Títol: Recomanador de ponències a una conferència

Volum: 1/1
Alumne: Núria Bassols García

Director/Ponent: Ricard Gavaldà Mestre
Departament: LSI
Data: 03/01/2011

DADES DEL PROJECTE

Títol del Projecte:	Recomanador de ponències a una conferència
Nom de l'estudiant:	Núria Bassols García
Titulació:	Enginyeria Informàtica
Crèdits:	37.5
Director/Ponent:	Ricard Gavaldà Mestre
Departament:	LSI

MEMBRES DEL TRIBUNAL (nom i signatura)

President:	Jordi Delgado Pin
Vocal:	Jaume Soler Villanueva
Secretari:	Ricard Gavaldà Mestre

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

Índex

1 Introducció **11**

1.1 Origen del projecte 11
1.2 Raó i oportunitat 11
1.3 Objectius generals del projecte 12

2 Definició del projecte **13**

2.1 Abast del projecte 13
2.2 Desenvolupament dels objectius 14
2.3 Usuaris del sistema 14
2.4 Anàlisi dels Requisits 14
2.4.1 Requisits funcionals 15
2.4.2 Requisits no funcionals 16
2.5 Planificació 17
2.5.1 Fases del projecte i metodologia 17
2.5.2 Planificació temporal 18
2.5.3 Estimació de costos 20

3 Conceptes preliminars **22**

3.1 Sistemes de recomanació 22
3.1.1 Bases dels sistemes de recomanació 22
3.1.2 Plantejament segons tècnica de correspondència 23
3.1.3 Plantejament segons sistema de filtració 24
3.2 Recuperació de la informació 25
3.2.1 Procés de recuperació de la informació 26
3.2.2 Models de recuperació de la informació 29
3.2.3 Avaluació en la recuperació de la informació 32

4 Especificació **34**

4.1 Model conceptual de dades 34
4.1.1 Diagrama conceptual 35
4.2 Model de casos d'ús 36
4.2.1 Diagrama de casos d'ús 36
4.2.2 Especificació dels casos d'ús 39

5 Tecnologies escollides **40**

5.1 Aplicació web 40

5.2 Llenguatge de programació principal . . . 41
5.3 Servidor web . . . 42
5.4 Base de dades . . . 42
5.5 Apache Lucene . . . 42
5.6 Capa de presentació . . . 43
5.6.1 JSP . . . 43
5.6.2 HTML . . . 43
5.6.3 JavaScript . . . 43
5.6.4 AJAX . . . 44
5.7 *Frameworks* . . . 44
5.7.1 *Web development framework* . . . 45
5.7.2 *Persistence framework* . . . 45
5.7.3 *Web-page layout and decoration framework* . . . 45

6 Disseny del sistema recomanador **47**
6.1 Definició de les entrades . . . 47
6.1.1 Domini . . . 47
6.1.2 Entorn . . . 47
6.1.3 Dades de l'usuari i interaccions . . . 48
6.2 Plantejament del sistema de recomanació . . . 48
6.2.1 Tècnica de correspondència . . . 48
6.2.2 Tècnica de filtració . . . 49
6.2.3 Tècniques de recuperació de la informació . . . 49
6.2.4 Fonts de dades externes . . . 52

7 Disseny del software **54**
7.1 Patrons arquitectònics . . . 54
7.1.1 Arquitectura en tres capes . . . 54
7.1.2 Orientació a objectes . . . 55
7.2 Patrons de disseny . . . 56
7.2.1 Model-Vista-Controlador(MVC) . . . 56
7.2.2 *Front Controller* (Controlador Façana) . . . 57
7.2.3 Patró iterador . . . 57
7.3 Esquema de disseny . . . 57
7.4 Disseny de la capa de domini . . . 59
7.5 Disseny de la capa de gestió de dades . . . 60
7.5.1 Disseny lògic de la base de dades . . . 60
7.5.2 Estratègia de gestió de la persistència . . . 61
7.6 Disseny de la capa de presentació . . . 62

8 Implementació **64**
8.1 Entorn de treball . . . 64
8.1.1 Eclipse . . . 64
8.1.2 Subversion . . . 64
8.1.3 Firebug . . . 65
8.1.4 Luke . . . 65

8.1.5 Log4j . . . 66
8.2 Detalls d'implementació . . . 66
8.2.1 Indexació i cerca amb Lucene . . . 66
8.2.2 Extracció de text amb Tika . . . 68
8.2.3 Xifrat de la contrasenya d'usuari . . . 69
8.2.4 Validacions . . . 70
8.2.5 Localització . . . 70
8.2.6 Components de la interfície gràfica . . . 71
8.2.7 Problemes amb els navegadors . . . 73

9 Proves i experiments **74**
9.1 Especificació del pla de proves . . . 74
9.1.1 Proves generals del software . . . 74
9.1.2 Proves específiques del sistema de recomanació . . . 74

10 Documentació **76**

11 Conclusió **78**
11.1 Objectius assolits . . . 78
11.2 Revisió de la planificació . . . 79
11.3 Possibles ampliacions i millores . . . 79
11.3.1 Recomanacions . . . 79
11.3.2 Seguretat . . . 80
11.3.3 Temps de resposta . . . 80
11.3.4 Usabilitat . . . 81
11.3.5 Configuració . . . 81
11.3.6 Altres . . . 82
11.4 Conclusions personals . . . 82

Bibliografia **83**

Annex A Especificació **86**
A.1 Model de casos d'ús . . . 86
A.1.1 Accés a l'aplicació . . . 86
A.1.2 Administració de conferències . . . 88
A.1.3 Consulta de conferències . . . 92
A.1.4 Edició del perfil . . . 94
A.1.5 Recomanador . . . 95
A.1.6 Ajuda . . . 95

Annex B Implementació **96**
B.1 Detalls d'implementació . . . 96
B.1.1 Indexació amb Lucene . . . 96
B.1.2 Cerca amb Lucene . . . 100

Annex C Proves **103**
C.1 Dades utilitzades i entorn . . . 103

C.2 Anàlisi en l'indexació de presentacions 103
C.2.1 Proves amb diferents analitzadors 103
C.2.2 Visualització dels continguts de l'índex 107
C.3 Duració del procés d'indexació 110
C.4 Obtenció de dades d'un usuari en recursos externs 111
C.5 Consulta generada per a un usuari 114
C.6 Recomanacions obtingudes 116

Índex de figures

3.1 Entrades i sortides d'un sistema de recomanació 22
3.2 Exemple d'anàlisi basat en els elements 24
3.3 Exemple d'anàlisi basat en els usuaris 24
3.4 Gràfic *precision-recall* 33

4.1 Diagrama conceptual 36

5.1 Esquema del funcionament de SiteMesh 46

6.1 Entrades i sortides d'un sistema de recomanació 47

7.1 Arquitectura en tres capes 55
7.2 Patró MVC implementat per Struts 2 56
7.3 Esquema de disseny 58
7.4 Diagrama de classes del domini 60
7.5 Model de dades 61
7.6 Estructura de les pàgines 63

8.1 Procés d'indexació amb Lucene 67
8.2 Procés de cerca amb Lucene 68
8.3 Exemple de formulari amb missatges d'error de validació 70
8.4 Diàleg modal 71
8.5 Selector de dates 71
8.6 Editor de tags 72
8.7 Panell d'edició de recursos externs amb funcionalitat de cerca asíncrona i paginació. 72
8.8 Exemple de diàleg amb missatge d'alerta 72
8.9 Exemple de missatge informatiu 73

10.1 Índex de la documentació 76
10.2 Documentació del codi. Llista de classes 77
10.3 Documentació del codi. Referència d'una classe determinada 77

B.1 Procés d'indexació amb Lucene 96
B.2 Principals classes de Lucene per a indexar 97
B.3 Procés de cerca amb Lucene 101
B.4 Relació entre les classes principals de cerca 101

C.1 Termes indexats més freqüents del camp `title` 107

C.2 Termes indexats més freqüents del camp `abstract` 108
C.3 Termes indexats més freqüents del camp `fullArticle` 108
C.4 Termes indexats més freqüents del camp `title` 109
C.5 Termes indexats més freqüents del camp `abstract` 110
C.6 Termes indexats més freqüents del camp `fullArticle` 110
C.7 DBLP 112

Índex de taules

2.1 Estimació temporal 19
2.2 Estimació temporal de tasques addicionals PFC 20
2.3 Estimació de costos 21

3.1 Exemples de regles de substitució de l'algoritme de *Porter* 28
3.2 Exemple de l'estructura bàsica d'un índex invertit 29

11.1 Revisió de costos 79

C.1 Termes obtinguts de les publicacions de l'autor Ricard Gavaldà, la seva freqüència i el *boost* final assignat. 115
C.2 Termes obtinguts de les publicacions dels coautors del Ricard Gavaldà, la seva freqüència i el *boost* final assignat. 116

Capítol 1

Introducció

1.1 Origen del projecte

Aquest projecte va ser idea del professor Ricard Gavaldà (director del projecte), i el vaig escollir ja que em va semblar una proposta de Projecte de Final de Carrera interessant on podria aplicar molts dels coneixements adquirits al llarg de la carrera.

