //Inserim la nova font
                nodeFonts.insertBefore(elementFont, nodeFonts.getFirstChild());
                System.out.println("Afegint font ..." + sFont );
            }
          }
        //}
        return iNovesFonts;
    }

    /*****************************************************************************
     *  Afegeix dins de content.xml la definició d'estils de paràgraf, secció i llista
     *  que tenim a l'objecte IndexReader per a cada paraula.
     *  Afegeix a l'inici del document la informació sobre els índexos trobats.
     *  Utilitza els mètodes de la classe:
     *      findFirstChild
     *      findLastItems
     *****************************************************************************/
    public void processarDocument(){
        Node           autoStyles;
        Node           fonts;
        Element        officeBodyStart; /* element <office:body> */
        Element        officeTextStart; /* element <office:text>  */
        Element        textStart;       /* posició on inserirem els indexs */
        Element        element;

        if (document == null) {
            return;     //Verifiquem que tenim un document a examinar.
        }
        documentRoot = (Element) document.getDocumentElement(); //Obtenim l'arrel del
document
        String sTag = "text:h"; //"office:text";
        headingElements = document.getElementsByTagName(sTag);  //Busquem els TAGS
del text
        if (headingElements.getLength() == 0){
            sTag = "text:p";
            headingElements = document.getElementsByTagName(sTag);     //Busquem els
TAGS del text
            if (headingElements.getLength() == 0){
                return;
            }
        }

        //************Afegirem els estils i fonts definits a IndexReader******

        //********************** Afegir Fonts *********************************
        //Només permetrem tres tipus de fonts Arial, Comic Sans i Times New Roman
```

```
        //--Busquem el primer Node de l'element <office:font-face-decls>
        System.out.println("*****AFEGIR FONTS *****");
        int iFont = afegirFonts (indexReader);
        if ( iFont == 1){
            System.out.println ("Afegida la font " + indexReader.get_font()+ " en el document
OpenOffice");
        }
        else if (iFont == 2){
            System.out.println ("La font " + indexReader.get_font()+ " no és vàlida");
        }
        else
            System.out.println ("La font " + indexReader.get_font()+ " ja existia en el document
OpenOffice");


        //--Busquem el primer Node de l'element <office:automatic-styles>
        autoStyles = findFirstChild(documentRoot, "office:automatic-styles");   //Busquem el
node d'autoestils
        //--Busquem el primer número de paràgraf, llista i secció no ocupat en aquest
document
        findLastItems(autoStyles);

        //Ens situem en els primers valors lliures de paràmetre, llista i secció
        lastParaNumber++;
        lastListNumber++;
        lastSectionNumber++;
        //Si no hem trobat cap node de paràmetre anem al següent node des de la posició
actual
        if (lastParaNode != null){
            lastParaNode = lastParaNode.getNextSibling();
        }
        //Si no hem trobat cap node de llista anem al següent node des de la posició actual
        if (lastListNode != null){
            lastListNode = lastListNode.getNextSibling();
        }
        //Si no hem trobat cap node de secció anem al següent node des de la posició actual
        if (lastSectionNode != null){
            lastSectionNode = lastSectionNode.getNextSibling();
        }

        //Crearem un estil de paràgraf per cada font afegida a font-face-decls
         //Set <String> sParaulesIndex = indexReader.ObtenirParaulesIndex();
         //Iterator <String> it = sParaulesIndex.iterator();
         String sParaPerDefecte = "P" + lastParaNumber;
         Element properties;
         //while (it.hasNext()){
            //String clau = it.next();
            //String sFont = indexReader.getFont(clau);
             String sFont = indexReader.get_font();
            //String sColor = indexReader.getColor(clau);
             String sColor = indexReader.get_color();
            if (indexReader.isFontValida()){//indexReader.FontAfegida(clau)){
                System.out.println("Afegint paràgraf ..." + lastParaNumber);
                //Creem un nou element <style:style> pel nou paràgraf
                element = document.createElement("style:style");
                element.setAttribute("style:name", "P" + lastParaNumber);
```

