Disseny i implementació d'un editor col·laboratiu de LaTeX vía web

Autor: Jesús Ojeda Contreras

Universitat de Lleida
Escola Politècnica Superior

Director: Josep Maria Ribó Balust

Enginyeria Tècnica en Informàtica de Gestió

Índex

Capítol 1

Introducció

Des de sempre s'ha necessitat treballar en grups o equips de persones per poder abordar tasques o problemes de gran extensió o que abarquen diferents camps. És aquí on entren les eines que facil·liten el treball coordinat de les persones integrants d'aquestos grups.

El principal problema està, en la majoria dels casos, en la sincronització de les diferents parts del treball global que cada membre del grup fa per separat, i és en aquest aspecte en el que intenten ajudar aquestes eines.

Donat que LaTeX és emprat àmpliament al món científic gràcies a la seva facilitat per l'expressió de tota mena de fórmules i funcions, moltes vegades equips sencers de matemàtics, físics, químics o altres treballen conjuntament per escriure articles o llibres. Sense eines col·laboratives, normalment una persona ha d'esperar a rebre el treball de l'altre per poder continuar o en el millor dels casos, cadascú treballa per la seva part i a posteriori es posa tot en comú per tal de donar-li la forma final al text.

Mitjançant el correu electrònic i la missatgeria instantània aquesta forma de treballar és més ràpida si cada persona treballa a un lloc diferent. No obstant, no seria més dinàmic i útil per l'equip poder treballar tots alhora coneixent els canvis que uns fan o desfan per poder adaptar la seva part o fins i tot corregir el treball de l'altre? Aquí és on entra la present aplicació.

1.1 Objectius

Aquesta aplicació permet la creació i edició de documents LaTeX mitjançant la web, per part d'un equip de persones que poden treballar sobre el document concurrentment, sempre i quan treballin sobre seccions diferents, ja que el sistema que controla e impossibilitat l'opció d'incoherències es basa en bloquejos de les seccions sobre les quals es treballa.

Pel desenvolupament d'aquesta aplicació web s'han ajuntat vàries tecnologies en pràctica en l'actualitat; com són, per exemple, JSP, Struts, AJAX -que al seu torn no és una tecnologia, sinó un aglomerat de vàries d'elles-, Javascript i *servlets* Java. S'han triat aquestes

concretament perquè estan en auge dins del marc de la Web 2.0 -AJAX bàsicament-.

S'ha triat la plataforma web perquè aquesta ens permet una accessibilitat total per part dels usuaris, sense limitar quin sistema operatiu utilitzin, ja que la web està basada en un conjunt d'estàndards que tothom ha de complir. L'únic requisit al que s'ha d'ajustar el client és tenir un navegador web compatible -Javascript pot ser problemàtic-.

De la mateixa manera, també s'ha llicenciat aquest software amb llicència GPL, de tal manera que qualsevol persona pugui accedir i millorar el codi per evitar errors o fins i tot fer-ne versions diferents, adaptant el codi a noves necessitats.

Capítol 2

Problema de l'edició cooperativa

Existeixen cert tipus de projectes, tasques o treballs en els quals el treball d'una única persona és insuficient. Ja sigui per la complexitat del mateix projecte o per la manca de temps es necessiten grups de treball en els que subdividir les tasques. Aquí entra en joc la col.laboració o cooperació.

Concretament, la col.laboració en projectes sol ser mitjançant text. D'aquesta manera ens trobem que s'ha de poder editar el mateix text -en ocasions en grans quantitats- per diferents persones. Així doncs, totes aquestes persones han de poder accedir al text per modificar-lo. Això planteja un primer problema: integritat. S'ha de buscar un mètode que ajudi a controlar els accesos concurrents i mantenir de qualsevol manera el text íntegre. Control de versions, bloqueig de parts... hi ha diferents maneres.

De qualsevol manera, no sempre les persones que formen aquest grups estan situades al mateix lloc -a vegades ni tan sols al mateix país!-. Això ens suposa un altre problema: la distribució. Tenim diferents maneres de solventar-ho, cadascuna amb pros i contres, com poden ser la centralització dels continguts en un servidor, o per altra banda la replicació de dades.

En qualsevol cas, aquest és un problema complex de resoldre i en el que cada aproximació és vàlida segons el context.

Actualment, totes les solucions -o almenys la gran majoria- fan ús de servidors i xarxes d'informació per tal de realitzar la seva escomesa, permetre l'edició cooperativa solventant d'una manera o altra els dos principals problemes comentats anteriorment.

No obstant, amb la gran implantació que ha tingut la WWW i el recent fenomen anomenat Web 2.0, les noves aproximacions al problema alliberen els clients d'instal.lacions i configuracions, ja que tot el contingut i el mateix editor són accessibles des de qualsevol navegador -que suporti l'aplicació-. Això provoca que totes les aplicacions que es fan dir Web 2.0 no surtin de l'estat beta, ja que les noves millores són introduïdes de manera transparent a l'usuari i sempre estan obertes a noves modificacions.

Un problema, però, que tenen aquestes pàgines web -són precisament això, aplicacions webs, en algun cas potser serveis web- és la dificultat de la seva programació i manteniment

del codi. Això és degut a que, tot i que hi ha molta part de programació per la banda del servidor, cal dir que hi ha molta programació a la banda del client. Si els usuaris accedeixen mitjançant un navegador, aquesta programació estarà feta amb Javascript, amb els problemes que això comporta.

Tot i això, aquestes aplicacions Web 2.0 permeten una major interacció amb l'usuari, simulant aplicacions d'escriptori en el seu navegador.

2.1 Descripció d'altres propostes

El present projecte no és l'únic en aquest camp, de fet n'hi ha molts altres més que permeten el treball col·laboratiu tant per l'escriptura com per altres utilitats com pot ser el tractament d'imatges. No obstant, aquí descriure una mica alguns dels que més s'apropen a la present aplicació, tot i que del següent llistat potser nomes un suporta l'edició i de LaTeX. Així, cadascun empra les seves pròpies tècniques, des de bloquejos de seccions -com el present projecte- fins a edició totalment simultània. La gran majoria d'aquestos editor es poden definir com WYSIWYG -*What You See Is What You Get*-.

2.1.1 MediaWiki

Mediawiki és un paquet de softwate amb llicència GNU GPL. Està basat en la web i escrit amb PHP i necessita de MySQL -encara que té suport també per PostgreSQL-. Inicialment va ser desenvolupat per servir les necessitats de la Wikipedia, l'enciclopèdia wiki lliure.

El software wiki és un tipus de lloc web que permet als usuaris afegir, esborrar o editar i canviar tot o part dels continguts, a vegades sense necessitat de registrar-se. Aquesta facilitat d'interacció i operació fa dels wikis una eina efectiva per l'escriptura col·laborativa.

Mediawiki és usat actualment per tots els projectes de la Wikimedia Foundation, tots els wikis que resideixen a Wikia, així com molts altres wikis populars. També se n'estan aprofitant de les seves caractirístiques empreses com Novell, utilitzant-lo com a sistema gestor de coneixement o com a gestor de continguts.

Mediawiki aporta un nucli ric en característiques i un mecanisme per afegir **Extensions** i proveïr funcionalitats afegides.

Va ser originalment escrit per la Wikipedia per Magnus Manske, estudiant a la Universitat de Colònia.

Ofereix les següents característiques:

- A diferència dels wikis clàssics, els noms de las pàgines no tenen perqué estar formats en "CamelCase", lo qual permet tenir noms més naturals.

- Espais de noms: permeten separar pàgines de diferents tipus. Així, es pot tenir un espai de noms per artícles, un altre per plantilles, debates, etc. que el software tracta de diferent manera.
- Pàginas de discussió: cada pàgina del wiki té una pàgina associada de discussió própia, dedicada a parlar sobre la millora o altres fins.
- Soport de TeX, per visualitzar fórmules matemàtiques. Les fórmules poden mostrar-se de varies maneres, segons las capacitats del navegador.
- Llistes de seguiment, de tal manera que cada usuari pugui seguir els canvis als artícles del seu interés.
- Sistema de plugins que permet extendre fàcilmente el software. Els plugins instal·lats són llistats automàticament en "Pàgines especials".
- Capacitat de bloquejar temporalmente usuaris o pàginas.
- Suport de plantilles personalitzades amb paràmetres.
- Creació de línies de temps a través de codi wiki.
- Sistema de categoríes jeràrquic, que permet fer llistats d'artícles o de "thumbnails"d'imatges.
- Admet diversos nivells d'usuari, així como la possibilitat de que només els usuaris registrats puguin editar, o d'impedir el registre de més usuaris. Així, pot utilizar-se com CMS o Groupware.
- Suport per memcached y el sistema de caché Squid.
- Pells ("skins") personalitzables per cada usuari.

Més informació: [21]

2.1.2 Zoho Writer

Zoho Writer també està basat en la web peró el codi és tancat. És un editor de text tradicional programat amb Javascript, sense informació de l'estructura del servidor. Està en versió beta. Segons les característiques que la seva web anuncia, es destaquen les següents:

- Possibilitat d'accedir, editar i compartir els documents des de qualsevol lloc.
- Creació, edició o reformatament de documents mitançant un editor WYSIWYG.

- Multiples usuaris poden treballar en un mateix document simultàniament.
- Capacitat d'importar arxius Microsoft Word (DOC), OpenOffice (ODT i SXW), HTML, RTF, JPG, GIF i PNG. Generate PDF/DOC/ODT
- Exportació dels documents creats a formats com PDF, DOC, ODT, SXW, RTF, HTML i text pla.
- Possibilitat de compartir els documents o publicar-los públicament.
- Zoho Writer pot publicar documents directament als blogs personals com WordPress, Blogger, TypePad i Live Journal.
- Permet control de versions. No cal múltiples cópies.
- Proveeix suport per múltiples llengües.

La mateixa companyia ofereix de la mateixa manera programes similars de fulla de càlcul i presentació de diapositives.

Més informació: [22]

2.1.3 Writely

Aquest és un altre editor de text vía web, actualment en versió beta. Va ser comprat per Google el 9 de Març de 2006.

Writely no té un límit màxim de memória per emmagatzemar arxius, peró imposa un parell de condicions: els arxius de text no poden excedir els 500KB i les imatges no poden ocupar més de 2MB. Una característica interessant és la possibilitat de guardar els documents en els següents formats: MS Word (DOC), Postscript (PS), Rich Text Format (RTF), ODF (ODT), HTML i PDF. D'aquesta manera s'aconsegueix una interoperabilitat més facil. Una altra característica és la integració amb serveis de blog, Blogger inclós.

Aquesta aplicació utilitza Ajax i és un dels exemples més comuns quan es parla sobre la Web 2.0. Actualment està implementada amb la tecnologia Microsoft ASP.NET. Es creu que això sera incompatible amb el suport de les teconologies basades en Linux que utilitza Google. No obstant, s'especula que Writely serà portat per ser executat en Linux sota el projecte Mono.

Més informació: [23]

2.1.4 SynchroEdit

SynchroEdit és un editor multiusuari i simultani basat en la web. Permet que múltiples usuaris editin un únic document al mateix temps, sincronitzant contínuament els canvis de manera que tots els usuaris tinguin sempre la mateixa versió.

L'editor principal és completament WYSIWYG, dinàmicament mostra els diferents estils de les lletres, a més de diferents tipus de justificacions, indentacions i estils de llistats al temps que l'autor ho va introduïnt. SynchroEdit també suporta un editor de text pla per document més bàsics. Per facilitar l'experiència multiusuari, l'editor marca amb diferents colors els canvis que fan diferents usuaris i manté una marca on cada usuari està actualment editant.

Actualment es troben en la versió alpha 0.4.2 i es basa en DOM bàsicament. El servidor que s'encarrega de respondre a les peticions està fet amb PHP i Perl, encara que tenen planejades implementacions mitjançant MediaWiki i Wordpress. El servidor que s'encarrega de la sincronització està fet amb Java, mentre que el client s'ha desenvolupat amb Javascript.

Més informació: [24]

2.1.5 Writeboard

Writeboard és un altre tipus d'editor de text vía web que permet compartir la informació i que altres usuaris, prèvia invitació puguin llegir i editar. Aquest editor permet desfer qualsevol canvi, ja que es basa en un control de versions. Un altre afegit que conté és la possibilitat de comentar els escrits. I per últim, permet fer subscripcions mitjançant RSS i així tenir notificacions cada cop que algú canvia el text.