La idea d'aquesta proposta va sorgir mentre el director del projecte participava en la conferència informàtica European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (2009), on se li va ocórrer que podria ser interessant que hi hagués algun sistema de recomanació de presentacions d'una conferència (especialment en les conferències grans, amb sessions paral·leles, etc.) que ajudés als assistents a trobar les que més li interessen.

1.2 Raó i oportunitat

Contínuament ens enfrontem a situacions en les que tenim moltes més opcions de les que podem escollir realment i això implica haver de prendre decisions. "Quin llibre em podria comprar? Quina pel·lícula val la pena anar a veure? On podria anar de vacances? Quines pàgines web em resultarien interessants?"

És fàcil ampliar aquesta llista amb altres exemples en els que la gent ha de prendre decisions sobre com volen invertir els seus diners o el seu temps. Aquestes decisions sovint es basen en valoracions que fem a partir de recomanacions d'altres persones, informació obtinguda a partir de ressenyes, guies, etc. Però aquestes estratègies no sempre són útils. D'una banda, pot ser que no tinguem els mateixos gustos que altres persones, de manera que la informació que proporcionin pot no ser del nostre interès. D'altre banda, passar hores buscant informació sobre totes les possibles opcions pot crear més confusió que decisions ràpides i bones, donat el gran volum d'informació disponible. En definitiva, donar consells bons i útils no és fàcil.

És per això que hi ha un creixent interès per els sistemes de recomanació, que són capaços de processar tot aquest volum d'informació i fer-ne una tria per arribar a allò que volem d'una manera més simple i precisa.

Actualment s'utilitzen solucions basades en sistemes de recomanació en diversos dominis com pel·lícules, llibres, música, productes electrònics, notícies i pàgines web entre d'altres.

Els sistemes recomanadors han tingut èxit en molts i diversos àmbits, però no es té constància que n'existeixi cap dins del marc de les conferències, cosa que pot suposar una oportunitat per introduir l'idea d'un nou producte.

1.3 Objectius generals del projecte

A grans trets, l'objectiu d'aquest projecte consisteix en desenvolupar des de zero una aplicació que contingui un sistema recomanador de presentacions per a assistents d'una conferència. Aquest sistema haurà de ser capaç de processar tota la informació de la conferència i fer una tria automàtica que permeti que l'assistent trobi allò que li interessa de manera més ràpida i eficaç.

No obstant, l'objectiu d'aquest projecte no serà donar un recomanador extraordinàriament bo ja que no és viable. Això no és una ciència exacta, sinó que es podria estar perfeccionant constantment en base a l'experiència obtinguda dels resultats i per mitjà de l'aprofundiment en coneixements d'aquesta àrea d'investigació. Per aquest motiu, es donarà l'èmfasi en aconseguir un software que, sí, tingui un recomanador mínimament bo, però que quedi obert a que es pugui seguir desenvolupant en un futur per millorar-lo. En canvi, sí que tindrà prioritat que el software tingui la resta de funcionalitats de gestió i consulta, necessàries per a poder-lo començar a utilitzar i experimentar amb ell.

Capítol 2

Definició del projecte

En aquest capítol es marquen les directrius generals del projecte definint el seu abast, llistant els requisits funcionals i no funcionals que haurà de complir el sistema, i fent una planificació temporal i dels costos del projecte.

2.1 Abast del projecte

Tal com s'ha definit en la introducció, l'objectiu d'aquest projecte consisteix en desenvolupar una aplicació que contingui un sistema recomanador de presentacions per a assistents d'una conferència.

Desenvolupar software no consisteix només en programar, sinó que comporta una sèrie de disciplines associades a l'enginyeria del software que van des de la captura de requisits, fins a l'especificació, l'estudi i tria de tecnologies, el disseny del software, la implementació, la realització de proves i la documentació.

En aquest projecte, a més, cal incloure el disseny d'un algoritme recomanador que tampoc és una feina trivial. Per tal d'aconseguir un bon recomanador cal fer un estudi de la informació que es té, la que es pot obtenir i com es vol tractar per fer les recomanacions. A més cal fer moltes proves reals i estudiar el grau d'encert en cada cas. Amb els resultats obtinguts caldria estudiar com refinar el procés per tal de millorar els resultats i repetir les proves successivament.

Donat que es tracta d'un Projecte de Final de Carrera es té, en certa manera, un temps limitat per a presentar-lo per tal de no sobrepassar excessivament les hores que cal dedicar al PFC segons indica la Facultat d'Informàtica de Barcelona. Per aquest motiu ens centrarem en una part i la resta quedarà com a plans de futur o millores que podrà seguir l'autora del PFC com a projecte personal o es podran reprendre per altres projectistes interessats.

Ens centrarem en desenvolupar un software complet que permeti la gestió i consulta de conferències que contindrà un prototip de recomanador. Amb això es tindrà una plataforma completa funcionant, que ja podrien començar a utilitzar els usuaris finals, sobre la qual es podran realitzar proves addicionals en un futur per tal de millorar les recomanacions.

2.2 Desenvolupament dels objectius

Per tal de poder fer les recomanacions s'utilitzarà la següent informació:

- Informació de la conferència. Títols, autors i, opcionalment, resums i textos complets de les presentacions introduïdes pels organitzadors.
- Informació de l'assistent. D'una banda, dades sobre publicacions recents de l'usuari que fa la consulta, que el sistema buscarà de forma implícita en repositoris com ara DBLP, Google Scholar, l'ACM Digital Library, etc. D'altra banda, també es donarà la opció de que l'usuari pugui introduir explícitament altres dades que permetran millorar la definició del seu perfil.

Per tant, l'aplicació ha de permetre que es pugui introduir i gestionar informació de la conferència, així com oferir la possibilitat de que l'assistent introdueixi informació del seu perfil.

D'altre banda, aprofitant que es tenen les dades, també és interessant que es pugui consultar la informació de la conferència manualment així com el seu programa.

Un cop analitzats i desenvolupats els objectius, podem concloure que el sistema ha de permetre:

- Introduir i gestionar les dades d'una conferència.
- Veure informació de la conferència així com el seu programa.
- Introduir informació del perfil de l'assistent.
- Demanar recomanacions.

2.3 Usuaris del sistema

Els usuaris, també anomenats actors, són les persones que interactuen amb el sistema. En aquest projecte podem distingir dos tipus d'usuari:

- Organitzador de la conferència. És l'encarregat de desplegar l'aplicació i, un cop en marxa, administrar els continguts d'una conferència.
- Assistent a una conferència o altres usuaris que puguin estar interessats en entrar a l'aplicació per obtenir recomanacions personalitzades de les presentacions o, simplement, consultar el programa de la conferència.

2.4 Anàlisi dels Requisits

En aquest apartat es llisten els requisits funcionals i no funcionals que haurà de complir el sistema.

Per a obtenir els requisits, en primer lloc es va dur a terme una reunió amb el client per parlar de les característiques que hauria de tenir el sistema, partint de l'objectiu definit. A partir de les conclusions tretes, el client va redactar una primera versió dels requisits. Després d'analitzar aquesta primera versió, es van fer petites rectificacions i es va donar per vàlida com a punt de partida pel procés de desenvolupament.

2.4.1 Requisits funcionals

Accés a l'aplicació

- Un usuari anònim s'ha de poder registrar en l'aplicació.
- Un usuari registrat ha de poder identificar-se per entrar a l'aplicació (iniciar sessió).
- Un organitzador ha de poder identificar-se per entrar a l'aplicació amb permisos d'administrador (iniciar sessió).
- Un usuari o organitzador identificat en el sistema ha de poder tancar sessió.
- Un usuari ha de poder seleccionar la conferència que vol veure (o administrar, si es tracta d'un administrador).

Administració de conferències

- Un organitzador ha de poder afegir, editar i eliminar una conferència
- Un organitzador ha de poder afegir, editar i eliminar sessions d'una conferència.
- Un organitzador ha de poder afegir, editar i eliminar presentacions d'una conferència.
- Un organitzador ha de poder afegir, editar i eliminar categories de presentació d'una conferència.
- Un organitzador ha de poder canviar la visibilitat d'una conferència: publicar o passar-la a privada.

S'ha definit que una conferència estarà dividida en sessions i que cada sessió agruparà un conjunt de presentacions en un mateix dia. Habitualment les xerrades d'una sessió tenen temàtica similar, però el programa no farà aquesta suposició.

Una presentació vindrà definida per: un títol, un autor o autors, la sessió a la que pertany, l'hora en la que comença i una categoria de presentació. Les categories de presentació es podran personalitzar en cada conferència (p.ex. plenària o invitada, presentació tècnica, poster, workshop, demos...). Opcionalment, també es podrà introduir un resum, paraules clau i adjuntar el fitxer amb el text complet (amb diferents formats).