```
            element.setAttribute("style:family", "paragraph");
            element.setAttribute("style:list-style-name", "L" + lastListNumber);
            element.setAttribute("style:parent-style-name", "Standard");
            //Afegim els propietats de la font
            properties = document.createElement("style:text-properties");
            if (sColor.compareToIgnoreCase("vermell")==0)
               properties.setAttribute("fo:color","#ff0000");
            else if (sColor.compareToIgnoreCase("vert")==0)
               properties.setAttribute("fo:color","#00ae00");
            else if (sColor.compareToIgnoreCase("blau")==0)
               properties.setAttribute("fo:color","#0000ff");
            //Per defecte fo:color és negre
            properties.setAttribute("style:font-name",sFont);
            String sTipus = indexReader.getTipus();
            //if (indexReader.getPuntsTipus(clau).equals("bold"))
            if ( sTipus.compareToIgnoreCase("bold")==0)
               properties.setAttribute("fo:font-weight","bold");
            else
               properties.setAttribute("fo:font-style", sTipus);
            properties.setAttribute("fo:font-size",indexReader.getMida());
            element.appendChild(properties);
            //Inserim el nou estil de paràgraf
            autoStyles.insertBefore(element, lastParaNode);
            //indexReader.GuardarParagraf(clau,  "P" + lastParaNumber);
            indexReader.setParagraf("P" + lastParaNumber);
            lastParaNumber++;
            if (lastParaNode!=null){
               lastParaNode=lastParaNode.getNextSibling();
            }
         }
         else{
            //indexReader.GuardarParagraf(clau, sParaPerDefecte);
            indexReader.setParagraf(sParaPerDefecte);
         }
      //}
      //Afegim un estil de paràgraf pel salt de línia
       element = document.createElement("style:style");
       element.setAttribute("style:name", "P" + lastParaNumber);
       element.setAttribute("style:family", "paragraph");
       element.setAttribute("style:parent-style-name", "Standard");
       properties = document.createElement("style:paragraph-properties");
       properties.setAttribute("fo:break-before","page");
       element.appendChild(properties);
       properties = document.createElement("style:text-properties");
       properties.setAttribute("style:font-name","Arial");
       element.appendChild(properties);
       autoStyles.insertBefore(element, lastParaNode);
       if (lastParaNode!=null)
          lastParaNode=lastParaNode.getNextSibling();
       sParagrafBreak = "P" + lastParaNumber;
       lastParaNumber++;

      //Creem un nou element <style:style> per a la nova secció
      //En aquesta secció és on inclourem l'índex
      element = document.createElement("style:style");
      element.setAttribute("style:name", "Sect" + lastSectionNumber);
```

```
        element.setAttribute("style:family", "section");

        //--afegim valors per defecte a la secció
        addSectionProperties(element);

        //afegim el nou estil de secció
        autoStyles.insertBefore(element, lastSectionNode);

        //Creem un estil de tipus llista
        element = document.createElement("text:list-style");
        element.setAttribute("style:name", "L" + lastListNumber);
        //--Afegim bullets a la definició
        addBullets(element);
        //afegim el nou estil de llista
        autoStyles.insertBefore(element, lastListNode);

      /* Inserim el text dins d' <office:body> <office:text>.
       * Després de la primera aparició de <text:sequence-decls> */
        //--Busquem el primer node fill de <office:body>
        officeBodyStart = findFirstChild(documentRoot, "office:body");
        //--Busquem el primer node fill de <office:text>
        officeTextStart = findFirstChild(officeBodyStart, "office:text");
        //--Busquem el primer node de <text:sequence-decls>
        textStart = findFirstChild(officeTextStart, "text:sequence-decls");
        //--busquem el final de 'text:sequence-decls'
        textStart = getNextElementSibling( textStart );