Més informació: [25]

2.1.6 MoonEdit

MoonEdit és un editor col·laboratiu de text que permet a un conjut d'usuaris editar el mateix document mitjançant internet. Tots els usuaris poden modificar des de qualsevol lloc i en qualsevol moment, sense restriccions. Es poden veure els canvis que altres usuaris fan en temps real ja que cada usuari té assignat un color determinat, i aquest color serà el que denotarà els canvis fets. Va ser fet per Tom Dobrowolski l'any 2005. Aquest editor és una aplicació d'escriptori amb els següents requeriments mínims:

- Processador: Pentium II 400MHz
- SO: Windows 98, Windows XP o Linux
- Gràfics: SVGA 800x600

MoonEdit utilitza el protocol de xarxa UDP, i el port determinat és el 32132. Es pot treballar connectant a un servidor de MoonEdit que estigui actiu o un usuari fer de servidor durant la sessió de treball. Tots els altres usuaris es connectaran al servidor mitjançant la IP o URL. És totalment gratuït per ús no comercial.

Més informació: [26]

2.1.7 ACE

ACE és un altre editor col·laboratiu. També és una aplicació d'escriptori, peró té versions interoperables per les majors plataformes actuals, entre elles Windows, Linux i Mac, ja que utilitza la tecnologia Java.

Múltiples documents poden ser editats al mateix temps. A més, es poden compartir documents amb altres usuaris en diferents ordinadors, connectats a una xarxa de comunicació. Ace també pot descubrir usuaris i els documents que comparteixen en una xarxa local. Per tot això no cal cap tipus de configuració perquè ACE està basat en una xarxa de zero-configuració (també conegut com *Bonjour* o *Rendezvous*).

Un cop un usuari s'ha unit a l'edició d'un document compartit, pot editar-lo lliurement amb tots els participants com un equip virtual.

El nucli de l'aplicació es un algoritme de control de concurrència basat en el concepte innovatiu de Transfomació Operacional, que permet l'edició sense bloquejos d'un document per part de múltiples usuaris. No imposa cap restricció i resol tots els conflictes automàticament.

Més informació: [27]

2.1.8 Gobby

Gobby, com els anteriors, és un editor col·laboratiu en temps real que s'executa des del propi computador. Té disponibles versions per Windows, Mac OS X (no de forma nativa) i Linux, a més d'altres sabors d'UN*X.

Té les següents característiques:

- Col·laboració en temps real a través de canals encriptats. (a partir de la versió 0.4)
- Cada usuari té el seu propi color
- Chat d'estil IRC per comunicar-se amb els companys durant l'edició dels documents.
- Resaltat de sintaxi per molts llenguatges de programació.
- Edició de múltiples document en una sessió.
- Drag'n'drop de documents cap a Gobby.
- Sincronització del document sota petició.
- Suport per configuració zero.
- Suport Unicode.
- Gobby és sotfware lliure sota llicència GPL 2.

Més informació: [28]

2.1.9 Emacs

Es poden dir moltes coses sobre l'editor Emacs. Peró la que aquí ens ocupa és la col·laboració. En aquest sentit Emacs proveeix suportbàsic per l'edició col·laborativa sota el sistema X Window, usant la comanda *make-frame-on-display* o *File-New Frame on Display* per obrir més d'una vista sobre un mateix arxiu des de diferents ordinadors.

Capítol 3

Anàlisi de requeriments

3.1 Requeriments funcionals

L'exit o el fracàs d'una aplicació ve condicionat per la realització d'un bon anàlisi de les funcionalitats que ha de tenir aquesta. D'aquesta manera, de l'estudi de les necessitats es deriva un conjunt de requeriments funcionals que corresponen a les diferents característiques que l'aplicatiu ha de tenir.

Aquest capítol descriu els requeriments funcionals de la present aplicació.

3.1.1 Requeriments generals

L'objectiu final és la creació i edició de manera cooperativa de documents LaTeX, per la posterior compilació del document i la generació de l'arxiu pdf resultant. A més a més, es pot permetre l'accés a documents en mode de només lectura. Per tal de permetre l'edició cooperativa mantenint la integritat dels documents s'implementarà un sistema de bloquejos de parts que impedirà que dos usuaris entrin a editar la mateixa part del document alhora. No obstant, aquest sistema de bloquejos permetrà a un usuari consultar el contingut que un altre usuari estigui editant en el mateix moment.

3.1.2 Usuaris

Seran usuaris totes aquelles persones que s'hagin registrat correctament a l'aplicació des de la pantalla corresponent. Això implica la creació d'un espai virtual per cadascun on guardaran i organitzaran els seus propis documents, així com aquells sobre els que tinguin provilegis -edició o lectura-.

Cada usuari podrà crear nous documents i compartir-los amb les persones que cregui convenient, sempre i quan conegui els seus noms d'usuari.

L'eliminació d'un document només podrà ser efectuada pel seu creador. L'edició -o visualització en cas de només lectura- podrà ser efectuada sobre documents propis o altres,

sobre els quals ens hagin atorgat els respectius privilegis.

Aquests privilegis es poden asignar en el moment de la creació pero també en qualsevol altre moment accedint a les propietats del document, atorgant o revocant privilegis a altres usuaris i fins i tot canviant el títol associat al document.

Per tal de poder definir un usuari, es necessita:

- Login
- Password
- Nom
- Adreça de correu electrónic

El login és un camp identificador, per tant, no poden existir dos usuaris amb el mateix login.

3.1.3 Documents

Considerarem un document aquells que estiguin escrits en LaTeX. Per tal d'implementar el sistema de bloquejos per parts, aquest document s'ha de dividir en dites parts. Aquestes parts són: *chapter, section, subsection i subsubsection.* El text inicial del document -abans del primer chapter- es guardarà a part, i les seccions com bibliography i altres que vagin al final es guardaran juntament amb la última part -sigui quina sigui-.

LaTeX és un llenguatge complex i la seva sintaxi pot ser utilitzada de moltes maneres diferents. Per tant, i per facilitar la creació del parser que s'encarregarà de separar el text en parts, s'han d'imposar certes restriccions al text creat. Per exemple, per crear subparts d'una altra, s'ha de bloquejar la part superior i editar el text associat, creant la subpart i posant-la a disposició dels usuaris. D'aquesta manera es redueix la complexitat del parser, tot i que també la seva funcionalitat.

Els documents tindran llavors els següents camps:

- Identificador
- Títol
- Creador
- Llista d'usuaris amb permisos d'escriptura
- Llista d'usuaris amb permisos de lectura

3.1.4 Parts

Com ja he descrit abans, els documents es divideixen en parts. Es podrien pensar cada part -*chapter, section, subsection, subsubsection*- com elements diferents, pero realment contenen el mateix: text. D'aquesta manera, són elements molt similars, i gràcies a això només ens cal mantenir d'alguna manera l'estructura del document per poder recuperar-lo en ordre.

Així, aquestes parts seran definides com:

- Identificador
- Tipus
- Títol
- Text
- Si està Bloquejada
- Qui bloqueja

3.2 Casos d'ús

3.2.1 Cas d'ús: Login

Actors: Usuari.
Precondicio: -
Postcondició: L'usuari accedeix als continguts que té disponibles.

Flux bàsic:

1. L'usuari s'autentifica.
2. El sistema valida l'usuari.

Extensions:

- Si el sistema no valida l'usuari, aquest es informat degudament.

3.2.2 Cas d'ús: Logout

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió.
Postcondició: S'ha tancat la sessió de l'usuari.

Flux bàsic:

1. L'usuari demana el tancament de sessió.
2. El sistema tanca la sessió de l'usuari degudament per tal d'evitar nous accessos sense el permís de l'usuari.

Extensions:

- Si l'usuari tenia algun document obert i aquest ha estat modificat però no guardat perderà els canvis.

3.2.3 Cas d'ús: Creació d'un nou document

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió.
Postcondició: Un nou document de text en format LaTeX s'ha creat al sistema. L'autor del document és l'usuari.

Flux bàsic:

1. L'usuari accedeix a la creació d'un nou document facilitant el títol i el llistat d'usuaris que tindrà permís per accedir-hi tan per escriptura com per lectura.
2. El sistema crea un document sense contingut, amb l'estructura bàsica i el títol anterior amb el propi usuari com autor i creador, i l'emmagatzema.
3. El sistema fa accessible el document a tots els usuaris que tinguin permisos.

Excepcions:

- Si no es dona cap títol al document, aquest no serà creat.

3.2.4 Cas d'ús: Edició de les propietats (title, users) del document

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió i té algun document del qual és creador.
Postcondició: Les propietats del document han estat canviades.

Flux bàsic:

1. L'usuari demana les propietats d'un document.
2. El sistema facilita aquestes propietats.
3. L'usuari modifica les dades
4. El sistema actualitza aquestes dades, atorgant i denegant privilegis a altres usuaris.

Extensions:

- Si a un usuari se li deneguen els privilegis mentre esta en edició o consulta d'una part concreta del document, aquesta denegació es farà vàlida en quant abandoni la sessió o accedeixi a un document diferent.

3.2.5 Cas d'ús: Creació d'una nova carpeta

Actors: Usuari
Precondicio: L'usuari ha iniciat sessió.
Postcondició: L'usuari disposa d'una nova carpeta al seu directori virtual.

Flux bàsic:

1. L'usuari demana la creació d'una carpeta, facilitant un nom per la mateixa.
2. El sistema actualitza el directori virtual de l'usuari, afegint una nova carpeta amb el nom facilitat.

3.2.6 Cas d'ús: Eliminació d'una carpeta

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió.
Postcondició: S'ha eliminat la carpeta seleccionada per l'usuari del seu directori virtual.

Flux bàsic:

1. L'usuari demana l'eliminació d'una carpeta ja existent al seu directori virtual.
2. El sistema actualitza el directori virtual de l'usuari, eliminant dita carpeta.

Excepcions:

- Si la carpeta no és buida no serà eliminada.

3.2.7 Cas d'ús: Accés a una part

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió i ha seleccionat un document dels que té accés.
Postcondició: L'usuari té a la seva disposició el text associat a la part seleccionada.

Flux bàsic:

1. L'usuari selecciona la part a la que vol accedir
2. El sistema comprova els permisos de l'usuari i la disponibilitat de la part.
    - Opció 1. L'usuari té permisos de només lectura. El sistema li retornarà el text associat a la part en mode lectura.
    - Opció 2. L'usuari té permios d'escriptura i la part no està bloquejada. El sistema bloquejarà la part per aquest usuari i li retornarà el text per edició.
    - Opció 3. L'usuari te permisos d'escriptura i la part està bloquejada. Com en el primer cas, el sistema retorna el text associat en mode lectura.

3.2.8 Cas d'ús: Edició d'una part del document (chapter, section...)

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió i té accés a algun document amb permisos d'escriptura i ha bloquejat una part.
Postcondició: El document està modificat (encara no s'hi guarden els canvis).

Flux bàsic:

1. El sistema retorna el text de la part seleccionada.
2. L'usuari afegeix i/o modifica lliurement el contingut del text de la part seleccionada.

3.2.9 Cas d'ús: Guardar canvis

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió i ha modificat alguna part del document.
Postcondició: Les modificacions han estat guardades. L'usuari continua en mode edició.

Flux bàsic:

1. L'usuari demana guardar canvis.
2. El sistema guarda l'estat actual del document.

3.2.10 Cas d'ús: Compilació i descàrrega del document pdf

Actors: Usuari.
Afectats: Altres usuaris.
Precondicio: L'usuari ha iniciat sessió i té accés a un document.
Postcondició: El document ha estat compilat a pdf i l'usuari l'ha descarregat. Qualsevol usuari que hi tingui permisos, pot accedir-hi.

Flux bàsic:

1. L'usuari demana la compilació del document.
2. El sistema configura l'arxiu .tex corresponent i el compila.
3. El sistema retorna l'arxiu a l'usuari.

Extensions:

- Si el document ja estava compilat i l'arxiu pdf és més nou que la última modificació del document, no es compila i es retorna directament el pdf.
- Si el document ja estava compilat pero existexen modificacions noves al document, es torna a compilar i es retorna el pdf.

Excepcions:

- Si la compilació genera qualsevol tipus d'error que impedeixi la generació del pdf, l'arxiu retornat serà un log en text pla on estaran recollides les dades de la compilació.

3.2.11 Cas d'ús: Eliminació del document

Actors: Usuari.
Afectats: Altres usuaris.
Precondicio: L'usuari ha iniciat sessió i té un document del qual és creador.
Postcondició: El document ha estat esborrat completament.

Flux bàsic:

1. L'usuari demana l'eliminació del document.
2. El sistema elimina el document dels directoris virtuals de tots els usuaris que en tenien permisos.
3. El sistema suprimeix el document.

Extensions:

- Si hi ha algun usuari actiu en el document, l'eliminació no serà possible.

3.2.12 Cas d'ús: Donar d'alta un usuari

Actors: Usuari.
Precondicio:-
Postcondició: L'usuari pot iniciar sessió.