Aquesta estructura s'ha triat de tal forma que sigui prou flexible com per ajustar-se a la majoria de les conferències.

D'altre banda, també s'ha establert que, en l'aplicació, una conferència pugui tenir visibilitat pública o privada. Mentre estigui marcada com a privada només podra accedir-hi l'administrador, així la resta d'usuaris només veuran els seus continguts quan l'administrador la publiqui.

Consulta de conferències

- Un usuari ha de poder consultar la informació bàsica d'una conferència pública.
- Un usuari ha de poder consultar la llista de sessions d'una conferència pública.

- Un usuari ha de poder consultar la informació detallada d'una sessió d'una conferència pública.
- Un usuari ha de poder consultar la llista de presentacions d'una conferència pública.
- Un usuari ha de poder consultar la informació detallada d'una presentació d'una conferència pública.
- Un usuari ha de poder consultar el programa complet d'una conferència pública.

Edició del perfil

- Un usuari registrat ha de poder editar el seu nom i cognoms.
- Un usuari registrat ha de poder editar la seva contrasenya.
- Un usuari registrat ha de poder identificar-se en un repositori d'autors i/o publicacions determinat.
- Un usuari registrat ha de poder introduir paraules clau sobre temes del seu interès.

Recomanador

- Un usuari registrat ha de poder obtenir suggeriments personalitzats de presentacions.

Ajuda

- Un usuari ha de poder visualitzar un manual d'ajuda per a l'utilització del sistema.
- Un administrador ha de poder visualitzar un manual d'ajuda per a l'administració.

2.4.2 Requisits no funcionals

Usabilitat

- L'aplicació ha de ser molt fàcil d'utilitzar, especialment per la part dels assistents. Altrament, si després de 30 segons l'usuari no ha vist com utilitzar l'aplicació, la deixarà córrer.

Instal·lació i configuració

- La configuració (instal·lació o setup) per part d'un organitzador ha de ser relativament senzilla, o si no molt ben explicada pas a pas per a usuaris no necessàriament molt experts en informàtica.

Idioma

- L'idioma de l'aplicació hauria de ser l'anglès. Es requereix que la interfície sigui en anglès ja que es vol utilitzar en conferències internacionals.

Eficiència

- El preprocés de dades pot ser lent, però la resposta a l'usuari assistent ha de ser ràpida (menys d'un minut).

Extensibilitat

- Es vol que l'aplicació sigui extensible i millorable per part d'altres desenvolupadors. En concret, l'aplicació haurà de ser modular i l'idioma utilitzat en el codi (noms de variables, classes, mètodes, comentaris, etc) així com en la documentació interna haurà de ser l'anglès.

Modularitat

- És important que el disseny de l'aplicació sigui ben pensat per recollir les abstraccions importants i susceptibles de ser reutilitzades (com els mòduls que processen informació externa, mòdul recomanador, etc.).

Política de llicències software

- L'aplicació ha d'estar desenvolupada amb programari lliure.
- L'aplicació quedarà disponible sota una llicència de codi lliure.

2.5 Planificació

2.5.1 Fases del projecte i metodologia

El projecte s'ha dividit en les següents fases:

Definició del projecte. Aquesta fase inclou totes les tasques de gestió inicial de qualsevol projecte que inclou l'estudi de la primera proposta del client, la definició dels objectius del projecte, el recull de requisits del sistema que es vol desenvolupar i la planificació del projecte.

Formació i aprenentatge en sistemes de recomanació. S'ha inclòs aquesta fase per tal de formar-nos sobre temes específics de l'àmbit del projecte, és a dir, sobre els sistemes de recomanació. Concretament, s'investigaran els mecanismes que es poden emprar en els sistemes de recomanació i també s'estudiaran tècniques de l'àmbit de la recuperació de la informació. Aquests coneixements seran necessaris per a poder dissenyar el sistema recomanador del projecte.

Especificació. En aquesta fase es definiran més detalladament les funcionalitats del sistema i es determinarà quin serà el seu comportament davant de l'usuari sense tenir en compte les tecnologies que s'utilitzaran. També inclou la documentació típica d'aquesta etapa com el diagrama conceptual i el diagrama de casos d'ús.

Disseny. En aquesta etapa es farà un estudi de les possibles tecnologies que es podrien utilitzar en el projecte i s'escolliran les que es creguin més adients. Partint d'aquestes decisions, es definirà el sistema amb el suficient detall per a la seva implementació. També es dissenyarà el sistema de recomanació utilitzant els coneixements apresos en l'estudi de conceptes preliminars.

Implementació En aquesta etapa es duu a terme la implementació del sistema dissenyat en les fases anteriors, utilitzant les tecnologies seleccionades. També inclou la documentació del codi.

Proves. En aquesta etapa es duran a terme proves sobre el funcionament general de l'aplicació així com proves més específiques del sistema recomanador.

Documentació. En cadascuna de les etapes anteriors s'haurà generat la documentació corresponent a l'etapa. Aquesta etapa només inclou la creació de guies per a l'usuari final.

2.5.2 Planificació temporal

Al no tenir un client final i una data d'entrega concreta, el temps per dur a terme aquest projecte és bastant lliure. No obstant s'ha fet una estimació del temps a invertir que es detalla en la figura 2.1. Per calcular els dies s'ha suposat una dedicació de 5 hores diàries, tot i que el temps disponible serà variable.

Fase	Temps (hores)	Temps (dies)
Definició del projecte	**40**	**8**
Estudi proposta client	10	2
Definició objectius	5	1
Anàlisi requisits	15	3
Planificació	10	2
Formació i aprenentatge en sistemes de recomanació	**50**	**10**
Especificació	**50**	**10**
Model conceptual	25	5
Casos d'ús	25	5
Disseny	**250**	**50**
Estudi tecnologies	50	10
Disseny sistema de recomanació	100	20
Disseny software	100	20
Implementació	**425**	**85**
Estructura general	50	10
Accés a l'aplicació	15	3
Administració de conferències	125	25
Perfil usuari	125	25
Recomanador	100	20
Ajuda	10	2
Proves	**50**	**10**
Documentació	**50**	**10**
TOTAL	**915**	**183**

Taula 2.1: Estimació temporal

Apart del temps dedicat a les fases del desenvolupament del projecte de software en si, també caldrà tenir en compte les tasques necessàries pel fet de ser un projecte de final de carrera. En la figura 2.2 es pot veure l'estimació d'aquestes tasques.

Fase	**Temps (hores)**	**Temps (dies)**
Informe previ	5	1
Memòria	100	20
Presentació	50	10
TOTAL	**155**	**31**

Taula 2.2: Estimació temporal de tasques addicionals PFC

Sumant les hores, ens dona un temps estimat de 1070 hores que, suposant que es puguin dedicar 5 hores diàries de dilluns a dissabte, són 214 dies estimats de treball. Si a aquests dies li sumem les vacances, els festius i altres dies que es sap des d'un principi que no es tindrà disponibilitat, es preveu una durada real del projecte d'uns 9 mesos i mig.

2.5.3 Estimació de costos

No hi haurà costos de software, ja que tot el que s'utilitzarà serà lliure i gratuït. En quant al hardware, només s'utilitzarà l'ordinador personal de l'autora del projecte, que no comptem.

Tot i que en la pràctica, el projecte ha estat desenvolupat per una sola persona, en aquesta secció es farà una estimació del cost del personal suposant que en el projecte hagués estat desenvolupat per un equip de treballadors amb els següents perfils:

- Director de projecte. S'encarregarà de definir objectius, realitzar la planificació i organitzar les tasques i treballadors del projecte.
- Analista. S'encarrega de dur a terme l'anàlisi de requeriments i l'especificació del sistema.
- Arquitecte. S'encarregarà de dissenyar el software a desenvolupar.
- Programador. Programarà totes les classes i mòduls que es necessitin.
- Dissenyador gràfic. S'encarregarà de la part gràfica de l'aplicació.

Fase	Perfil	Hores	Preu	Total
Definició del projecte	Director	25	35 €/hora	875 €
	Analista	15	25 €/hora	375 €
Formació i aprenentatge en sistemes de recomanació	Analista	20	25 €/hora	500 €
	Arquitecte	30	25 €/hora	750 €
Especificació	Analista	50	25 €/hora	1250 €
Disseny	Arquitecte	250	25 €/hora	6250 €
Implementació	Programador	410	15 €/hora	6150 €
	Dissenyador gràfic	15	15 €/hora	225 €
Proves	Programador	50	15 €/hora	750 €
Documentació	Programador	50	15 €/hora	750 €
TOTAL	-	**915**	-	**17875€**

Taula 2.3: Estimació de costos

Capítol 3

Conceptes preliminars

En aquest capítol s'estudia com funcionen els sistemes de recomanació, dedicant especial atenció a les tècniques de recuperació d'informació.