        //creem una secció al text amb els valors calculats
        element = document.createElement("text:section");
        element.setAttribute("text:style-name", "Sect" + lastSectionNumber);
        element.setAttribute("text:name", "Section" + lastSectionNumber);
        //--Afegim les capçaleres amb els índexos dins del text

        cercarIndexos (element);
        addHeadings(element);

        //Anem al final del text
        Node nl = (Node) textStart;
        while (nl != null){
//DEBUG         if    (nl.getNodeType()    ==    Node.ELEMENT_NODE){System.out.println
(nl.getNodeName());
            nl = nl.getNextSibling();
        }
        textStart = (Element) nl;

        //Afegim salt de pàgina
        Element elementTmp = document.createElement("text:p");
        elementTmp.setAttribute("text:style-name",sParagrafBreak);
        officeTextStart.insertBefore(elementTmp, textStart);
        //Afegim el text de l'índex
        elementTmp = document.createElement("text:p");
        elementTmp.setAttribute("text:style-name",sParaPerDefecte);
        Text text = document.createTextNode("");
        text.appendData("INDEX");
        elementTmp.appendChild(text);
        officeTextStart.insertBefore(elementTmp, textStart);
```

```
        //Afegim les paraules a indexar
        officeTextStart.insertBefore(element, textStart);

        //Guardem l'índex en un document xml
        indexReader.guardarIndex();
    }

    /*************************************************************************
    * Busquem el primer Node des d' startNode on apareix l'element tagName.
    * Retorna un apuntador a aquest node
    * ************************************************************************/
    public Element findFirstChild(Node startNode, String tagName){
        startNode = startNode.getFirstChild();
        while (! (startNode != null && startNode.getNodeType() == Node.ELEMENT_NODE
&&
            ((Element)startNode).getTagName().equals(tagName)))
            {       startNode = startNode.getNextSibling();       }
        return (Element) startNode;
    }

    /*************************************************************************
     * Busca el darrer número i node pels elements llista, estil-secció i
     * estil-paràgraf dins del node "office:automatic-styles"
     * Són escrits als atributs de classe:
     * Número:  lastParaNumber
     *          lastSectionNumber
     *          lastListNumber
     * Nodes:   lastParaNode
     *          lastSectionNode
     *          lastListNode
     * ************************************************************************/
     public void findLastItems(Node autoStyle){
        NodeList  children;
        Node child;
        Element element;

        children = autoStyle.getChildNodes(); //Recorrem tots els fills d'office:automatic-styles
        for (int i = 0; i < children.getLength(); i++){
            child = children.item(i);   //Obtenim el Node fill
            if (child.getNodeType() == Node.ELEMENT_NODE){     //Si és un ELEMENT el
//tractem
                element = (Element) child;
                //Mirem si és del tipus estil
                if (element.getTagName().equals("style:style")){
                    String styleName = element.getAttribute("style:name");
                    //Si és un estil-paràgraf
                    if  (element.getAttribute("style:family").equals("paragraph")){
                        int paraNumber = Integer.parseInt(styleName.substring(1));
                        lastParaNumber = Math.max(paraNumber, lastParaNumber);
                        lastParaNode = child;
                    }
                    //Si és un estil-secció
                    else if (element.getAttribute("style:family").equals("section")){
                         int sectionNumber = Integer.parseInt(styleName.substring(4));
                         lastSectionNumber = Math.max(sectionNumber, lastSectionNumber);
```

```
                lastSectionNode = child;
            }
        }
        //Mirem si és un estil de llista
        else if (element.getTagName().equals("text:list-style")){
            String styleName = element.getAttribute("style:name");
            int listNumber = Integer.parseInt(styleName.substring(1));
            lastListNumber = Math.max(listNumber, lastListNumber);
            lastListNode = child;
        }
      }
   }
}