Flux bàsic:

1. L'usuari accedeix a la pàgina per donar-se d'alta i omple el formulari corresponent, les dades del qual són totes obligatòries.

2. El sistema registra l'usuari i li envia un e-mail amb les dades amb les que ha omplert el formulari.

3.2.13 Cas d'ús: Moure elements al directori virtual: Retallar

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió i té elements al seu directori virtual.
Postcondició: El sistema guarda en memòria l'element seleccionat que l'usuari vol retallar.

Flux bàsic:

1. L'usuari selecciona un element del seu directori virtual.
2. L'usuari demana retallar l'element.
3. El sistem guarda aquest element en memòria.

3.2.14 Cas d'ús: Moure elements al directori virtual: Enganxar

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió, té elements al seu directori virtual i prèviament a retallat algun.
Postcondició: L'element ha estat mogut.

Flux bàsic:

1. L'usuari selecciona un element del seu directori virtual on moure l'element prèviament retallat.
2. L'usuari demana enganxar l'element.
3. El sistem mou l'element.

Extensions:

- Si l'element seleccionat on moure el retallat es un document, es selecciona automàticament el seu parent, que serà una carpeta.

3.2.15 Cas d'ús: Pujar figures associades a un document

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió i ha seleccionat un document dels que té disponibles.
Postcondició: S'ha pujat la figura al servidor i s'ha associat al document.

Flux bàsic:

1. L'usuari selecciona una figura del seu propi ordinador i l'envia al servidor.
2. El sistema s'encarrega d'emmagatzemar-la adientment, associant-la al document seleccionat.

Extensions:

- Si ja existeix una figura amb el mateix nom, la nova figura tindrà afegit al seu nom un sufix amb un número, per tal de diferenciar-les.

3.2.16 Cas d'ús: Eliminar una figura

Actors: Usuari.
Precondicio: L'usuari ha iniciat sessió i seleccionat un document que té figures associades.
Postcondició: La figura ha estat eliminada.

Flux bàsic:

1. L'usuari selecciona una figura associada al document i demana la seva eliminació.
2. El sistema elimina la figura

Capítol 4

Model de domini

Els requeriments d'una aplicació modelitzen un domini, el qual pot estar format per diferents components. Aquest components poden ser independents o poden relacionar-se entre ells. En el moment en que es treballa amb una base de dades, es té la necessitat de modelitzar aquest domini per poder entendre els diferents components que el formen.

Després d'haver identificat totes les entitats i les relacions a partir dels requeriments funcionals descrits anteriorment, el model de domini resultant d'aquesta aplicació té el següent diagrama associat:

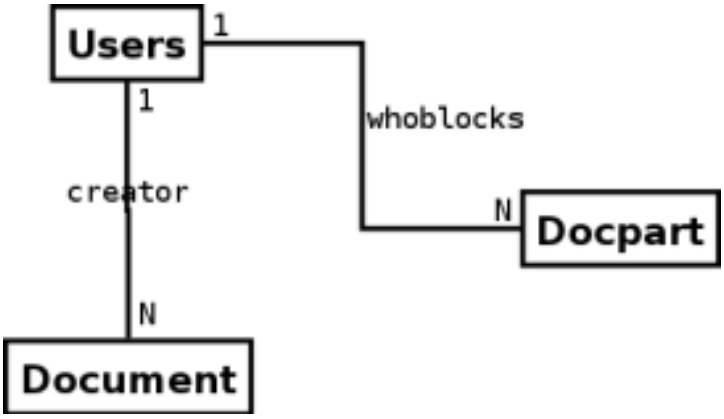

Figura 4.1: Model E-R

4.1 Entitats i relacions

Les entitats corresponen a les taules creades en el model relacional, les quals han estat modelitzades a partir de la informació extreta en la fase inicial de requeriments. Totes les entitats estan formades per atributs que representen les seves característiques.

Les entitats són:

- **Users**: Representa un usuari del sistema. Les dades més rellevants són el nom, l'identificador, la contrassenya i l'adreça de correu electrònic.
- **Document**: És la modelització d'un document LaTeX. La informació més significant associada és: el títol, el nom del creador i el propi identificador del document.
- **Docpart**: Modelitza una part d'un document. La informació que emmagatzema és: l'identificador de la pròpia part, el text que conté, el títol i si està bloquejada en algun moment i per qui.

I aquestes són les relacions:

- **creator**: Indica la relació que hi ha entre Users i Document. Aquesta relació permet associar un document a un usuari, concretament aquell que el va crear. Aquesta relació conté una cardinalitat del tipus 1 a N amb la qual cosa s'afegeix l'atribut identificador de l'entitat amb cardinalitat 1 (Users) a l'entitat amb cardinalitat N (Document).
- **whoblocks**: Indica la relació que hi ha entre Users i Docpart. Concretament, aquesta relació mostra qui bloqueja una part del document en un moment determinat. Es tracta d'una relació del tipus 1 a N amb la qual cosa s'afegeix l'atribut de l'entitat amb cardinalitat 1 (Users) a l'entitat amb cardinalitat N (Docpart).

Capítol 5

Aspectes de disseny i implementació

5.1 Arquitectura

5.1.1 Patró MVC

Un dels objectius de qualsevol aplicació -en aquest cas, web- és mostrar a l'usuari informació.

Les interfícies tenen tendència a modificar-se molt més respecte la lògica i les dades. Així, en un disseny en el qual no s'utilitza el patró MVC, les interfícies estan lligades amb la lògica, la qual cosa provoca que el manteniment del codi per part del desenvolupador es faci cada cop més difícil.

Per solucionar aquest problema, s'intenta dividir l'aplicació en petits mòduls que s'encarreguin de dur a terme funcions concretes. En concret, el patró Model Vista Controlador -MVC- ens permet dividir les aplicacions en diferents components, aconseguint així, una arquitectura en la qual es poden realitzar modificacions en el codi d'una manera més simple i aconseguir una millor reutilització.

El dividir l'aplicació en tres capes com fa el patró MVC, permet separar la presentació respecte l'accés de dades i les operacions que es realitzen sobre aquestes.Això imnplica que es pot dividir l'equip de treball en programadors i dissenyadors aconseguint que els temps de desenvolupament disminueixen ja que no tenen la necessitat de treballar en el mateix domini.

Tot seguit es descriuen les funcions de cadascun dels tres components del patró MVC:

- Model: El model representa les dades i les operacions que es realitzen sobre elles. Aquest és el component més reutilitzable. Responsabilitats:
  - Accedir a la capa d'emmagatzemament de dades.
  - Definir les funcionalitats del sistema -allò que és el que es fa amb les dades-.
  - Notificar a les vistes el seu canvi d'estat.

  - Encapsular les dades les quals són independents de la presentació.

- Vista: Mostra la sortida de les dades obtingudes del model a l'usuari. La vista és el component més susceptible a canvis. Responsabilitats:
  - Rebre i mostrar la sortida de les dades obtingudes del model a l'usuari.
  - Enviar informació de les peticions fetes per l'usuari al controlador.
- Controller: El controlador és l'encarregat de reenviar les interaccions que fa l'usuari a la vista, amb els models corresponents, actuant com a intermediari entre les vistes i els models. Responsabilitats:
  - Rebre events d'entrada.
  - Dur a terme determinades accions depenent de l'event d'entrada rebut.

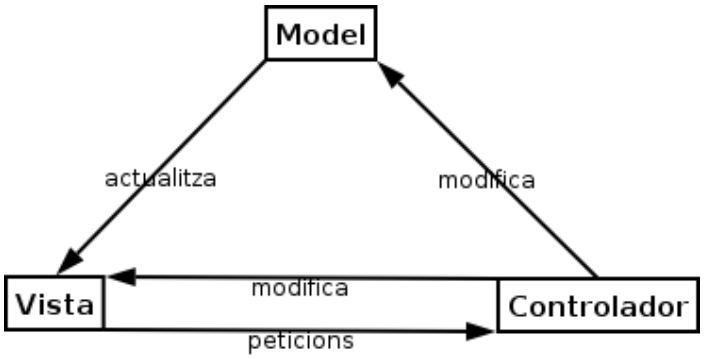

Figura 5.1: MVC

5.1.2 Front Controller

El patró Front Controller és un patró que adapta el patró MVC al disseny d'aplicacions web. El que fa aquest patró és centralitzar totes les peticions que realitza l'usuari en un mateix punt i és ell mateix, l'encarregat de delegar les peticions dels usuaris a les diferents operacions disponibles.

Aquesta centralització resulta d'utilitat per dur a terme diferentes funcions comuns en totes les parts de l'aplicació, fet que permet la reutilització i flexiblitat del codi.

El controlador es coordina mitjançant un component anomenat Dispatcher, reponsable de l'administració de les vistes i la navegació.

El flux bàsic del patró Front Controller es detalla a continuació:

1. El client realitza una petició.

2. La petició es capturada per un *servlet* que actua com a Front Controller.

3. El *servlet* s'encarrega d'obtenir la informació necessària de la petició i a partir d'aquesta s'encarrega d'escollir el model.

4. Un cop el model es seleccionat, transfereix el control a aquest.

5. Per finalitzar, la vista es retornada amb el model executat.

Donat que el mètode de peticions al servidor utilitzat -AJAX- no s'ajusta ben bé a l'anterior patró, s'ha optat per la creació de diferents *servlets* que agrupin funcionalitats relacionades, respectant al màxim la independència entre vista i dades, creant una situació similar a l'anterior però alhora certament diferent.

D'aquesta manera, totes les funcions a les que es pot accedir mitjançant la barra d'eines, s'han agrupat al `DoctreeHandler`, que s'encarrega de seleccionar el mètode correcte a executar, segons la petició de l'usuari.

De la mateixa manera, les funcions relacionades amb el bloqueig de parts i accés al text inherent, així com el altres funcions relacionades pròpiament amb el document, com l'obtenció del llistat de figures, o la compilació del mateix, s'han agrupat al `DocHandler`.

I per últim, l'encarregat de processar el text editat és el `TextHandler` i el que s'encarrega de tractar amb les figures pujades, `UploadFigure`.

Així doncs, tenim 3 controladors que reben totes les peticions i s'encarreguen d'enviar el missatges corresponents a les classes intermitges que fan d'intermediàries amb les dades -en aquest cas guardades en base de dades-.

5.2 Model de classes

A l'hora, de dissenyar les classes que conformen aquest treball, s'ha tingut en ment principalment el màxim aïllament entre dades, algoritmes i vistes.

D'aquesta manera s'han creat diferents classes, encapsulant cadascuna d'elles certes caraterístiques. Es poden diferenciar en diversos tipus:

- *Javabeans*

- *Wrappers*

Els *servlets* dels que he parlat abans s'encarreguen d'utilitzar les correspondents *Javabeans* i *wrappers* per tal de realitzar les funcions que té assignades.

5.2.1 Disseny de les classes

Javabeans

Els *JavaBeans* són uns components reutilitzables per la construcció d'aplicacions software escrits en Java.

Els *JavaBeans* són les classes de disseny de l'aplicació. De fet, són classes formades per simples mètodes *sets* i *gets*, a més d'altres possibles mètodes. El problema, però, es que en aquest treball, a certes classes no se'ls hi permet la modificació de certs dels seus atributs, per tant no tenen mètodes set per tals atributs.

Els utilitzats en el treball són:

- **Users**

  Login: Identificador de l'usuari.

  Passwd: Contrassenya.

  Name: Nom de l'usuari.

  Email: Adreça de correu electrònic.

  Path: Ruta absoluta al directori de l'usuari dins del disc dur.

- **TeXDocument**

  Doc_id: Identificador del document.

  Title: Títol del document.

  Creator: Identificador de l'usuari que ha creat el document.

  Users_rw: Llistat d'indentificadors d'usuaris que tenen accés en mode edició.

  Users_ro: Llistat d'identificadors d'usuaris que tenen accés en mode lectura.

  Docpath: Ruta absoluta al directori del document dins del disc dur.

  Lastmodified: Data de la última modificació feta al document.

- **DocPart**

  Part_id: Identificador de la part.

  Type: Tipus de la part (document, chapter, section...).

  Title:Títol de la part.

  Inner_text:Text contingut en aquesta part.

  Blocked:Flag que indica si la part està bloquejada (1) o no (0).

  WhoBlocks:En cas que la part estigui bloquejada, indica qui ha bloquejat aquesta part.

Wrappers

Aquests *Wrappers* encapsulen accesos a dades o certs mètodes i funcions que ha de ser accessibles en qualsevol punt reutilitzant codi i reduïnt el seu cost de manteniment.