3.1 Sistemes de recomanació

Un sistema de recomanació és una aplicació que té com a objectiu donar suport a l'usuari a l'hora de prendre decisions davant d'una gran quantitat d'informació. Aquests sistemes formulen recomanacions que poden interessar a l'usuari basant-se en les preferències que ha expressat de forma explícita o implícita.

Els conceptes que es veuran en aquesta secció es basen, principalment, en conceptes extrets de la següent referència bibliogràfica: [1].

3.1.1 Bases dels sistemes de recomanació

Per introduir el funcionament d'un sistema de recomanació, es començarà considerant el sistema com una caixa negra de la que s'explicaran les entrades que rep i les sortides que produeix.

Un sistema de recomanació necessita l'entrada d'un conjunt de dades que aportin la màxima informació sobre les recomanacions que es volen obtenir. Tal com es mostra en la figura 3.1, aquestes entrades es poden agrupar en quatre blocs que es detallaran a continuació.

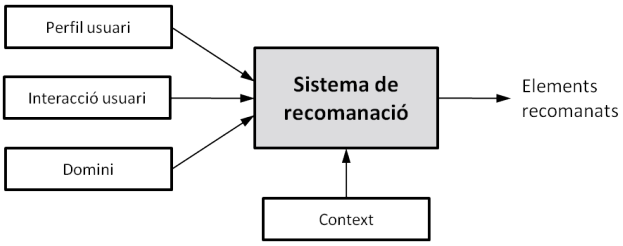


Figura 3.1: Entrades i sortides d'un sistema de recomanació

Perfil de l'usuari. És necessari conèixer el màxim possible d'informació referent a l'usuari per tal de poder-li proporcionar resultats adequats. Aquestes dades poden incloure des de dades socioeconòmiques fins a dades de caràcter personal.

Interaccions dels usuaris. Quan un usuari interacciona amb el sistema, proporciona una gran quantitat d'informació referent a les seves predileccions, ja sigui de forma implícita o explícita. Com a interaccions implícites podem entendre saber les pàgines que visita, les cerques que realitza, etc. D'altre banda, proporciona informació explícitament quan afegeix etiquetes a altres elements, vota, comenta, etc.

Informació del domini. Està compost per les dades sobre els diferents elements disponibles dins del domini en el qual treballa el sistema de recomanació. Sense informació sobre els elements susceptibles de ser recomanats, el sistema seria una eina sense valor. Per exemple, si es tractés d'un recomanador de llibres caldria tenir suficient informació sobre autors, editorials, títols, etc.

Context en el que s'utilitza el recomanador. Depenent del context en el que s'utilitzi el recomanador es pot voler acotar el conjunt de resultats aprofitant la informació de l'entorn. Així, el context pot aportar dades sobre el subconjunt d'elements que s'han de considerar.

A partir d'aquestes entrades, el sistema recomanador generarà un conjunt d'elements recomanats com a sortida.

Un cop estudiades les entrades i les sortides, ens introduïm en el funcionament intern dels sistemes de recomanació. Un dels punts més importants és triar un mètode de filtració d'informació que esculli la informació adequada per realitzar les recomanacions. També caldran establir tècniques de correspondència que ens permetin comparar un usuari o element amb altres usuaris o elements.

3.1.2 Plantejament segons tècnica de correspondència

A l'hora de desenvolupar sistemes de recomanació, existeixen dos enfocaments principals en funció de si el sistema busca correspondències entre els elements del domini o bé entre els usuaris.

Anàlisi basat en els elements.

En l'anàlisi basat en els elements (en anglès *item-based analysis*), el sistema computa la similitud entre els elements. Quan un usuari mostra interès per un element, el sistema li recomana elements relacionats amb aquest. Tal com es mostra en la figura 3.2, si els elements A i C son similars i a un usuari li agrada A, es pot recomanar l'element C a aquest usuari.

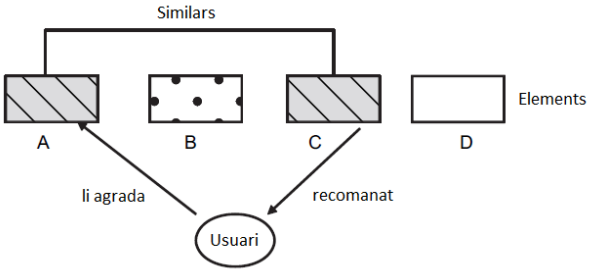


Figura 3.2: Exemple d'anàlisi basat en els elements

Per a trobar elements similars podem utilitzar tècniques basades en el contingut (per exemple, comparant el text associat a cada element) o tècniques col·laboratives (basant-se en com han interactuat els usuaris amb ells).

Anàlisi basat en els usuaris.

En l'anàlisi basat en els usuaris (en anglès *user-based analysis*), el sistema determina primer els usuaris que són similars. Tal com es mostra en la figura 3.3, si a un usuari li agrada l'element A, aquest element podrà ser recomanat a altres usuaris similars a aquest usuari.

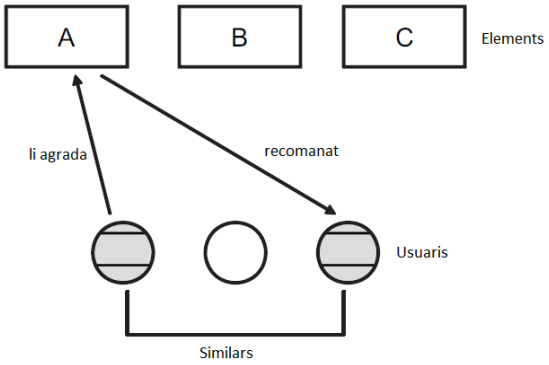


Figura 3.3: Exemple d'anàlisi basat en els usuaris

Els usuaris similars es poden obtenir utilitzant informació del perfil dels usuaris (sexe, edat, etc.) o per mitjà de tècniques col·laboratives, basant-se en les accions dels usuaris.

3.1.3 Plantejament segons sistema de filtració

Per tal trobar relacions entre els usuaris o elements, cal un llenguatge comú per a calcular la rellevància entre ells. Hi ha dos enfocaments principals segons si s'analitza text o accions dels usuaris.

Filtració basada en el contingut

Els sistemes que utilitzen l'anàlisi basat en el contingut (en anglès *content-based analysis*), analitzen el contingut associat a cada element o usuari i n'extreuen una representació, normalment en format vectorial, que posteriorment serà comparada amb altres per tal de determinar-ne els graus de similitud. Per exemple, es pot representar un llibre a partir d'un vector de termes i trobar llibres similars segons la similitud de termes que comparteixen. Utilitzant aquesta representació es poden comparar usuaris amb usuaris, usuaris amb elements i elements amb elements.

Filtració col·laborativa

Els sistemes basats en la filtració col·laborativa analitzen el conjunt d'elements amb els que un usuari ha interactuat a través del sistema per tal de predir els elements que puguin ser del seu interès. És a dir, un element es considera una caixa negra de la que no es mira el seu contingut, només les seves interaccions amb l'usuari.

Comparativa

A continuació es mostren alguns dels avantatges i inconvenients de les tècniques col·laboratives i les basades en el contingut.

- Les tècniques col·laboratives, a diferència de les basades en el contingut, permeten utilitzar el mateix motor de recomanació per a diferents idiomes i en diferents dominis, ja que tracten els elements com una caixa negra i no s'utilitza la informació sobre el contingut en si.
- En les tècniques basades en el contingut, l'algoritme no té noció de la qualitat de l'element ni la seva popularitat, només la similitud del seu contingut.
- En les tècniques col·laboratives, si no es té el suficient volum d'interaccions, el sistema recomanador pot tenir una pobre capacitat de predicció. Per exemple, el sistema pot no tenir suficient informació per determinar els usuaris similars d'un nou usuari que gairebé ha interactuat amb el sistema. De la mateixa manera, un nou element difícilment serà recomanat pel sistema fins que un nombre considerable d'usuaris hagin interactuat amb ell.

3.2 Recuperació de la informació

En aquest apartat es tracta de forma més detallada el tractament de text per mitjà de tècniques de recuperació de la informació. Els conceptes que es veuran segueixen, principalment, les següents referències bibliogràfiques: [11] i [8].

La recuperació de la informació és una àrea d'estudi que consisteix en trobar material (generalment documents) de naturalesa no estructurada[1] (generalment text), que satisfacin una determinada necessitat d'informació, dins de grans col·leccions.

[1]El concepte de "dades no estructurades" es refereix a les dades que no tenen una estructura clara i que, per tant, no són fàcilment interpretables per una màquina.

3.2.1 Procés de recuperació de la informació

La recuperació de la informació es pot dividir en les següents etapes:

Parsing En aquesta primera etapa s'analitzen els documents i s'extreu el text i la seva estructura. Els documents digitals generalment són *bytes* en un arxiu o en un servidor web. El primer pas del procés és convertir aquesta seqüència de *bytes* en una seqüència lineal de caràcters. Si es treballa amb diferents formats d'arxius, abans caldrà determinar el seu format per tal de determinar quin descodificador serà l'adequat per extreure el text.