/*************************************************************************
* Escriu el document (representació de content.xml) modificat amb els índexos
* al fitxer de sortida.
* ************************************************************************/
public void escriuDocument(){
   if (document == null){
      return;
   }
   PrintWriter out = null;
   try{
      out =new PrintWriter(new OutputStreamWriter(outJar, "UTF-8"));
   }
   catch (Exception e){
      System.out.println("Error creant el stream de sortida");
      System.out.println(e.getMessage());
      System.exit(1);
   }
   OutputFormat oFormat = new OutputFormat("xml", "UTF-8", false);
   XMLSerializer serial = new XMLSerializer(out, oFormat);
   try{
     serial.serialize(document);
   }
   catch (java.io.IOException e){
      System.out.println(e.getMessage());
   }
}

/*************************************************************************
* Llegeix el document d'entrada i construeix l'arbre del document
* en una variable del tipus Document
**************************************************************************/
public Document readContent( ) {
    try {
      DOMParser parser = new DOMParser();
      parser.setEntityResolver(new ResolveDTD());
      parser.parse(new InputSource(inJar));
      return parser.getDocument();
    }
   catch (Exception e){
       e.printStackTrace(System.err);
      return null;
    }
```

```
 }

/*******************************************************************************
 * Afegir propietats per defecte a un element de tipus Secció.
 * Creem l'element columns i properties. Columns és fill de properties i
 * properties es fill de sectionStyle.
 *                 sectionStyle  ---> properties ---> columns
 * *****************************************************************************/
public void addSectionProperties(Element sectionStyle){
   Element properties;
   Element columns;

   properties = document.createElement("style:section-properties");
   properties.setAttribute("text:dont-balance-text-columns","false");

   columns = document.createElement("style:columns");
   columns.setAttribute("fo:column-count", "0");
   columns.setAttribute("fo:column-gap", "0cm");

   properties.appendChild(columns);
   sectionStyle.appendChild(properties);
}

/*******************************************************************************
* Creem elements per definir bullets sobre un element del tipus estil de llista.
* Un element bullet i un de properties del bullet. Properties és fill de
* bullet i bullet de listLevelStyle.
*
*             listLevelStyle ---> bullet ---> propierties
* *****************************************************************************/
public void addBullets(Element listLevelStyle){
   int level;
   Element bullet;
   Element properties;

   for (level = 1; level <= 10; level++){
      bullet = document.createElement("text:list-level-style-bullet");
      bullet.setAttribute("text:level", Integer.toString(level));
      bullet.setAttribute("text:bullet-char", "\u2022");

      properties = document.createElement("style:list-level-properties");
      if (level != 1){
         properties.setAttribute("text:space-before",Double.toString((level-1) * 0.5) + "cm");
      }
      properties.setAttribute("text:min-label-width", "0.5cm");
      bullet.appendChild(properties);
      listLevelStyle.appendChild(bullet);
   }
}

/*******************************************************************************
*  Fem un recorregut dins d'un NODE ('node') en els seus fills fins arribar
*  al següent Sibling que és un ELEMENT.
* *****************************************************************************/
public Element getNextElementSibling( Node node ){
node = node.getNextSibling();
```

```
    while (node != null && node.getNodeType() != Node.ELEMENT_NODE){
      node = node.getNextSibling();
    }
    return (Element) node;
    }

    /*************************************************************************
     *  Aquesta rutina recorre tot el text del document OpenOffice llegint els
     *  diferents trossos de text. Per a cada tros separa les paraules que troba i
     *  les afegeix dins de l'índex que estem creant dinàmicament en un objecte de
     *  la classe IndexReader. En cas de coincidència amb alguna paraula ja indexada
     *  s'afegeix la referència a la llista de refèrencies associades a la paraula.
     * ************************************************************************/
     public void cercarIndexos (Element startElement){
        String sText, s="";
        long iContadorParaules=0, iParaulesIndexades=0;
        Node node=null;
        int totalNodes;