Així doncs, aquestos són els utilitzats:

- **DBConnection**. Obté una connexió a la BD, ofereix mètodes per fer consultes, insercions, actualitzacions i esborrar.
- **HtmlStripper**. S'encarrega de treure o ficar tags html en un text, per tal de la seva representació a la web
- **TeXParser**. S'encarrega de separar en parts un text en LaTeX i alhora guardar-ho a la base de dades donat l'identificador de la part.
- **DBParser**. S'encarrega de canviar certs caràcters que poden ser conflictius a l'hora de guardar text a la bases de dades per altres combinacions de caràcters fàcilment reconeguts. També fa el procés invers.
- **DOMCreator**. Retorna un document DOM donat una ruta absoluta a un fitxer xml.
- **Mailer**. Encapsula la forma de crear i enviar correus elèctronics des de codi Java.

5.3 Disseny de la Base de Dades

Tenint en compte que necessitem un medi físic on guardar les dades, i que, a més, necessitem tenir un bon control de concurrencia, integritat i temps de resposta raonable, optem per utilitzar una base de dades -cas típic, d'altra banda-. S'ha optat per l'utilització de MySQL com a SGBD. Per tant, i partint del model de dades vist anteriorment, es crea l'estructura que conformarà la Base de Dades

5.3.1 Model relacional

```
CREATE TABLE users(
login varchar(20) not null,
passwd varchar(15) not null,
name varchar(100),
email varchar(100) not null,
path varchar(200) not null,
primary key (login)
);
```

```
CREATE TABLE docpart(
part_id varchar(200) not null,
type varchar(15),
title varchar(50),
inner_text longtext,
blocked bool unsigned,
whoblocks varchar(20) references users(login),
primary key (part_id)
);

CREATE TABLE document(
doc_id varchar(64) not null,
title varchar(100),
creator varchar(20) references users(login) not null,
users_rw tinytext,
users_ro tinytext,
docpath varchar(200) not null,
lastmodified timestamp,
primary key (doc_id)
);

CREATE TABLE doctreechanges(
user varchar(20) references users(login),
selItem varchar(64),
action varchar(20),
time timestamp,
primary key(user,selItem,action,time)
);
```

En el anterior codi apareix la taula `Doctreechanges` que abans no ha estat modelitzada. L'explicació es dona a la secció Detalls de la implementació.

5.4 Detalls de la implementació

5.4.1 Tecnologies emprades

En aquest treball s'han utilitzat diverses tecnologies per aconseguir arribar a l'objectiu de crear l'aplicació de manera més propera a l'usuari, tot i que continua sent una aplicació web.

El servidor utilitzat es Apache Tomcat (vegeu Apèndix A.8 i [11],[12] i [13]), qui ens servirà tots els *servlets*. La base de dades s'ha instal·lat sobre MySQL (vegeu Apèndix

A.11 i [15] i [16]), donat que la seva administració es força asequible. En qualsevol cas, ambdós programes tenen llicències de software lliure.

Mitjançant JNDI (vegeu Apèndix A.5), integrat a Tomcat, disposem de connexions a la BD gràcies als serveix que ofereix de noms i directoris. D'aquesta manera podem configurar externament els recursos de BD i no haver de modificar el codi si aquestos canvien.

També he utilitzat pàgines JSP (vegeu Apèndix A.6), no gaires, per presentar el contingut. Struts (vegeu Apèndix A.7 i [14]) m'ha servit, al seu torn, per fer més fàcil el control de dades inserides pels usuaris en certs formularis -no he aprofundit gaire en aquest *framework*, ja que només tinc de fet un parell de formularis on fer-lo servir-.

Per tota la resta de peticions que l'usuari pot fer he utilitzat AJAX (vegeu Apèndix A.1.1) de manera que només hi ha una única pàgina central on l'usuari desenvolupa la seva feina. D'aquesta manera, la pàgina guanya interactivitat. Per tal de fer fàcil l'ús d'AJAX he utilitzat un *framework* amb el nom de Prototype (vegeu Apèndix A.1.2 i [17]), el qual cada dia s'estén més i ja està integrat en força solucions com, per exemple, Ruby on Rails.

No només AJAX millora l'experiència de l'usuari, sinò també tots aquells elements que s'apleguen sota el nom DHTML -Dynamic HTML- (vegeu Apèndix A.2.2.). Aquests components permeten noves funcionalitats a les pàgines web. Els dos components que he utilitzat que es basin en aixó són dhtmlXtree (vegeu Apèndix A.2.3 i [19]) i FCKeditor (vegeu Apèndix A.2.4 i [18]).

Res d'això no funcionaria si no treballés amb Javascript (vegeu Apèndix A.2.1), llenguatge d'script molt estés i difícil de depurar.

XML (vegeu Apèndix A.3) és un estàndar i molt útil, a més a més. Mitjançant XML faig els arxius descriptors dels directoris virtuals dels usuaris, ja que en realitat, aquest directoris no existeixen. Després i per tal de poder manipular aquests documents utilitzo JDOM i XPath (vegeu APèndix A.4 i [4],[5] i [6]); el primer per representar els arxius xml com estructures arbòrees, i el segon per poder fer cerques de manera ràpida i certera dins d'aquests arbres DOM.

Finalment, utilitzo dos tecnologies, o més aviat, funcionalitats, implementades en Java. JavaMail (vegeu Apèndix A.9 i [7] i [8]) em permet enviar i rebre correu electrònic. En aquest cas, jo ho he utilitzat per enviar correus als usuaris un cop s'han registrat per tal que coneguin i recordin les dades que han introduït a l'hora de fer el registre. FileUpload (vegeu Apèndix A.10 i [9] i [10]), al seu torn, em facilita el tema de pujar i tractar arxius al servidor. Això ho he emprat per poder pujar les figures que poden ser incloses dins dels documents LaTeX.

5.4.2 Arbre de directoris i fitxers virtual

A la pàgina principal de l'aplicació ens trobem un arbre que descriu l'espai virtual de que l'usuari disposa. Això s'ha aconseguit gràcies a l'ús d'un component DHTML: dhtmlXTree

(vegeu Apèndix A.2.3 i [19]). La definició d'aquest espai virtual està integrada a un document xml. Això ens permet la modificació del propi arbre de manera més senzilla. Però alhora genera un problema de concurrencia quan es volen actualitzar els xml d'altres usuaris, per tal de fer-los-hi accessible un document nou. Aquest problema es solventa a la pròxima part. Aquest arbre ens permet la utilització de diferents tipus d'imatges per cada element que puguem posar. Així podem tenir diferents icones per cada tipus d'element (directoris, documents propis, només lectura o lectura/escriptura).

5.4.3 Accés concurrent per la modificació dels arxius xml descriptors dels directoris virtuals

Cada usuari té associat un document *Doctree.xml* que representa el seu directori virtual com el següent:

```
<?xml version="1.0" encoding="UTF-8"?>
<tree id="0">
<item text="/" id="root" im0="folderClosed.gif" im1="folderOpen.gif"
im2="folderClosed.gif" open="1" select="1">
<userdata>root</userdata>
<item text="Own TeXs" id="own_texs" im0="folderClosed.gif" im1="folderOpen.gif"
im2="folderClosed.gif">
<userdata>directory</userdata>
</item>
<item text="Collaborated TeXs" id="collaborated_texs" im0="folderClosed.gif"
im1="folderOpen.gif" im2="folderClosed.gif">
<userdata>directory</userdata>
</item>
</item>
</tree>
```

En el moment en que s'atorguen o es deneguen privilegis sobre un document a altres usuaris, s'ha de modificar el document xml associat a cada usuari per manifestar el fet que se li ha obert la possibilitat d'accedir a un nou document. Si cada usuari s'encarrega de fer aquestes modificacions sobre els xml dels altres usuaris estem incorrent en un problema d'accés concurrent i tenim la possibilitat de tenir errors d'integritat. S'obren, almenys, 2 alternatives.

La primera manera consisteix en guardar aquestes modificacions a la BD i cada usuari llavors es fa càrrec de les que li afecten a ell mateix, realitzant els canvis a l'hora d'iniciar sessió, per exemple. D'aquesta manera els canvis que l'usuari fa sobre el seu mateix xml són en temps real, però l'afegiment o eliminació de documents de col·laboració només es fan efectius quan l'usuari inicia sessió. Això l'obliga a entrar i sortirde l'aplicació.

La segona manera consisteix en crear un *thread* paral.lel que monitoritza la BD per veure si es demanen canvis a qualsevol xml. D'aquesta manera, i en el mateix moment de trobar que es necessiten canvis, s'encarrega de dur a terme aquest canvis. Això centralitza qui fa els canvis als xml en aquest *thread*. També aconsegueix que els canvis es facin en temps real.

S'ha implementat la primera opció després de fer vàries proves amb la segona, degut a que la segona opció realment és més lenta i era més difícil poder controlar el *thread* per tal que no consumís excessius recursos.

Per tal d'aconseguir aquesta funcionalitat, es va crear una altra taula a la BD que emmagatzema els canvis requerits als xml. Aquesta taula es diu `Doctreechanges`.

5.4.4 Bloqueig de les parts de manera atòmica

Per poder fer el bloqueig d'una part i poder accedir per escriure una part necessitem comprovar primer si ja estava bloquejada amb anterioritat per un altre usuari. Llavors com fer-ho la comprovació i bloqueig de forma atòmica, per evitar problemes de integritat en cas que dos usuaris vulguin bloquejar la mateixa part al mateix moment?

El propi SGBD ens proporciona una manera: l'ús de les transaccions. El problema d'això és que Tomcat sempre accedeix a la BD des del mateix usuari -definit a *context.xml*-. Per tant, aquesta aproximació la descartem des del principi.

La solució és una altra. L'usuari A bloquejarà totes les parts inferiors i l'actual, sempre i quan no estiguin bloquejades prèviament per un altre usuari. Llavors comprovarà si totes les parts que ha bloquejat són en realitat totes les parts existents. En cas afirmatiu, podrà agafar el text per editar-lo. En cas contrari, això indicaria que un altre usuari ja havia bloquejat alguna part. Així, l'usuari A haurà de tornar ha desbloquejar les parts que havia bloquejat.

Si en qualsevol moment durant l'anterior procés un altre usuari B intentés bloquejar, no podria fer-ho, ja que la pròpia BD atòmicament fa les actualitzacions. D'aquesta manera, l'usuari B en intentar bloquejar es trobaria que ja estava tot bloquejat i només podria accedir al text en mode lectura.

5.4.5 Manteniment de l'estructura dels documents TeX a BD i TeXParser

Inicialment es va pensar en un l'extracció de l'estructura del document TeX cap a un fitxer xml associat al document. Llavors només recorrent-lo coneixeríem les parts que formaven el document i en quin ordre. El problema d'aquesta aproximació és la generació d'un problema d'accés concurrent a aquest xml. Donat que hi ha un xml per document i documents hi pot haver molts, crear un unic *thread* resultaria insuficient per mantenir les actualitzacions de tots els xml en temps -quasi- real. La solució seria emprar un pool

de *threads*. Una solució un tant complicada donat tot el treball ja realitzat i el poc temps de marge a errors.

Per tant, es decideix que l'estructura sigui guardada de forma inherent a la BD -havent de recorrer-la cada cop que es vol obtenir el llistat de parts-. La idea és que no calguin més accessoris a part de les pròpies claus identificatives de les parts. D'aquesta manera:

Llistat de claus per ordre alfabètic ascendent

```
al0n3_1381478412_86484446.0
   al0n3_1381478412_86484446.0_chapter1
       al0n3_1381478412_86484446.0_chapter1_section1
       al0n3_1381478412_86484446.0_chapter1_section2
           al0n3_1381478412_86484446.0_chapter1_section2_ssection1
   al0n3_1381478412_86484446.0_chapter2
       al0n3_1381478412_86484446.0_chapter2_section1
   ...
```

Així aconseguim mantenir l'estructura del document de manera implícita a la BD. Per tal d'aconseguir aquest funcionament, el TexParser -que és qui s'encarrega de separar les parts donat un text i emmagatzemar-lo en la BD- esborra les parts contingudes a la part actual de manera que en separar el text torna a introduír-ho tot.

Una de les restriccions imposades al TeXParser és la separació de només parts inferiors. Amb un exemple es veurà més clar:

Donat que un usuari està editant la *section1* del *chapter1*, en el seu text només podrà incloure *subsections* i *subsubsections*. En el cas que vulgui afegir un altra *section*, hauria de bloquejar el *chapter1* sencer per poder afegir-la. Si per un cas d'error l'usuari inclogués una nova *section*, aquesta seria guardada com a part del text de la última part reconeguda en allò que esta editant. Aquest error es soluciona per si sol, quan un cop bloqueja la part superior per edició, en el moment que `TeXParser` separi totes les parts del *chapter*, llavors guardarà correctament les *sections*.