Tokenization En aquesta etapa es divideix el text en trossos, generalment en paraules, anomenats *tokens*. A continuació es mostra un exemple:

Entrada: Friends, Romans, Countrymen, lend me your ears;
Sortida: | Friends | | Romans | | Countrymen | | lend | | me | | your | | ears |

La qüestió principal de l'etapa de la tokenització és saber com separar per tokens correctament. No obstant, a vegades es difícil definir que s'entén per una "paraula". Normalment els *tokenitzadors* es basen en heurístiques simples:

- Totes les cadenes de caràcters alfabètics contigus formen part d'un *token*, de la mateixa manera amb els números.
- Els *tokens* estan separats per espais en blanc, com un espai o salt de línia, o caràcters no alfanumèrics.

En idiomes com l'anglès, on les paraules estan delimitades per espais en blanc, aquest enfocament és senzill. No obstant això, la *tokenització* és més difícil per idiomes com el xinès que no tenen límits de les paraules. La *tokenització* simple basada en els espais en blanc també presenta dificultats a l'hora d'identificar conjunts de paraules que tenen un significat propi com "New York" que haurien de ser tractats com un sol token. Una possible solució per abordar aquest problema podria ser consultar taules de locucions o altres conjunts de paraules comuns per saber si trencar els termes o tractar-los conjuntament. En la tokenització també ens trobem en altres casos difícils depenent de l'idioma o de l'àmbit en el que ens trobem com per exemple:

- Diferents usos de l'apòstrof en anglès (possessiu i contracció).
- Paraules compostes separades per guions.
- Sigles, ja que en alguns casos es poden trobar les lletres separades per punts.
- Dates com per exemple "18/08/2009" o "Ago 18, 2009".
- Noms de llenguatges de programació com "C++" i "C#"
- Adreces de correu electrònic com "example@mail.com"
- Direccions web com "www.google.com"
- Direccions IP com "142.32.48.231"

Caldrà estudiar quina es la millor opció segons l'idioma o l'àmbit en el que ens trobem i el tipus d'informació que ens interessi.

Preprocessat lingüístic Un cop s'ha dividit el text en *tokens*, es duu a terme un preprocessat lingüístic per tal de trobar els tokens finals que seran indexats. Aquest preprocessat pot incloure les següents fases:

- Normalització
- Eliminació de paraules buides
- Lemmatització i *Stemming*
- *Enriching*

- Normalització Tenint els documents dividits en tokens (i també la consulta o *query*), caldrà comprovar la coincidència entre les llistes de tokens per determinar la seva rellevància. No obstant, cal tenir en compte que hi ha varis casos en els que dos seqüències de caràcters no són exactament iguals però es voldrien considerar que són coincidents. La normalització de tokens consisteix en canonitzar tokens per tal de trobar coincidències malgrat les diferències superficials.

Una estratègia comuna és passar tots els caràcters a minúscula. Normalment això és una bona idea ja que sinó les paraules de principi de frase es considerarien diferents només per tenir la primera lletra majúscula. No obstant, aquest mètode pot igualar tokens que s'haurien de tractar com a diferents. Molts noms propis provenen de noms comuns i, per tant, només es poden distingir segons si comencen per majúscules o no, com per exemple "Windows" i "windows". Si és amb la primera lletra majúscula es pot referir al sistema operatiu, enlloc de finestra en anglès, i quan es passa a minúscula es perd aquesta diferència. Es poden utilitzar heurístiques per intentar determinar quan passar a minúscula i quan no, però això pot no ser útil ja que els usuaris normalment fan les consultes en minúscula encara que siguin noms propis. Per tant, passar-ho tot a minúscula sol ser la solució més pràctica.

Es poden definir altres regles com per exemple eliminar punts o guions per formar un terme:

U.S.A, USA $\Rightarrow$ USA

anti-discriminatory, antidiscriminatory $\Rightarrow$ antidiscriminatory

El criteri més important a l'hora de definir com normalitzar és tenir en compte com els usuaris tendeixen a escriure aquestes paraules a l'hora de fer consultes i vindrà condicionat per com s'hagi establert la tokenització.

- Eliminar paraules buides Hi ha paraules molt freqüents que no solen aportar valor a l'hora de seleccionar documents que coincideixin amb una necessitat de l'usuari (com per exemple articles, preposicions, adverbis, etc.). Aquestes paraules s'anomenen paraules buides o *stop words* en anglès. Generalment sol interessar eliminar aquestes paraules buides ja que les estadístiques indiquen que pot suposar un 40% de la reducció de l'espai de l'index i, per tant, una reducció del temps de cerca. Una possible estratègia és tenir una llista de detenció que descarti els *tokens* que hi apareixin.

Moltes vegades no afecta descartar les paraules buides. No obstant, en algun tipus de consultes poden ser importants. Per exemple, alguns noms de cançons i versos de poesia estan totalment formats per paraules típicament considerades paraules buides (*Let It Be*, *To be or not to be*, ...).

- Lemmatització i *Stemming* En molts idiomes, les paraules poden aparèixer de varies formes segons la seva flexió (plurals, formes verbals, etc.). Per exemple el verb anglès 'to walk' es pot presentar de diverses formes:

walk, *walked*, *walks*, *walking* ...

A més, hi solen haver famílies de paraules derivades amb significats similars, com per exemple:

democracy, *democratic*, *democratization* ...

L'objectiu de la lemmatització i l'*stemming* és reduir les paraules a la seva arrel gramatical. D'aquesta manera les diferents variacions d'una paraula podran ser analitzades com un mateix *token*. *Stemming* generalment es refereix a un procés heurístic que talla seqüències de caràcters típics dels sufixes (i a vegades també prefixes) de les paraules amb l'esperança d'assolir aquest objectiu correctament la majoria de les vegades. La lemmatització, en canvi, utilitza diccionaris i analitza morfològicament les paraules per tal d'eliminar adequadament les desinències i tornar a la seva arrel o la forma que apareix en el diccionari.

L'algoritme més comú per l'stemming en anglès és el *Porter Stemmer*. L'algoritme de *Porter* es compon de cinc fases de reduccions de paraules que s'apliquen de forma seqüencial. Dins de cada fase es troben les diverses convencions per a triar les regles, com per exemple donat un grup de regles triar la que aplica al sufix més llarg. En la figura 3.1 es mostren algunes de les regles de substitució típiques.

Regla	**Exemple**
sses ⇒ ss	caresses ⇒ caress
ies ⇒ i	ponies ⇒ poni
ation ⇒ ate	hibernation ⇒ hibernate
full ⇒	successful ⇒ success

Taula 3.1: Exemples de regles de substitució de l'algoritme de *Porter*

Moltes de les regles utilitzen el concepte de 'mesura' d'una paraula, que calcula d'una forma aproximada el nombre de síl·labes per veure si una paraula es prou llarga per tal de considerar la part coincident a una regla com un sufix enlloc d'una part de l'arrel de la paraula. Per exemple, la regla:

(m >1) EMENT ⇒

substituiria *replacement* a *replac*, però no *cement* a *c*.

No obstant, en alguns casos aquestes tècniques poden empitjorar el rendiment. Per exemple amb el *Porter Stemmer* les paraules *operational*, *operating* i *operative* es convertirien en "oper" i es perdria precisió en algunes consultes com les següents, ja que en cada cas adopta un significat diferent:

operational research
operating system
operative dentistry

En aquest cas, la lemmatització no ho solucionaria del tot ja que determinades flexions s'utilitzen en col·locacions concretes. D'aquesta manera una frase amb les paraules *operate* i *system* no serà un bon resultat per la consulta *operating system*.

- *Enriching* Aquesta etapa té l'objectiu enriquir el contingut analitzat, per mitjà de diferents mètodes com per exemple donat un terme trobar sinònims, paraules relacionades, definicions, categories, etc.

Un cop tenim la llista final de *tokens*, podem passar a la següent etapa.

Indexing Per tal d'evitar haver d'analitzar linealment tots els textos per a cada consulta, s'indexen els continguts prèviament. Es crea un diccionari que conté tots els termes obtinguts en les etapes anteriors i, per a cada terme, una llista dels documents en els que apareix. D'aquesta manera es poden recuperar els documents que contenen un determinat terme de forma més eficient. Aquesta estructura s'anomena índex invertit.

Brutus	→	1	2	4	11	31	45	173	174
Caesar	→	1	2	4	5	6	16	57	132 ...
Calpurnia	→	2	31	54	101				

Taula 3.2: Exemple de l'estructura bàsica d'un índex invertit

En algunes variants, també es guarda la freqüència o les posicions del terme en cada document.

Querying Un cop s'ha fet tot el processat dels documents, es poden fer *queries* o consultes al sistema que representin unes determinades necessitats d'informació per tal d'obtenir un conjunt de resultats, cadascun amb el seu grau de rellevància.