        Node iniciOfficeBody = findFirstChild(documentRoot, "office:body");
        //--Busquem el primer node fill de <office:text>
        Node iniciOfficeText = findFirstChild(iniciOfficeBody, "office:text");
        //--Busquem el primer node de <text:sequence-decls>
        Node iniciText = findFirstChild(iniciOfficeText, "text:sequence-decls");
        //--busquem el final de 'text:sequence-decls'
        iniciText = getNextElementSibling( iniciText );

         while (iniciText != null){
//DEBUG  System.out.println("node name " + iniciText.getNodeName());
           NodeList listOfNodes = iniciText.getChildNodes();
          totalNodes = listOfNodes.getLength();
          for (int q = 0; q < listOfNodes.getLength(); q++) {

           Node primerNodeFill = listOfNodes.item(q);
          if  (primerNodeFill.getNodeType() == Node.TEXT_NODE){
              Text text = (Text)primerNodeFill;
//DEBUG   System.out.println ("* " + firstPersonElement.getWholeText());
              sText = text.getWholeText();
              //Analitzar el text
              String sParaula="";
//DEBUG   System.out.println(sParaula);
                //Mirem si la paraula existeix dins del mapa d'índex
                 StringTokenizer st = new StringTokenizer (sText);
                 while (st.hasMoreTokens()){
                    sParaula = st.nextToken();
                    iContadorParaules++;

                    Set <String> sParaulesIndex = indexReader.ObtenirParaulesIndex();
                    Iterator <String> it = sParaulesIndex.iterator();

                    boolean bParaulaTrobada =false;
                    while (it.hasNext()){
                       String clau = it.next();
                       if (sParaula.equalsIgnoreCase(clau)){
                          indexReader.ParaulaTrobada(clau, iContadorParaules);
                          bParaulaTrobada = true;
```

```
                    }
                  }

                  //Si el mot no està indexat l'afegim.
                  if ( bParaulaTrobada == false ){
                    indexReader.AfegirParaulaIndex(sParaula, iContadorParaules);
                    iParaulesIndexades++;
                  }
                }
                s=s.concat(sText+"\n");
              }
            }
            iniciText = iniciText.getNextSibling();
            }
            indexReader.setITotalIndexs(iParaulesIndexades);
//DEBUG  System.out.println("********** text llegit ******************\n " + s);
//DEBUG indexReader.ImprimirMapaIndexos();
        }

        /*****************************************************************************
        * Afegim unes capçaleres amb el text a l'entrada de secció dins de "text"
        * ****************************************************************************/
        public void addHeadings(Element startElement) {
            int     headingLevel, i;
            Element element = null, listItem, paragraph;
            Text    textNode;
            String sIndex;
            ListIterator<String> itList;

            Set <String> sParaulesIndex = indexReader.ObtenirParaulesIndex();
            //Iterator <String> it = sParaulesIndex.iterator();
            i=0;
            Vector v = new Vector (sParaulesIndex);
            Collections.sort(v, new String ().CASE_INSENSITIVE_ORDER);
            Iterator <String> it = v.iterator();
            while (it.hasNext()){
                String clau = it.next();
                headingLevel = 0;
                document.createElement("text:list");
                element = document.createElement("text:list");
                element.setAttribute("text:style-name","L" + lastListNumber);

                listItem = document.createElement("text:list-item");
                paragraph = document.createElement("text:p");
                //paragraph.setAttribute( "text:style-name", indexReader.ObtenirParagraf(clau));
                 paragraph.setAttribute( "text:style-name", indexReader.getParagraf());
                textNode = document.createTextNode("");
                sIndex = clau + "  ";
                if (indexReader.getITotalIndexs()>0){
                    itList = indexReader.ObtenirReferencies (clau);
                    String sReferencia;
                    while (itList.hasNext()){
                        sReferencia = itList.next();
                        sIndex = sIndex + sReferencia + ", ";
                    }
                    sIndex = sIndex.substring(0, sIndex.length()-2);
```

```
            }
            textNode.appendData(sIndex);
            paragraph.appendChild(textNode);
            listItem.appendChild(paragraph);
            element.appendChild(listItem);
            i++;
            startElement.appendChild(element);
        }
        startElement.appendChild(element);
    }

}
```