5.4.6 Editor

Per tal de poder incloure un editor a l'aplicació (sense editor l'aplicació no podria fer res) he agafat FCKeditor (vegeu Apèndix A.2.4 i [18]).

Inicialment, volia mostrar el text de només lectura en HTML pla i el text que l'usuari pot modificar en l'editor, amagant un tipus o altre de finestra per tal de no omplir la pàgina. El problema rau, en que quan l'editor s'amagava, es bloquejava, impedint l'edició de cap més text.

Així doncs, vaig optar per presentar tot tipus de text a la mateixa finestra, la de l'editor. La única restricció que vaig haver d'imposar era la impossibilitat de poder utilitzar la funció de guardar quan el text només era per llegir, o fins i tot quan no s'hi havia carregat cap text.

Capítol 6

Conclusions i Treball futur

6.1 Conclusions

Aquesta aplicació ha arribat al punt que s'havia pactat a l'inici. No he aconseguit fer tot allò que volia, algunes d'aquestes coses ja estan descrites a la part de treball futur. No obstant, fa la seva funció.

El principal objectiu era l'edició col·laborativa de documents LaTeX, que, fins on he pogut investigar i conéixer, és una idea nova, ja que no hi ha res al mercat amb aquestes mateixes funcionalitats. Inicialment anava dirigida a ser una aplicació web normal. Posteriorment, i donada la popularitat que està agafant AJAX i les pàgines web més dinàmiques i interactives, es va canviar la direcció cap a una aplicació Web 2.0.

Això m'ha permés aprendre molt més sobre noves tecnologies -AJAX, JDOM, ...-, usos i abusos de noves formes de programar -un cop més AJAX, CSS, ...- i sobre tot m'ha fet adonar de la utilitat d'una bona previsió i anàlisi exhaustiu previs. No obstant, i donat que aquest ha estat un dels meus primers projectes seriosos, crec que no ha acabat malament després de tot.

L'aplicació ha estat provada satisfactòriament amb Mozilla Firefox i MS Internet Explorer. Amb altres navegadors amb els que s'ha provat s'han detectat certes incompatibilitats amb dos dels components principals: dhtmlXTree i FCKeditor.

No obstant, estic content d'aquest treball perqué hi ha una part de mi en ell :)

6.2 Treball futur

Com ja he dit en diversos punts, aquesta aplicació ben bé es podria ficar dins del marc de la Web 2.0. I per aquest mateix motiu podria estar en constant millora. No obstant, ara per ara hi ha certs aspectes que es podrien fer per tal d'ampliar les seves funcionalitats.

La interfície està tota en anglés i molts del missatges que l'usuari rep estan hardcoded -integrats al codi java-. Per tant, aquest seria un dels posibles primers punts a millorar,

treure tots els missatges de text a arxius de configuració. D'aquesta manera l'aplicació seria fàcilment internacionalitzable.

Un altre aspecte de la interfície es la manca d'informació a l'hora de realitzar certes accions. Per exemple, a l'hora d'esborrar un document i que no s'hagi fet, falta algun missatge que indiqui a l'usuari el motiu. Això s'hauria de millorar, usant més Javascript i AJAX, per poder rebre els missatges i mostrar-los dinàmicament a l'usuari. A més a més hi ha força parts que podrien millorar la seva presentació.

El control de concurrencia que evita la modificació dels arxius que defineixen el directori virtual de cada usuari -Doctree.xml- està realitzat de manera que cada usuari s'encarrega del seu propi xml. Aquesta solució, tal com està implementada implica que l'usuari ha de sortir i tornar a entrar de l'aplicació per tenir constància de certs canvis. Un possible millora d'això seria buscar la manera que aquest canvis també és fessin visibles en temps real -o quasi-. Alternativament, es podria descentralitzar amb un *pool de threads* o fins i tot que el servidor estigués distribuït. D'aquesta manera aconseguiriem que totes les màquines anessin més lleugeres de càrrega, sempre i quan aconseguissim un bon balanceig. També estaria bé canviar el sistema d'atorgació i denegació de privilegis, de tal manera que els usuaris rebessin notificacions i poguessin acceptar o rebutjar la col.laboració en document.

L'estructura dels documents es manté gràcies al format que tenen els identificadors de les parts. Això es podria fer de manera diferent per tal de no haver de recórrer la base de dades cada cop que volem recuperar l'estructura del document. Inicialment podríem pensar en un arxiu xml que mantingués aquesta estructura. Però un altre cop tindríem problemes d'actualitzacions concurrents. I un altre cop es podria solucionar amb un *pool de threads*.

Es podria implantar, a més de la tecnologia AJAX, una nova solució coneguda com Comet. Es basa en usar connexions HTTP de llarga durada per reduïr el temps de latencia en que els missatges són enviats al servidor. En essència, les aplicacions que utilitzen Comet no demanen peticions ocasionalment. En lloc d'això, el servidor té una línia de comunicació oberta que pot enviar data cap al client. Resumint-ho molt, una espècie de patró observador a la web. Això es aplicable al cas en que un usuari ésta en mode de lectura d'una part. No caldria fer peticions al servidor cada x segons, si no que el servidor enviaria una actualització del text en cas que hagués canviat a la base de dades. Es pot veure una comparativa a la figura 8.1.

Una altra cosa a millorar pot ser el parser de TeX. S'hauria de refer de manera que actués com ho fa un parser de xml. D'aquesta manera, definint callbacks podriem personalitzar les funcionalitats que volem que tingui, és a dir, com separar el text.

I per últim millorar el control d'errors i excepcions, que en aquest entrega és mínim.

En efecte, hi ha molts aspectes que poden millorar. I tots els projectes ho poden fer. Aquest concretament es troba en fase beta i conseqüentment en constant millora si surt com a producte.

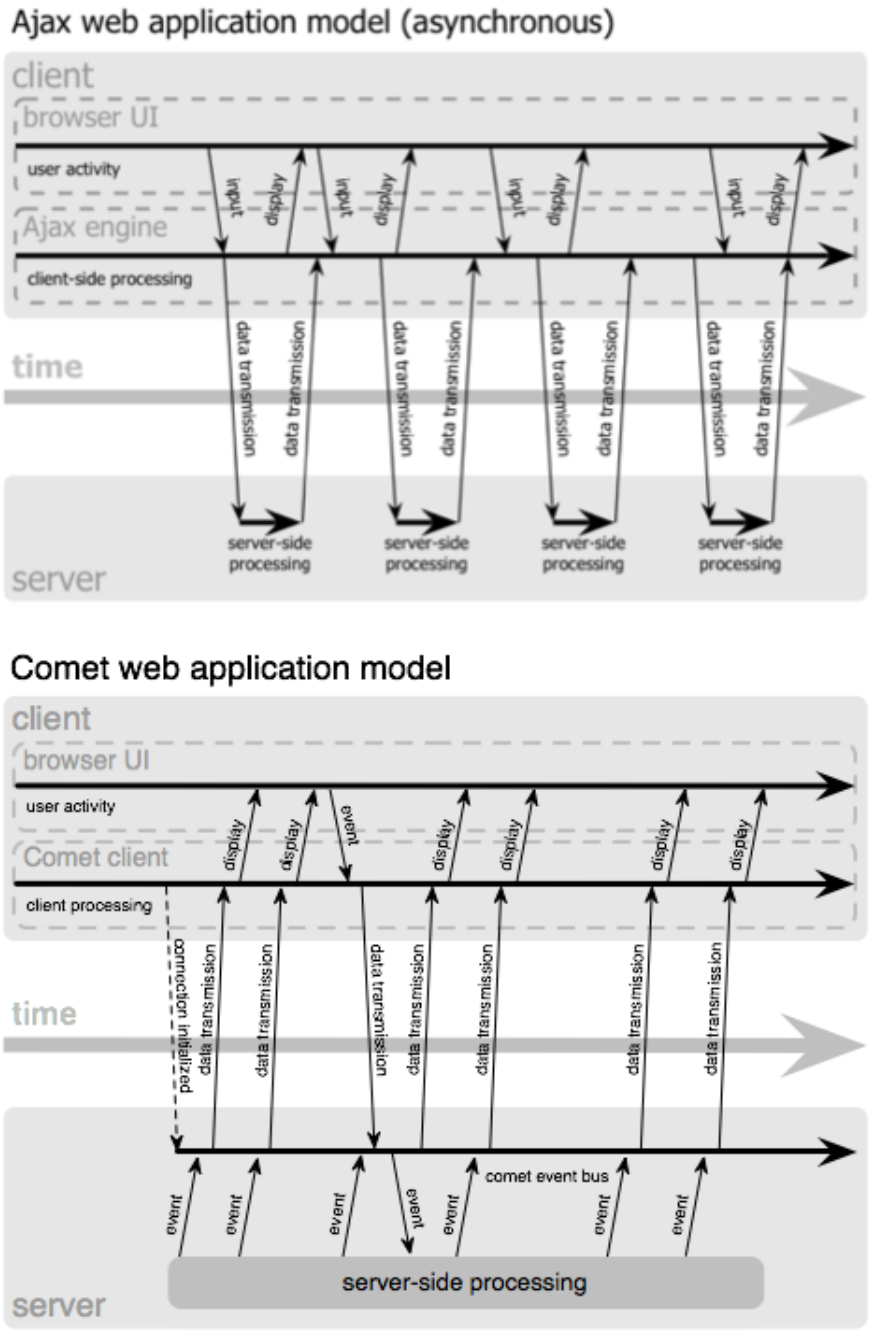

Figura 6.1: Comparació entre Ajax i Comet

Apèndix A

Tecnologies emprades

Durant l'execució d'aquest treball s'ha treballat amb diferents tecnologies *opensource* per aconseguir arribar a les fites proposades, així com programes, *toolkits* i *frameworks* per facilitar el desenvolupament i millorar la presentació així com la simplicitat en el codi.

En aquest capítol es mostren les diferents tecnologies, llenguatges i programes escollits per desenvolupar l'aplicatiu.

A.1 AJAX - Prototype

A.1.1 AJAX

Ajax -*Asynchronous Javascript and XML*- és una tècnica de desenvolupament web per crear aplicacions web interactives. L'objectiu és fer que les pàgines web tinguin millor resposta, intercanviant petites quantitat d'informació amb el servidor sense haver d'actualitzar la pàgina cada cop que el usuari fa un canvi. Això implica un increment de la interactivitat, velocitat i usabilitat de la pàgina web.

Aquesta tècnica usa una combinació de:

- XHTML -o HTML- i CSS.
- DOM accedit mitjançany un llenguatge d'script de banda del client, especialment implementacions de ECMAScript com JavaScript i JScript.
- L'objecte XMLHttpRequest per intercanviar data de forma asíncrona amb el servidor web.
- XML és usat a vegades com a format per la transfeerència de dades entre el servidor i el client, tot i que qualsevol format pot funcionar, incloent HTML, text pla, JSON i inclús EBML. Aquests fitxers han de ser creats dinàmicament per un servidor.

La càrrega de continguts dinàmics de forma asíncrona es porta fent des de que Internet Explorer va integrar l'element IFRAME. No obstant no va ser fins al 1998 que Microsoft va crear l'objecte XMLHttpReader -com a objecte ActiveX-. Llavors Mozilla va implementar una versió del mateix integrada al seu navegador, i conseqüentment Netscape també.

Pros

- Utilització de l'ample de banda
  Generant html localment en el navegador i portant només les dades actuals i les crides JavaScript, les pàgines web Ajax semblen carregar més ràpidament, ja que la càrrega és menor. Així, carregant només la pàgina inicial i repoblar la interfície amb noves dades en successives crides d'Ajax aconseguim una càrrega més ràpida dels continguts.

- Interactivitat
  Les aplicacions Ajax són executades comunment a la màquina de l'usuari, manipulant la pàgina actual del navegador utilitzant mètodes DOM. Ajax es pot usar per múltiples tasques sense el requeriment d'omplenar una pàgina HTML completa cada cop que un canvi es realitza. Generalment només petites peticions són enviades al servidor, i respostes relativament petites són retornades.Això permet el desenvolupament de més aplicacions interactives caracteritzant interfícies amb millor resposta degut a l'ús de tècniques DHTML.