Visualització Un cop es tenen els resultats de la consulta, s'ordenen (si es vol) per ordre del grau de rellevància, s'obté la informació necessària dels documents i es mostren els resultats de la forma desitjada.

3.2.2 Models de recuperació de la informació

Un model de recuperació de la informació defineix:

- Una vista lògica dels documents. *Quina informació es guardarà dels documents?*
- Un llenguatge de consulta. *Quin tipus de consultes es poden demanar?*
- Un concepte de "rellevància" o similitud. *Quan coincideix un document amb una consulta?*

N'hi ha varis models de recuperació de la informació. En els següents apartats veurem alguns dels més comuns.

Model booleà

El model booleà és un model de la recuperació de la informació en la que es pot plantejar qualsevol consulta que estigui en forma d'expressió booleana, és a dir, combinant termes amb els operadors

AND, OR i NOT. Aquest model veu cada document com el conjunt de paraules que conté, sense tenir en compte les seves freqüències.

Donat el conjunt de termes $\{t_1, t_2, ..., t_T\}$ es pot representar un document com un vector de bits de dimensió T:

$$d = (d_1, d_2, ..., d_T)$$

tal que

$d_i = 1$, si i només si t_i apareix en d
$d_i = 0$, si i només si t_i no apareix en d

D'aquesta manera, si tenim una *query* q que és la AND de varis termes, podem definir la seva similitud amb un document d de la següent forma:

$$similitud(d, q) = \text{“}d \text{ conté tots els termes de } q\text{”}$$

En aquest cas, el sistema buscarà en l'índex els documents que continguin cadascuna de les paraules i farà una intersecció dels conjunts obtinguts.

En aquest tipus de model les consultes són exactes, és a dir, un document és completament rellevant o completament irrellevant. No dona informació sobre el grau de rellevància dels resultats, només determina si compleix tota la consulta o no. El model booleà pot ser efectiu si busquem un document concret però sol ser massa limitat per a la majoria de necessitats d'informació dels usuaris i/o aplicacions.

Model vectorial

La idea d'aquest model és fer que un document pugui ser rellevant si coincideix amb alguns dels termes de la consulta, tenint amb compte el número d'aparicions dels termes en el document i donar una puntuació a cada document segons el grau de similitud que s'estima.

En el model vectorial, les consultes s'expressen com a text lliure que es tracta de la mateixa forma que el text dels documents. Tant els documents com les consultes es representen per mitjà de vectors que contenen els pesos relatius per a cadascun dels termes. Aquests vectors són T-dimensionals (R^T), sent T el nombre de termes diferents. D'aquesta manera podem representar el vector de pesos d'un document d de la següent manera.

$$w_d = (w_{d,1}, w_{d,2}, ..., w_{d,T})$$

on $w_{d,j}$ representa el pes del terme t_j en el document d. A diferència del model booleà, aquest valor no està limitat a valors binaris.

Una de les estratègies més comunes per assignar el valor dels pesos és utilitzar el model ***tf-idf***. Segons aquest model, el pes d'un terme t_j en un document d_i és pot definir com el producte entre dos càlculs: *term frequency* (tf) i *inverse document frequency* (idf).

$$w_{i,j} = tf_{i,j} \cdot idf_j$$

Term frequency indica la freqüència amb la que apareix un terme en un document. Normalment es normalitza aquest valor per tal de prevenir un biaix cap als documents mes llargs (que poden tenir una freqüència més gran independentment de la importància real d'aquest terme en el document).

$$tf_{i,j} = \frac{fi,j}{max_k \{f_{i,k}\}}$$

on $f_{i,j}$ és el nombre d'ocurrències del terme t_j en el document d_i.

Les paraules que apareixen sovint en un document poden ser més rellevants per a descriure el contingut. No obstant, no tots els termes aporten la mateixa informació. Donat un domini particular, algunes paraules són molt comunes en tots els documents i no són molt selectives. El factor *idf* pretén donar més valor als termes menys freqüents en una determinada col·lecció de documents. Si tenim que N és el número de documents i que n_j és el número de documents en els que apareix el terme t_j, el factor *idf* d'aquest terme es pot calcular de la següent manera:

$$idf_j = log\left(\frac{n}{n_j}\right)$$

D'aquesta manera, si un terme apareix en tots els documents el seu *idf* és $log(1)$ que és 0.

Un cop definida la representació de les consultes i els documents, passem a veure com calcular el grau de rellevància dels documents per a una determinada consulta. El model vectorial mesura la similitud entre el vector de la consulta i el vector del document per tal d'obtenir una puntuació numèrica que estimi el grau de rellevància. Normalment s'utilitza l'angle entre dos vectors com a mesura de divergència entre els vectors. Així, es considera que dos vectors són similars si apunten cap a la mateixa direcció. Una de les opcions més comunes és prendre el cosinus de l'angle com a valor numèric d'aquesta similitud, ja que té la propietat de valer 1 en vectors idèntics i 0 en vectors sense cap terme en comú.

L'expressió geomètrica del producte escalar entre dos vectors, ens permet obtenir el cosinus de l'angle.

$$a \cdot b = \|a\|\|b\|cos\beta = \sum_{i=1}^{n} a_i \cdot b_i$$

$$cos\beta = \frac{a \cdot b}{\|a\| \cdot \|b\|} = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}$$

Si els vectors són unitaris, llavors $\|a\| = \|b\| = 1$ i es pot simplificar la fórmula.

$$cos\beta = a \cdot b = \sum_{i=1}^{n} a_i \cdot b_i$$

D'aquesta manera, en el model vectorial, podem definir la similitud entre una consulta (q) i un document (d) tal com es mostra a continuació.

$$similitud(d,q) = w_d \cdot w_q = \sum_{t=0}^{T} w_{d,t} \cdot w_{q,t}$$

tal que w_d i w_q són vectors unitaris en un espai T-dimensional (R^T), sent T el nombre de termes diferents.

Donat que les consultes i els documents es representen de la mateixa forma, també podem utilitzar la mateixa mesura per a calcular la similitud entre dos documents.

El model vectorial té les següents avantatges respecte el model booleà estàndard:

- Els documents poden coincidir parcialment amb la consulta. Un document pot ser rellevant encara que no coincideixi amb tots els termes de la consulta.
- Els pesos dels termes no estan limitats a valors binaris.
- Permet calcular un grau de similitud entre consultes i documents (i també entre documents i documents).
- Permet ordenar els documents segons la seva rellevància per a una determinada consulta.

3.2.3 Avaluació en la recuperació de la informació

L'enfocament estàndard per a l'avaluació d'un sistema de recuperació de la informació es basa en la noció de documents rellevants i no rellevants, considerant un document rellevant aquell que respon a una necessitat d'informació donada.

Precision i *recall* són les dos mesures més freqüents i bàsiques per avaluar l'eficàcia d'un sistema de recomanació. *Recall* mesura l'eficàcia del sistema a l'hora de trobar els documents rellevants, mentre que la precisió mesura l'eficàcia a l'hora de filtrar els documents irrellevants. Cal tenir en compte que les tècniques que tendeixen a millorar el *recall* tendeixen a perjudicar la precisió, i a l'inrevés.

En primer lloc veurem aquestes mesures en el cas més simple en el que el sistema retorna un conjunt de resultats sense ordre ni grau de rellevància.

Precision és la fracció de documents recuperats pel sistema que es consideren rellevants.

$$Precision = \frac{|\{documents\ rellevants\} \cap \{documents\ recuperats\}|}{|\{documents\ recuperats\}|}$$

Recall és la fracció de documents rellevants que són recuperats pel sistema.

$$Recall = \frac{|\{documents\ rellevants\} \cap \{documents\ recuperats\}|}{|\{documents\ rellevants\}|}$$

En el cas de tenir consultes que retornen llistes ordenades en funció d'una puntuació, podem utilitzar les mesures precisió i *recall* per a subconjunts que contenen els i primers resultats.

D'aquesta manera, es pot generar una gràfica *precision-recall* posicionant el resultat obtingut per a aquests subconjunts, fent variar el valor de la i.

Es considera que la consulta perfecte conté tots els documents rellevants en les primeres posicions i després els no rellevants.

$$|\{rellevants\} \cap \{recuperats_{(i)}\}| = i, \text{ si } i < |rellevants|$$
$$|\{rellevants\} \cap \{recuperats_{(i)}\}| = |recuperats|, \text{ si } i \geq |rellevants|$$

De manera que per a qualsevol qualsevol valor del *recall*, la consulta perfecta tindria una precisió del 100%.

D'altra banda, es considera una consulta que retorna documents ordenats totalment a l'atzar. Si tenim que D és el total de documents:

$$|\{rellevants\} \cap \{recuperats_{(i)}\}| = i \cdot \frac{|rellevants|}{D}$$

$$Precision_{(i)} = \frac{|rellevants|}{D}$$

La corba de les gràfiques *precision-recall* típiques solen tenir la coneguda forma de dent de serra. Si l'element $k+1$ recuperat no és rellevant, el *recall* és el mateix que per als k primers documents, però disminueix la precisió. Si és rellevant, aleshores augmenta la precisió i el *recall*.