Contres

- Usabilitat
  Les aplicacions web que utilitzen Ajax trenquen la esperada utilitat del botó "enrera"del navegador. Els usuaris esperen que clicant en tal botó, el navegador els torni a la última pàgina que es va carregar, i en les aplicacions Ajax aquest no acostuma a ser el cas.
  Un altre problem es que les actualitzacions dinàmiques de la pàgina web fan difícil a l'usuari afegir com favorit un estat particular de l'aplicació

- El temps de resposta importa
  La latència de la xarxa -l'interval entre una petició i la conseqüent resposta- necessita ser considert amb compte en el desenvolupament amb Ajax. Sense una resposta en temps real a l'usuari, la precàrrega de dades i un apropiat control de l'objecte XMLHttpRequest, els usuaris notaran un retard en la interfície de l'aplicació, una cosa que els usuaris no esperaran o no entendran. L'ús d'elements visuals que indiquin a l'usuari l'activitat que s'esta portant en background és recomanable per aquests problemes de latència.

A.1.2 Prototype

El Framework Prototype és un framework de JavaScript que aporta un framework Ajax entre altres utilitats. Disponible com una biblioteca autònoma, Ruby on Rails l'integra, així com altres projectes com Script.aculo.us o Rico. És compatible amb els majors navegadors actuals i va ser creat al 2005 per Sam Stephenson.

Posseeix 3 funcions que s'encarreguen de la funcionalitat Ajax:

- Ajax.Request(url, method, pars, options): Fa una petició a la url que es declari i espera una resposta xml. El tractament de la qual es fa en funcions que declarem a les opcions. Veguem un exemple:
  Posem que tenim un *servlet* que ens retornat el següent llistat:

```
<?xml version="1.0" encoding="utf-8" ?>
<ajax-response>
<response type="object" id="productDetails">
<monthly-sales>
<employee-sales>
<employee-id>1234</employee-id>
<year-month>1998-01</year-month>
<sales>$8,115.36</sales>
</employee-sales>
<employee-sales>
<employee-id>1234</employee-id>
<year-month>1998-02</year-month>
<sales>$11,147.51</sales>
</employee-sales>
</monthly-sales>
</response>
</ajax-response>
```

  Recuperar aquest XML és molt simple:

```
<script>
function searchSales()
{
var empID = $F('lstEmployees');
var y = $F('lstYears');
var url = 'http://yourserver/app/get_sales';
var pars = 'empID=' + empID + '&year=' + y;

var myAjax = new Ajax.Request( url,
```

```
{ method: 'get', parameters: pars, onComplete: showResponse });

}

function showResponse(originalRequest)
{
//put returned XML in the textarea
$('result').value = originalRequest.responseText;
}
</script>

<select id="lstEmployees" size="10" onchange="searchSales()">
<option value="5">Buchanan, Steven</option>
<option value="8">Callahan, Laura</option>
<option value="1">Davolio, Nancy</option>
</select>
<select id="lstYears" size="3" onchange="searchSales()">
<option selected="selected" value="1996">1996</option>
<option value="1997">1997</option>
<option value="1998">1998</option>
</select>
<br><textarea id=result cols=60 rows=10 ></textarea>
```

En aquest cas, només s'ha inclós el text XML dins de la textarea, però es podria hacer fet un tractament sencer ja que podem accedir a l'arbre DOM generat amb la resposta XML.[1]

- Ajax.Updater(container, url, method, pars, options): De la mateixa manera que l'anterior, peró aquí espera codi HTML amb el que plenarà el container determinat. Per exemple,

```
<script>
 function getHTML()
 {
   var url = 'http://yourserver/app/getSomeHTML';
   var pars = 'someParameter=ABC';
   var myAjax = new Ajax.Updater('placeholder',
url, {method:'get', parameters:pars});
  }
```

[1]Les funcions $('x') i $F('x') retornen l'element amb identificador "x", la primera i el valor d'aquest element, la segona. També formen part del framework Prototype.

```
</script>

<input type=button value=GetHtml onclick="getHTML()">
<div id="placeholder"></div>
```

- Ajax.PeriodicalUpdater(container, url, method, pars, options): Crides repetides a Ajax.Updater().

A.2 JavaScript i DHTML - dhtmlXTree i FCKEditor

A.2.1 JavaScript

JavaScript és el nom de la implementació del ECMAScript per part de la Netscape Communications Corporation, un llenguatge de programació d'scripting basat en el concepte de prototipus i una sintaxi basada en C. Aquest llenguatge és ben conegut pel seu ús a pàgines web, peró també es utilitzat per permetre l'accés d'scripting a objectes incrustat en altres aplicacions.

Va ser originalment desenvolupat per Brendan Eich sota el nom Mocha, canviat a LiveScript i posteriorment a JavaScript.
Un dels majors usos de JavaScript a la web és escriure funcions incrustades o incloses a pàgines HTML que interactúen amb el DOM de la pàgina per realitzar tasques impossibles amb HTML.

A.2.2 DHTML

Dynamic HTML o DHTML és un terme usat per una col.lecció de tecnologies, usades conjuntament per crear pàgines web interactives i animades usant una combinació de llenguatge estàtic de maruqes -com HTML-, un llenguatge d'scripting que s'executi al client -com JavaScript-, el llenguatge de definició de presentació -CSS-, i el DOM. Els scripts DHTML tendeixen a no funcionar correctament en totes les plataformes. Noves tècniques ,com Ajax i altres, tenen similars resultats peró en d'una manera accessible i complint els estàndards.

Algunes desventatges del DHTML són la dificultat de desenvolupament i debugació degut als diferents nivells de suport entre navegadors.

A.2.3 dhtmlXTree

Aquest component permet crear estructures jeràrquiques amb una bona API en JavaScript amb suport d'Ajax. És compatible amb els majors navegadors actuals i té una llicència open source -GPL-.

Té força característiques, de les quals destaco:

- Suport Multinavegador/Multiplatforma.
- Multiselecció.
- Control total amb l'API JavaScript.
- Càrrega dinàmica.
- Suport XML.
- *Drag-&-Drop*.

A.2.4 FCKEditor

Un editor de text lleuger fet amb JavaScript amb moltes caracterítiques i una llicència open source -LGPL-.

- Compatibilitat multinavegador.
- Suport de CSS.
- Lleuger i ràpid.
- Integració amb ASP, ASP.NET, Java, ColdFusion, Perl, PHP, JavaScript i més.
- Per programadors, fàcil d'instal.lar i configurar.
- Per usuaris, fàcil d'usar.

A.3 XML

XML (eXtensible Markup Language) és un llenguatge d'etiquetes desenvolupat per la W3C (World Wide Web Consortium). Els principis de XML es remunten als anys 70 quan IBM va desenvolupar un llenguatge anomenat GML (General Markup Language), que va sorgir degut a la necessitat que tenia IBM d'emmagatzemar grans quantitats de dades. Després de la creació de GML, es va crear el SGML -una estandarditzación del GML-. L'any 1998 la W3C va començar amb el desenvolupament de XML per solucionar els problemes que hi havien amb el HTML, ja que aquest no complia amb tots els estàndards.

Els objectius amb el que es va crear el XML són aquests:

- Concís desde el punt de vista de les dades i la manera de guardar-les.
- Extensible.
- Fàcil d'editar i escriure.

- Fàcil d'implantar, programar i aplicar a diferents sistemes.

L'estructura d'un document XML és similar als documents HTML, peró en els XML, es separa la presentació del contingut. S'utilitzen etiquetes per determinar de quin conjunt de dades es tracta i la definició d'aquestes etiquetes es deixa en mans del usuari que crea el fitxer XML.

Els documents XML són ben formats quan compleixen totes les definicions bàsiques de format i poden ser analitzats per qualsevol parser. Per altra banda un document XML és vàlid quan el document és ben format i, a més a més, la seva estructura es correspon amb la definida en un fitxer extern com pot ser un dtd o un XML Schema.

Les característiques d'un document XML són les següents:

- Només pot haver-hi un element arrel.
- Els valors dels atributs han d'estar delimitats per cometes.
- XML és case-sensitive.
- És necessari assignar noms als elements de l'estructura.
- Els elements sense contingut no tenen perquè tenir l'element de tancament. Pot ser tancat amb un '/'al final de l'element.

Un exemple de fitxer XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<tree id="0">
<item text="/" id="root" im0="folderClosed.gif" im1="folderOpen.gif"
im2="folderClosed.gif" open="1" select="1">
<userdata>root</userdata>
<item text="Own TeXs" id="own_texs" im0="folderClosed.gif" im1="folderOpen.gif"
im2="folderClosed.gif">
<userdata>directory</userdata>
</item>
<item text="Collaborated TeXs" id="collaborated_texs" im0="folderClosed.gif"
im1="folderOpen.gif" im2="folderClosed.gif">
<userdata>directory</userdata>
</item>
</item>
</tree>
```

A.4 JDOM - XPath

JDOM respon a les sigles angleses de Java Document Object Model i va ser creada per Brett McLaughlin and Jason Hunter l'any 2000. Aquesta llibreria defineix una API que té per objectiu parsejar, manipular i serialitzar documents xml amb Java d'una manera més simple que les proporcionades de manera estàndard amb SAX o DOM.

Per comprendre el motiu pel qual es va crear una API per manipular documents xml en Java com en aquest cas és JDOM, es poden revisar les següents consideracions:

1. Independència de llenguatge. DOM va ser dissenyat sense basar-se en cap llenguatge concret. Aquesta neutralitat implica que les característiques específiques de cada llenguatge no poden ser utilitzades.

2. Jerarquies estrictes: Amb DOM tot té una jerarquia. Per exemple, tant Element com Attribut o Document implementen la interfície Node.

3. Treball amb interfícies: L'API DOM consisteix en una sèrie de interfícies per fer recorreguts o manipular l'arbre.

Amb aquest tres punts en ment, es veu que DOM proporciona una API molt general i alhora complexa pel que fa a la seva adaptació per als programadors Java. L'aparició de JDOM soluciona els tres punts anteriors:

1. JDOM és específic per Java: JDOM treballa amb el llenguatge Java en tots els seus aspectes, amb la qual cosa proporciona un ambient agradable i intuïtiu pels programadors Java que desitgin manipular fitxers XML.

2. No hi ha jerarquies: Al contrari que amb DOM, amb JDOM no existeixen jerarquies. Un element és un element i un atribut és un atribut amb la qual cosa el programador no s'ha de preocupar per distingir de quin tipus es cada node com es feia amb DOM.

3. Treball amb classes: Amb JDOM es treballa amb classes en lloc d'interfícies.

D'aquesta manera, es poden crear elements d'una manera més simple amb l'operador new en lloc de fer ús de factories com a DOM.

XPath és un llenguatge recomanat per la W3C per fer cerques en fitxers xml. Està implementat en la majoria de llenguatges de programació i es troba en els diferents packages XML. La manera de treballar amb XPath consisteix en utilitzar una notació de camí per arribar a l'element o elements desitjats. Per exemple, si dins de l'anterior exemple de fitxer XML, volem buscar un item, estigui on estigui, l'identificador del qual es "demonworld", s'utilitzaria //item[@id='demonworld']. Aquest camí ens retornaria l'item -si existeix- que té per identificador "demonworld".

A.5 JNDI

JNDI és una API especificada en tecnologia Java que proveeix de funcionalitats de noms i directoris a aplicacions escrites amb Java. Està dissenyada especialment per la plataforma Java utilitzant el model d'objecte de Java. Usant JNDI, les aplicacions Java poden guardar i recuperar objectes Java de qualsevol tipus. Adicionalment, JNDI proporciona mètodes per fer operacions estàndards de directori, així com associar atributs a objectes i buscar objectes mitjançant els seus atributs.

JNDI també està definit independentment de cap implentació de serveis de noms o directoris específica. Proporciona a les aplicacions la possibilitat d'accedir a diferents i possiblement múltiples serveis de noms i directoris utilitzant una API comuna.
Això ens permet no incloure al codi, per exemple, la direcció remota de la base de dades a la que ens hem de connectar. D'aquesta manera es pot canviar sense haver de recompilar el codi.

Tomcat ja inclou aquesta funcionalitat mitjançant l'arxiu context.xml del directori META-INF de l'aplicació:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/LaTeXEDbetafinal">
<Resource auth="Container" driverClassName="com.mysql.jdbc.Driver"
   maxActive="10" maxIdle="5" name="jdbc/latexed" password=""
   type="javax.sql.DataSource" url="jdbc:mysql://nebuchadnezzar/latexed2"
   username="al0n3"/>
</Context>
```

A.6 JSP

Java porporciona una tecnologia que permet desenvolupar de manera senzilla aplicacions web, combinant-se amb la potència que proporciona el llenguatge de programació Java. La tecnologia JSP permet construir aplicacions web amb contingut dinàmic generat al servidor.