Figura 3.4: Gràfic *precision-recall*

Capítol 4

Especificació

Aquest capítol té la finalitat de definir més detalladament les funcionalitats del sistema i determinar quin serà el seu comportament davant de l'usuari.

Per a especificar el sistema s'ha utilitzat UML (Unified Modeling Language) [12] , un llenguatge gràfic estàndard per especificar, visualitzar i documentar els artefactes d'un sistema de programari. L'UML permet capturar l'estructura del sistema i el seu comportament d'una forma precisa.

4.1 Model conceptual de dades

El model conceptual ens permet representar els conceptes significatius en el domini del problema amb l'objectiu de facilitar la comprensió dels elements que intervenen en el problema. Mostra principalment classes d'objectes i els seus atributs, relacions entre classes d'objectes, i restriccions d'integritat per especificar tot allò que no es pot expressar gràficament.

Partint de la definició del projecte, extraurem els elements principals del domini del problema i en farem una descripció detallada que ens permetrà generar posteriorment el model conceptual.

Un dels conceptes principals és el de **conferència**. Una conferència es considera un esdeveniment en el que hi tenen lloc varies presentacions relacionades amb un camp o àmbit concret. Es representa per un nom i un codi i té una data d'inici i una de fi. Opcionalment també es pot especificar una descripció, la seva ubicació i la URL de la seva pàgina web. Cada conferència podrà estar en mode privat o públic, segons si poden veure els seus continguts només els administradors o també els usuaris externs.

Les conferències estan dividides en **sessions** i cada sessió agrupa un conjunt de presentacions, generalment de temàtica similar, que es duen a terme en un mateix dia. Les sessions comencen a una determinada hora i, opcionalment, es pot detallar la seva hora de fi, la seva ubicació i la seva descripció.

Cadascuna de les **presentacions** tindrà un títol i un autor o autors. A més de pertànyer a una determinada sessió, també es classificarà segons una categoria de presentació. Opcionalment també es podrà introduir un resum, paraules clau i adjuntar un fitxer amb el text complet. Del fitxer ens interessa la seva ruta i el seu tipus.

Les **categories de presentació** es podran personalitzar per a cada conferència. Per exemple, es podrien tenir presentacions plenàries o invitades, presentacions tècniques, *posters*, *workshops* i *demos* entre d'altres. Es definiran mitjançant un nom, una descripció i un color per tal de poder-los identificar visualment.

D'altra banda tenim els **usuaris** que utilitzen el sistema, que engloben els administradors i els usuaris externs. D'aquests usuaris caldrà enregistrar el seu identificador (correu electrònic) i la seva contrasenya. Pels usuaris externs, a més, també ens interessa el seu nom i cognoms, les seves paraules clau preferides, els autors de recursos externs amb els que s'identifica ell mateix i altres llistes de paraules extretes pel sistema amb les que es representa l'usuari (paraules de les seves publicacions, paraules de les publicacions dels seus coautors).

Els **autors** s'identifiquen a partir de la font de la que s'han extret i el seu identificador dins d'aquesta font. Cada autor també tindrà un nom amb el que es mostrarà a l'usuari.

4.1.1 Diagrama conceptual

En la figura 4.1 es mostra el diagrama de classes del model conceptual.

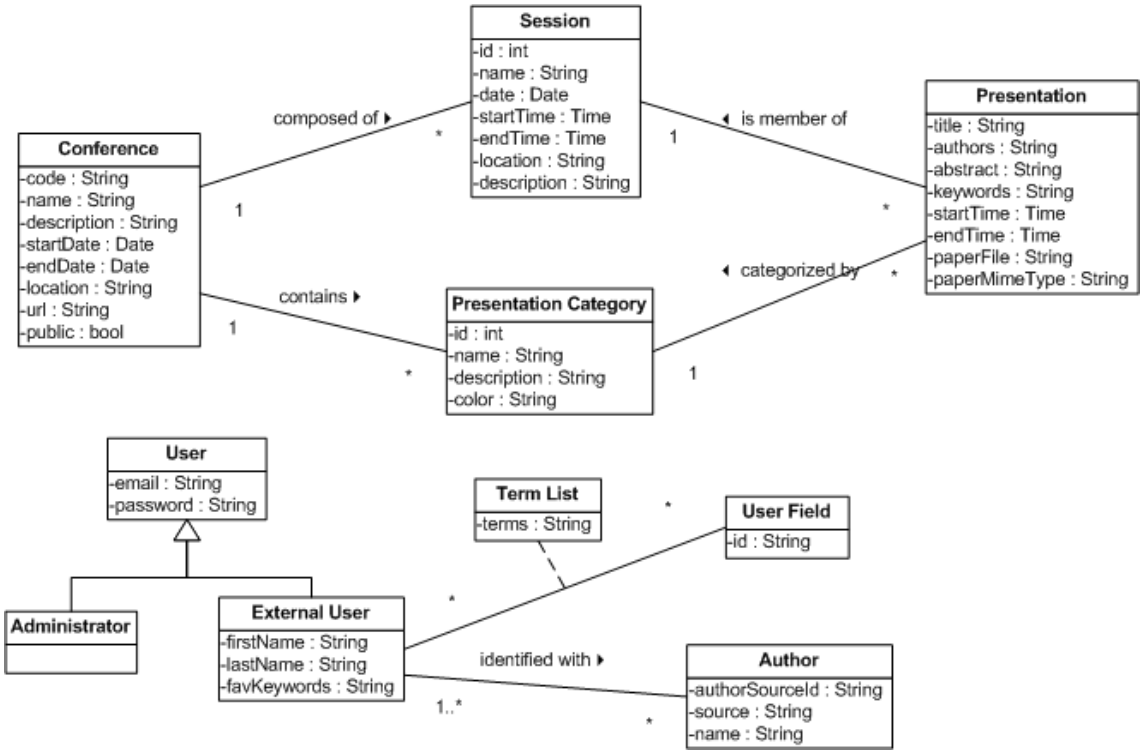


Figura 4.1: Diagrama conceptual

El diagrama conceptual va acompanyat d'unes restriccions d'integritat, que són propietats dels objectes i de les relacions que no es poden mostrar gràficament i que s'han d'especificar textualment. En el nostre cas, no les inclourem ja que són intuïtives i no aporten valor a la memòria.

4.2 Model de casos d'ús

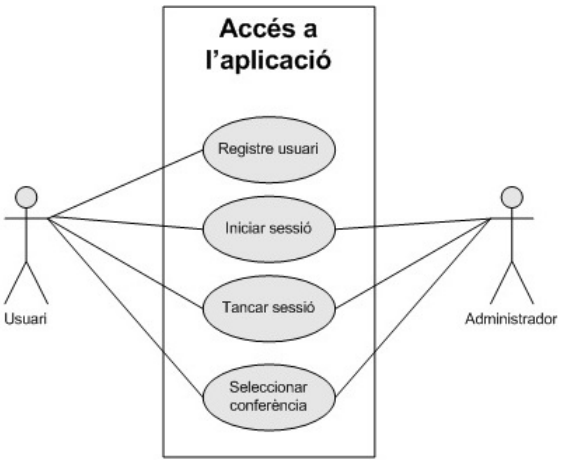
El model de casos d'ús ens ajudarà a especificar les funcionalitats del sistema mitjançant casos d'ús, és a dir, seqüències d'esdeveniments que descriuen de quina manera els actors (agents externs) utilitzen el sistema per assolir un determinat objectiu.

En l'aplicació podem distingir dos actors: els administradors (o organitzadors d'una conferència) i la resta d'usuaris (que generalment seran assistents d'una conferència).

4.2.1 Diagrama de casos d'ús

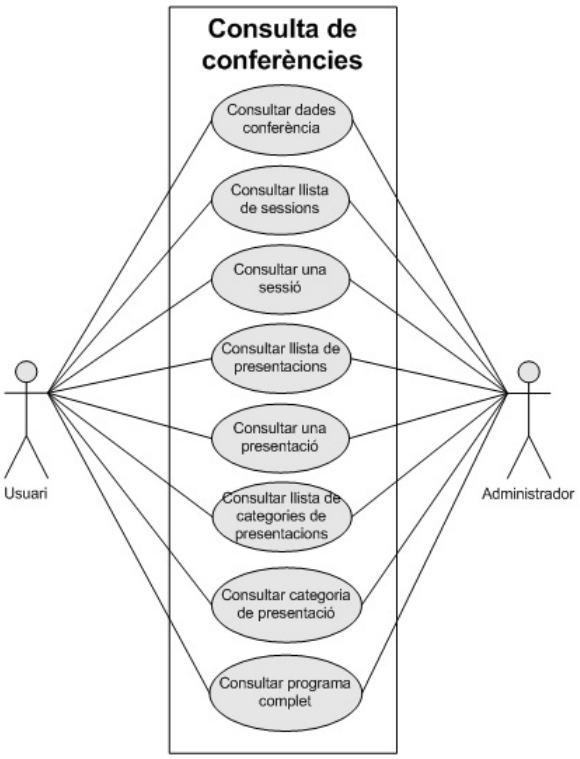
En el diagrama de casos d'ús es mostren, de forma simplificada, els actors del sistema, els casos d'ús i les relacions entre ells. S'han agrupat els casos d'ús segons la classificació del requisit funcional del que provenen.