L'especificació de JSP és una extensió de la especificació dels *servlets* i va sorgir per la necessitat de desenvolupar contingut web dinàmic d'una manera més simple que amb els *servlets*.
El problema que hi havia en l'utilització de *servlets* per dissenyar pàgines dinàmiques era la inserció de codi HTML -el propi disseny de la pàgina- dins del codi Java. Això produïa una dificultat afegida dins l'equip de desenvolupament de l'aplicació web, ja que les responsabilitats d'un dissenyador eren diferents de les d'un programador. Tenint en compte aquest problema i per tal de solucionar-lo, es van idear les anomenades Java Server Pages, amb l'objectiu de proporcionar una manera de construir aplicacions web sense haver d'incrustar codi HTML dins dels *servlets*.

Amb les JSP es pot dissenyar una pàgina web com si d'una pàgina HTML es tractés, per posteriorment ser traduida a un *servlet* mitjançant un container com pot ser Apache Tomcat, Java System Application Server o JBoss entre altres. Un cop la pàgina JSP és traduïda a un *servlet* pel container aquest l'executa i dóna com a resultat una pàgina HTML que conté el codi inicial juntament amb el contingut generat dinàmicament. Quan el resultat és obtingut, aquest es retorna al client, qui només veurà HTML en el codi font de la pàgina.
JSP proporciona unes etiquetes especials que són:

- Directives: Ordres que s'executen abans de començar el processament de la pàgina JSP i poden modificar el resultat del mateix. Proporciona informació al container.

```
<%@ page language='java' contentType="text/html"
import='java.util.*' errorPage='error.jsp' %>
```

- Declaracions: Ens permet declarar variables, classes o mètodes que són usats en l'execució de la pàgina.

```
<%! int x = 1; %>
```

- Scripts: Els scripts representen els blocs de codi java dins de les pàgines JSP.

```
<% Codi java %>
```

- Expressió: Realitza l'avaluació d'una expressió i imprimeix el resultat. Noo requereixen d'una finalització amb ';' com és el cas de les declaracions.

```
<%= contador++ %>
```

A.7 Struts Framework

Struts és una eina de suport pel desenvolupament d'aplicacions Web sota el patró MVC i en la plataforma J2EE -Java 2, Enterprise Edition-. Struts formava part del projecte Jakarta de l'Apache Software Foundation, pero actualment es un projecte independent conegut com Apache Struts.

Struts permet reduïr el temps de desenvolupament. El seu caràcter de software lliure i la seva compatibilitat amb totes les plataformes en que Java Enterprise està disponible,

el converteix en una eina altament recomanable.
Quan es programen aplicacions Web amb el patró MVC, sempre sorgeix el dubte d'utilitzar un únic controlador o diversos controladors. Si considerem usar un sol controlador on tenir tota la lógica en el mateixa lloc, ens trobem amb un greu problema, ja que el controlador es converteix en el es coneix com *fat controller*, és a dir, un controlador saturat de peticions.
Struts sorgeix com la solució a aquest problemaja que implementa un únic controlador -*Actionservlet*- que avalúa les peticions de l'usuari mitjançant un arxiu configurable -struts-config.xml-.

Components del model
Corresponen a la lógica del negoci amb el qual es comunica l'aplicació web. Normalment el model comprén accessos a bases de dades o sistemes que funcionen independentment de l'aplicació web.

Components de control
Són els encarregats de coordinar les activitats de l'aplicació, des de la recepció de dades de l'usuari, verificacions de forma i selecció d'un component del model a ser cridat. Per la seva part, els components del model envien al control els seus eventuals resultats o errors de manera que es pugui continuar amb altres pasos de l'aplicació.

Aquesta separació simplifica enormement l'escriptura tant de vistes com de components del model: les pàgines JSP no ha d'incloure control d'errors, mentre que els elements de control simplement decideixen el següent pas a seguir.

Entre les característiques de Struts es poden mencionar:

- Configuració del control centralitzada.
- Les interrelacions entre Accions i pàgina o altres accions s'especifiquen mitajançant taules XML en lloc de codificarles als programes o pàgines.
- Component d'aplicació, que són el mecanisme per compartir informació bidireccionalment entre l'usuari de l'aplicació i les acciones del model.
- Llibreries d'entitats per facilitar la majoria de les operacions que normalment realitzen les pàgines JSP.
- Struts conté eines per validar camps de plantilles sota diversos esquemes que van des de validacions locals a la pàgina (JavaScript) fins a validacions fetes a nivell de les accions.

En definitiva, Struts permet que el programador es centri en el disseny d'alicacions complexes com una sèrie de simple components del model i de la vista intercomunicats per un control centralitzat. Dissenyant d'aquesta manera s'obté una aplicació més consistent i més fàcil de mantindre.

A.8 Apache Tomcat

Tomcat és un módul del servidor d'aplicacions Apache, més concretament Tomcat és un contenidor de *servlets* que implementa les especificacions dels *servlets* i JSP desenvolupat pel projecte Jakarta -un conjunt de solucions Java dins de la fundació Apache-. Aquest container esta desenvolupat en un ambient opensource i es pot trobar amb llicència Apache Software License. El codi de Tomcat va ser donat per Sun a la fundació Apache l'any 1999.

La funció que fa Tomcat és la de rebre peticions i executar *Servlets* -o en el cas d'una pàgina JSP, que es tradueix a un *servlet* prèviament-.

L'estructura del container és la següent:

- Bin. En aquesta carpeta es troben els fitxers necessaris per arrancar o aturar el container.
- Conf. Es troben els fitxers de configuració necessaris per poder utilitzar Tomcat. El fitxer més important és server.xml, que conte informació de configuració de Tomcat com, per exemple, la configuració de contextes.
- Logs. Es troben tots els arxius .log que es van creant i actualitzant mentre el container està funcionant.
- Webapp. Aquí s'emmagatzemen les aplicacions web própiament.
- Common. En aquesta carpeta es troben les classes necessàries per poder dur a terme les operacions bàsiques i comuns que porta a terme Tomcat.

La manera més senzilla d'instal·lar una aplicació a Tomcat és mitjançant la creació d'un fitxer WAR que conté l'aplicació amb una estructura determinada. Tomcat s'encarregarà de desplegar-la automàticament.

A.9 JavaMail

L'API JavaMail és un conjunt d'APIs abstractes que modelitzen un sistema de correu. Proveeix un framework independent de plataforma i protocol per construir aplicacions clients de correu electrónic basades en la tecnologia Java. Facilita la lectura i enviament de emails. Els proveïdors de serveis imlementen els protocols particulars, i mentre que alguns d'ells estan inclosos a l'API de JavaMail -POP3, IMAP i SMTP-, altres estan disponibles per separat. Es un paquet opcional que pot ser usat en JDK 1.4 i posterior en qualsevol sistema operatiu, encara que és requerit com a part de Java EE.

A.10 FileUpload

El package Commons FileUpload forma part del project Jakarta i permet afegir la capacitat de pujar fitxers, de manera fàcil, robusta i amb alt rendiment als nostres *servlets* i aplicacions web.

FileUpload parseja les peticions HTTP que s'adequen al RFC 1867, "Form-based File Upload in HTML". Així, si una petició HTTP és enviada usant el mètode POST, i amb un contingut de tipus "multipart/form-data", llavors FileUpload pot parsejar la petició i fer disponibles els resultats de manera fàcil.

Pot ser utilitzat de diferents maneres, depenent dels requeriments de l'aplicació. En el cas més simple, es pot cridar un simple mètode que parseji la petició i processi el llistat de ítem tal com l'aplicació necessita. Peró d'altra banda també podem decidir personalitzar FileUpload per obtenir màxim control en la manera en que els ítems individuals són guardats; com per exemple, guardar el continut en una base de dades partint d'un fluxe.

A.11 MySQL

Una base de dades és un conjunt de dades estructurades segons una sèrie de patrons. La necessitat d'accedir, modificar, inserir o eliminar aquestes dades implica haver de treballar amb un sistema gestor de base de dades, i això és el que ens proporciona MySQL.——MySQL és un sistema gestor de base de dades relacional, desenvolupatcom software lliure sota un llicenciament dual per l'empresa sueca MySQL AB, fundada el 1995 per David Axmark, Allan Larsson, y Michael Widenius.

Aquest gestor es va crear per la necessitat de connectar el gestor mSQL amb les taules propies de l'empresa MySQL AB. Després de diferents proves, van veure que mSQL no era el suficientment ràpid per les seves necessitats. Aquest fet va provocar l'adaptació de la vella interfície mSQL a les noves necessitats.

Al contrari de projectes com Apache, on el software és desenvolupat per una comunitat pública, y el copyright del codi està en mans de l'autor individual, MySQL està poseït y patrocinat per una empresa privada, que poseeix el copyright de la major part del codi. Això es el que possibilita l'esquema de llicenciament anteriorment mencionat.

MySQL està dissenyat de forma multifil i multiusuari. La seva facilitat d'ús així com la seva rapidesa fan de MySQL un dels gestors de base de dades més utilitzats en el món. Està escrit en ANSI C/C++ i funciona en diferentes plataformes.

Apèndix B

Manual d'ús

La pantalla inicial (figura B.1) ens dona la possibilitat d'entrar en l'aplicació si ja estem registrats o de registrar-nos.

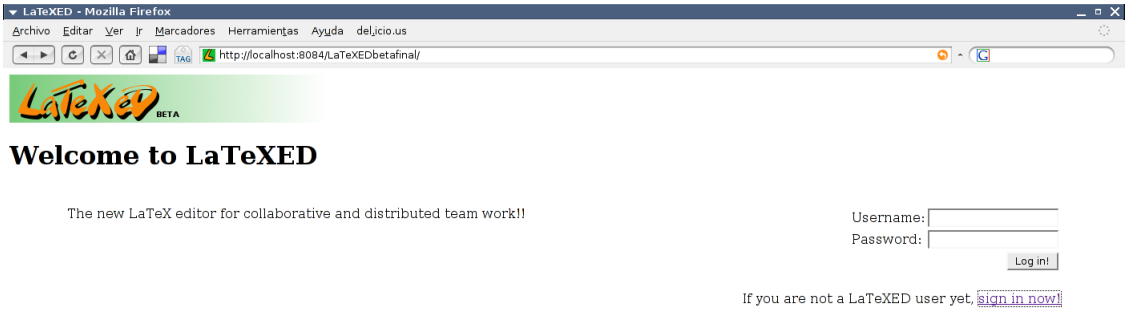

Figura B.1: Pantalla principal

En el registre (figura B.2) em d'introduïr el nom, el login -podem comprovar la disponibilitat del mateix-, el password -de entre 4 i 15 caràcters-, i una direcció de correu electrònic.

Un cop entrem a l'aplicació tenim el que apareix la figura B.3.

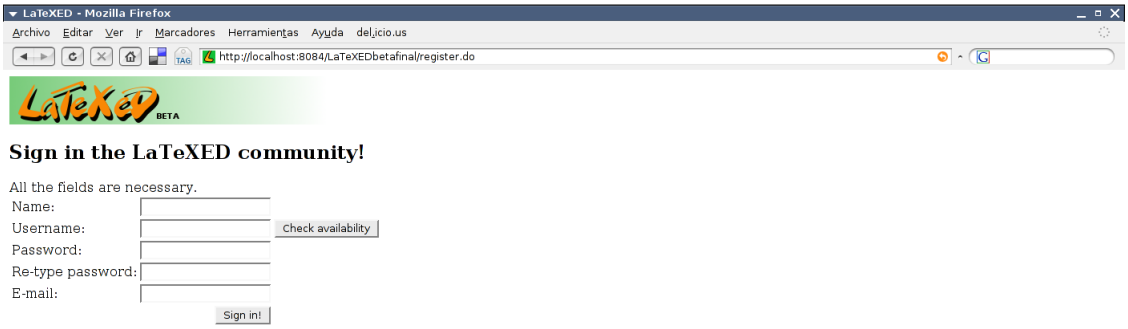

Figura B.2: Registre

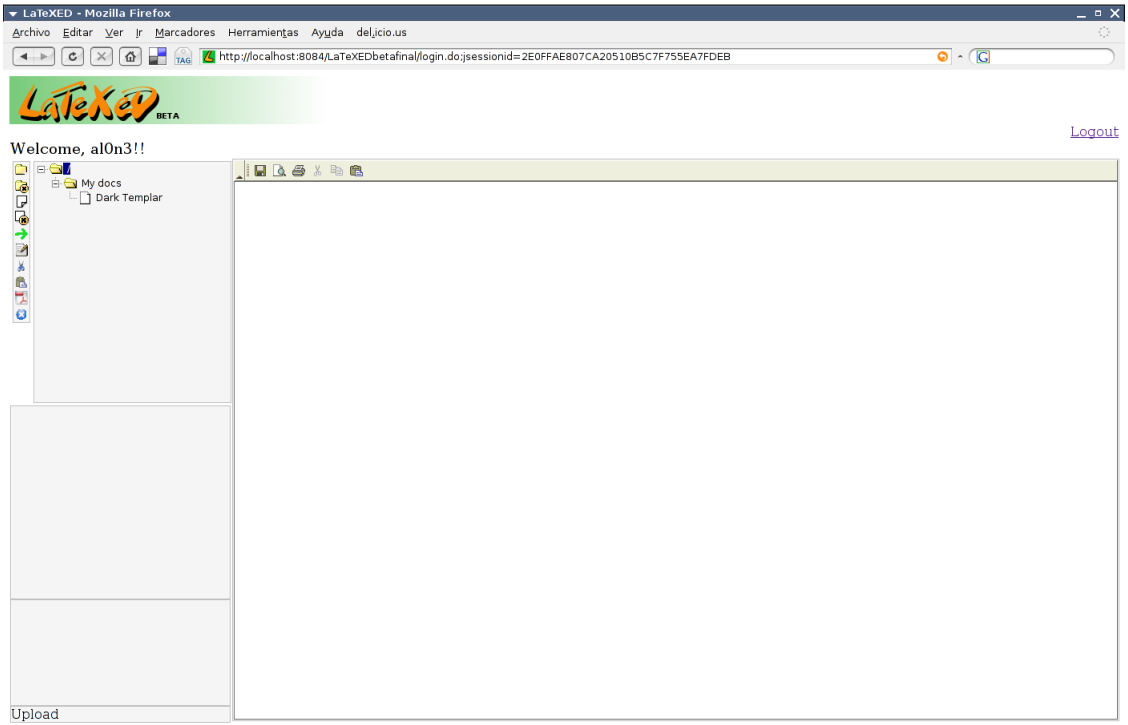

Figura B.3: Espai virtual

A la barra d'eines de l'esquerra de la pantall tenim totes les funcionalitats sobre l'arbre de directoris amb les quals es pot interactuar. De dalt a baix de la figura B.4:

Figura B.4: Barra d'eines

- **Crear una nova carpeta**. A l'hora de crear un carpeta se'ns demanarà un nom per aquesta.

- **Esborrar una carpeta**. Per tal d'esborrar un carpeta, ha d'estar buida.
- **Crear un document**. Ens demanarà un títol, la llista d'usuaris que tindrà privilegis d'escriptura i els usuaris de només lectura.
- **Esborrar un document**. Esborrarà un document del sistema.
- **Editar un document**. Carregarà a la pantalla noves seccions, com el selector de parts i el llistat de figures, per permetre la posterior edició del document.
- **Propietats del document**. Carregarà les propietats que se li van donar al document en el moment de crear-lo, per tal de poder modificar-les.
- **Retallar & Enganxar**. Permeten moure elements dins de l'arbre virtual.
- **Compilar PDF**. Compila el document i permet la descàrrega del PDF resultant.
- **Tancar el document actual**. Si tenim un document actiu, el tanca.

Quan seleccionem editar un document, es carreguen el selector de parts i el llistat de figures. En el selector de parts (figura B.5) podem seleccionar la part que volem editar -o llegir- i aquesta es carregarà en el editor de text -que fins ara no es veia-. En cas que la part seleccionada estigui bloquejada o nomes tinguem accés en mode lectura, l'editor impedirà que poguem guardar si fem qualsevol canvi.

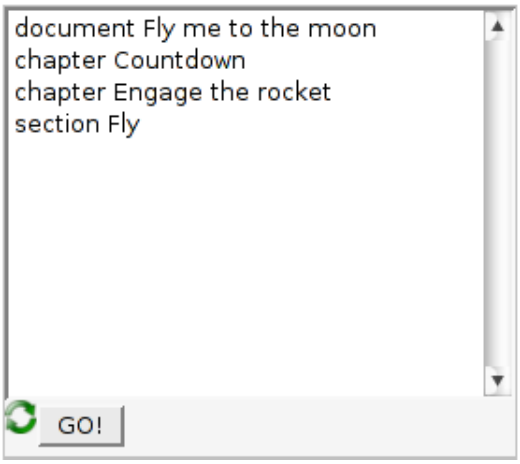

Figura B.5: Selector de parts

El llistat de figures(figura B.6) ens mostra les figures pujades i associades a aquest document. Aquestes han de ser en els següents formats: PNG, JPG o PDF. Per tal de pujar noves figures, només cal anar a la pestanya -per anomenar-ho d'alguna manera- Upload. Aquesta ens facilitarà un formulari per tal de pujar els arxius que volguem -un per

un-. A l'hora d'incloure una figura al document, nomes cal insertar la macro corresponent i el nom de la figura determinada, l'aplicació s'encarrega de que siguin incloses al document.

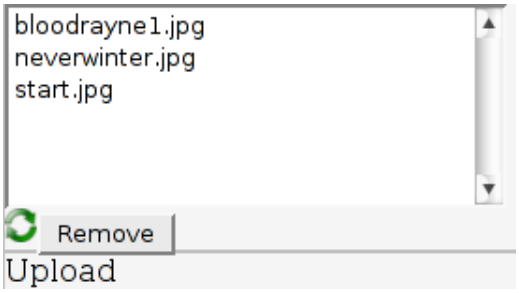

Figura B.6: Llistat de figures

L'editor(figura B.7) ens facilita l'edició de text -que gran veritat-.

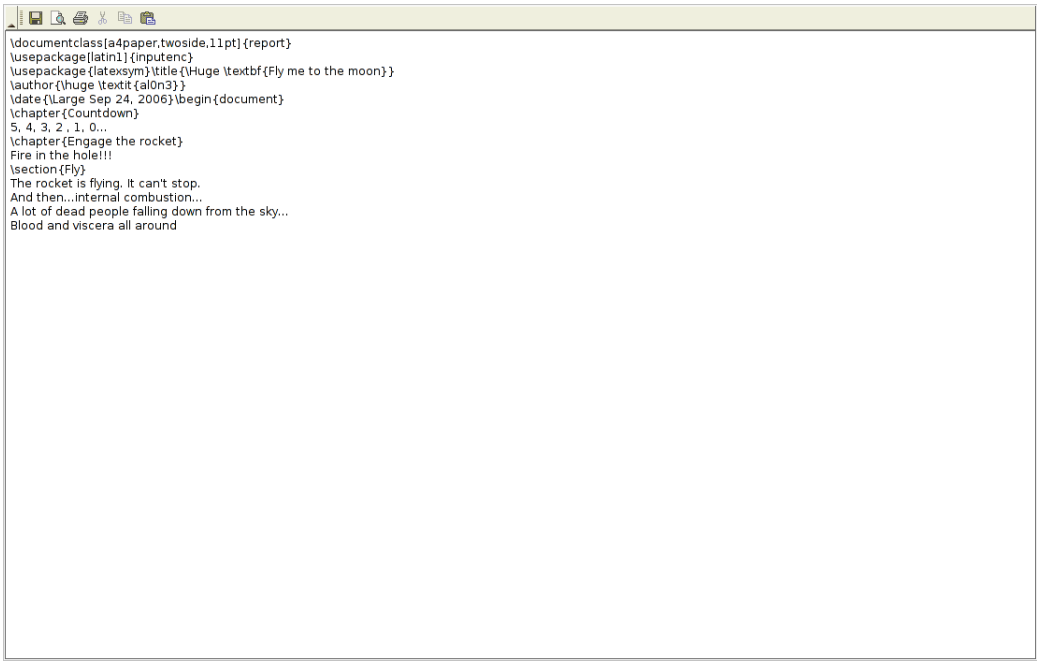

Figura B.7: Editor

Apèndix C

Manual d'instal·lació

C.1 Requeriments previs

Aquesta aplicació s'ha desenvolupat en un sistema GNU/Linux. Concretament en Ubuntu Dapper. Per aquest motiu, i donat que en el procés de creació dels documents pdf s'han d'executar certs scripts, es recomana l'ús sobre una distribució Linux qualsevol, sempre i quan compleixi els següents requeriments:

- JRE versió 5 o superior.
- Apache Tomcat 5.5.17 o superior (obligatori suport de Java 1.5).
- pdflatex i totes les seves dependències correctament instal·lades i si cal, configurades (es pot utilitzar un software diferent que faci la conversió de tex a pdf, sempre i quan es modifiquin els scripts pertinents).
- MySQL 5.0.19 (es possible utilitzar qualsevol altre base de dades, si aquesta té driver per Java i s'hi fan un parell de modificacions en els tipus de dades a l'arxiu que descriu l'estructura de la base de dades).

Les versions declarades han estat les utilitzades per desevolupar i provar l'aplicació, però no es descarta que pugui funcionar correctament en versions anteriors.

No hi ha suport sobre Windows, però es pot adaptar tot canviant el scripts utilitzats, si i només si es compleixen els mateixos requeriments.

C.2 Instal·lació

Com qualsevol altra aplicació web desenvolupada sobre JSP i un contenidor de *servlets*, només cal fer el desplegament de l'aplicació en el contenidor concret, en aquest cas, Apache Tomcat.

C.3 Configuració

Posteriorment al desplegament cal configurar uns parell de paràmetres que definiran on es guarda la informació generada pels usuaris. Aquests paràmetres són els següents:

- *UsersPath*: determina la ruta absoluta al disc dur on es guardaran les carpetes dels usuaris i els seus documents pdf. Aquesta ruta ha de tenir permisos d'escriptura per l'usuari que inicia el contenidor de *servlets*. Aquest paràmetre està situat a la secció *context-param* de l'arxiu *web.xml* situat a la carpeta WEB-INF de l'aplicació.

- *jdbc/latexed*: mitjançant JNDI podem utilitzar diferents tipus de recursos amb la mateixa manera d'accedir als continguts. Aquest paràmetre es troba a l'arxiu *context.xml* dins de la carpeta META-INF de l'aplicació i especifica el driver utilitzat i la url determinada per la connexió correcta a la base de dades. Aquest arxiu de configuració és típic de Tomcat, altres contenidors tindran altres mètodes de configuració.

Bibliografia

[1] Portal de Java elaborat per Sun Microsystems.
http://java.sun.com
Darrer accés: Setembre 2006

[2] Tutorial de Java
http://java.sun.com/docs/books/tutorial/index.html
Darrer accés: Setembre 2006

[3] API de J2SE 1.5.0
http://java.sun.com/j2se/1.5.0/docs/api/
Darrer accés: Setembre 2006

[4] JDOM
http://www.jdom.org
Darrer accés: Agost 2006

[5] API de JDOM i XPATH
http://www.jdom.org/docs/apidocs/index.html
Darrer accés: Agost 2006

[6] Tutorial XPath
http://www.w3schools.com/xpath/
Darrer accés: Agost 2006

[7] JavaMail
http://java.sun.com/products/javamail/
Darrer accés: Agost 2006

[8] API de JavaMail
http://java.sun.com/products/javamail/javadocs/index.html
Darrer accés: Agost 2006

[9] Jakarta FileUpload
http://jakarta.apache.org/commons/fileupload/
Darrer accés: Agost 2006

[10] API de Jakarta FileUpload
http://jakarta.apache.org/commons/fileupload/apidocs/index.html
Darrer accés: Agost 2006

[11] Apache Tomcat
http://tomcat.apache.org
Darrer accés: Agost 2006

[12] *servlet* API - Tomcat
http://tomcat.apache.org/tomcat-5.5-doc/servletapi/index.html
Darrer accés: Setembre 2006

[13] JSP API - Tomcat
http://tomcat.apache.org/tomcat-5.5-doc/jspapi/index.html
Darrer accés: Agost 2006

[14] Tutorial d'Apache Struts
http://www.programacion.com/java/tutorial/joa_struts/
Darrer accés: Agost 2006

[15] MySQL
http://http://www.mysql.org/
Darrer accés: Agost 2006

[16] Manual de MySQL
http://dev.mysql.com/doc/refman/5.0/en/
Darrer accés: Setembre 2006

[17] Documentació Prototype
http://www.sergiopereira.com/articles/prototype.js.htm
Darrer accés: Setembre 2006

[18] Documentació FCKEditor
http://wiki.fckeditor.net/
Darrer accés: Setembre 2006

[19] Documentació dhtmlXtree
http://scbr.com/docs/products/dhtmlxTree/ Darrer accés: Agost 2006

[20] Lamport, L: *A document preparation system.*
Addison-Wesley Professional. 2 edition. June, 1994.

[21] MediaWiki
http://www.mediawiki.org

[22] Zoho Writer
http://www.zohowriter.com/

[23] Writely
http://www.writely.com/

[24] SynchroEdit
http://www.synchroedit.com/

[25] Writeboard
http://writeboard.com/

[26] MoonEdit
http://www.moonedit.com/

[27] Ace
http://ace.iserver.ch/

[28] Gobby
http://darcs.0x539.de/