Accés a l'aplicació

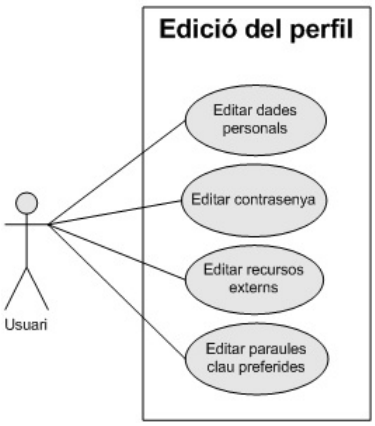


Administració de conferències

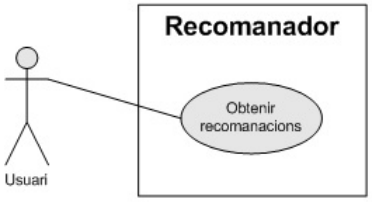
Consulta de conferències



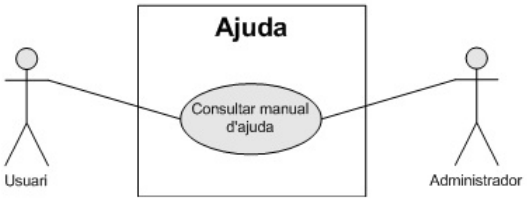
Edició del perfil



Recomanador



Ajuda



4.2.2 Especificació dels casos d'ús

Per a especificar els casos d'ús s'ha descrit la següent informació:

- Nom del cas d'ús
- Actors que intervenen
- Descripció breu
- Escenari d'èxit principal. Curs principal d'esdeveniments com a seqüència de passos numerats.
- Escenaris alternatius.

Per tal de facilitar la lectura de la memòria i degut a l'extensió d'aquesta secció, s'ha extret l'especificació dels casos d'ús en l'annex A Especificació.

Capítol 5

Tecnologies escollides

5.1 Aplicació web

S'ha decidit desenvolupar una aplicació web perquè es creu que ofereix diversos avantatges sobre el model d'aplicacions tradicional.

En primer lloc, a diferència de les aplicacions tradicionals, una aplicació web no necessita cap instal·lació especial per part de l'usuari. Obrint un navegador web i escrivint la direcció de l'aplicació ja es pot accedir al seu contingut immediatament. D'aquesta manera es pot accedir fàcilment a l'aplicació des de qualsevol ordinador i no són necessàries múltiples instal·lacions.

Una aplicació web també permet que usuaris amb diferents sistemes operatius puguin accedir, sense que sigui necessari desenvolupar i mantenir una aplicació específica per a cadascun d'aquests sistemes.

També resulta útil que es tracti d'una aplicació centralitzada. Totes les actualitzacions de l'aplicació que es desenvolupin seran visibles immediatament pels usuaris sense necessitat de noves descàrregues o instal·lacions.

Finalment, com que l'aplicació gestiona diferents tipus d'informació i documents i realitza cerques i recomanacions a partir d'ells, al realitzar-se en el servidor li evita a l'usuari la necessitat de descarregar tots els documents en el seu ordinador i l'allibera de la càrrega de les diferents tasques necessàries que realitza l'aplicació.

Encara que hem vist que una aplicació web ens ofereix molts avantatges, també hi ha algunes limitacions que cal tenir en compte.

La limitació principal és que es necessita una connexió a Internet per poder utilitzar l'aplicació. Però aquesta limitació no es podria solucionar ja que cal que els usuaris accedeixin a una mateixa base de dades comuna i, per tant, en qualsevol cas necessitarien una connexió a Internet.

També s'ha vist que és necessari un navegador web, però actualment tots els sistemes incorporen un navegador per defecte. D'aquesta manera, amb pràcticament qualsevol navegador web modern i sobre qualsevol sistema operatiu, un usuari podrà accedir i utilitzar immediatament l'aplicació.

Per aquests motius, es considera que una aplicació web és la millor solució per al nostre sistema.

5.2 Llenguatge de programació principal

Quan es tracta d'aplicacions web, hi ha una gran quantitat de llenguatges de programació disponibles que es poden utilitzar. Per desenvolupar l'aplicació s'ha intentat seleccionar el que millor s'adaptés als requisits.

Es buscava que es complissin les següents característiques:

- Suport per a programació web. Tal com s'ha vist en l'apartat anterior s'ha decidit desenvolupar una aplicació web i, per tant, calia un llenguatge que facilités el desenvolupament web.
- Multiplataforma. Es buscava un llenguatge que permetés que l'aplicació funcionés en el major nombre de sistemes possible, per a permetre a l'usuari utilitzar el que li fos més còmode.
- Facilitat d'instal·lació. L'aplicació resultant ha de se fàcil d'instal·lar per un usuari inexpert amb el mínim de configuració necessària, tal com s'especificava en els requisits.
- Llibreries i entorns de desenvolupament. El llenguatge ha de tenir disponibles llibreries i entorns de desenvolupament que ens ajudin durant el desenvolupament de l'aplicació web.

Després d'avaluar diverses opcions, es va decidir la utilització de **Java** [14, 13] de Sun Microsystems com a llenguatge de programació.

Java és un llenguatge de programació orientat a objectes. Al ser orientat a objectes, permet encapsular les funcionalitats que es requereixen en diferents objectes de classe, afavorint la reutilització de codi i facilitant la seva gestió.

El codi, normalment, es compila a un *bytecode* intermedi que després s'executa sobre una màquina virtual. L'ús d'una màquina virtual facilita que aplicació sigui multiplataforma, ja que la màquina virtual es troba disponible per a un gran nombre de plataformes. També disposa d'una àmplia comunitat i un gran quantitat de llibreries disponibles. D'entre aquestes llibreries, una de les més importants per a l'aplicació és la llibreria de recuperació d'informació Lucene, de la qual es parlarà més detalladament més endavant.

Existeix una versió empresarial de Java, anomenada JavaEE, que afegeix un conjunt de llibreries que permeten desenvolupar aplicacions basades en components modulars. Aquesta plataforma ofereix les APIs necessàries per a la programació de serveis web i comunicació amb base de dades que s'utilitzen en l'aplicació web.

Les aplicacions desenvolupades utilitzant la plataforma JavaEE es poden empaquetar, juntament amb totes les llibreries necessàries per a executar, dins d'un fitxer WAR. Aquest fitxer permet instal·lar l'aplicació de forma senzilla en un servidor d'aplicacions.

D'altra banda, també s'ha valorat el fet que és un llenguatge amb el que ja he treballat. Tot i que no tenia inconvenient en aprendre un nou llenguatge sempre que fos necessari o cost-efectiu, en aquest cas no s'ha trobat cap motiu. A més, aquest projecte ja tenia un volum de feina suficient (molt superior a les orientacions d'hores de la Facultat d'Informàtica de Barcelona) com per afegir-hi aprendre un nou llenguatge només pel fet d'aprendre.

Així, tal com hem vist, el llenguatge de programació Java ens permetrà desenvolupar una aplicació web amb l'ajuda de llibreries i entorns de desenvolupament, que després es podrà instal·lar fàcilment i que, gràcies a la màquina virtual, podrà executar-se en diferents plataformes.

5.3 Servidor web

Com s'ha vist, l'aplicació web s'instal·la en un servidor d'aplicacions JavaEE. S'ha decidit utilitzar **Apache Tomcat** [15, 13] desenvolupat per Apache Software Foundation, que si bé no és un servidor d'aplicacions complet, disposa d'un contenidor de servlets, que implementa les especificacions de servlets i JSP necessàries per al funcionament de l'aplicació.

5.4 Base de dades

Per dur a terme l'emmagatzematge de les dades necessàries per a l'aplicació s'utilitzarà una base de dades relacional, que permetrà poder organitzar-les adequadament en taules, definir relacions i expressar les restriccions per garantir la seva persistència.

Com que un dels requisits és la facilitat d'instal·lació, s'ha optat per incloure dins de l'aplicació una base de dades que permeti la seva utilització sense necessitat de configuració externa per part de l'usuari.

Com a base de dades interna s'utilitzarà **Apache Derby** [16, 13], una base de dades relacional escrita en Java i que es pot incloure directament en les aplicacions.

Es donarà la opció de configurar manualment la connexió a un servidor de base de dades extern com MySQL per als usuaris avançats.

5.5 Apache Lucene

Apache Lucene [17, 13] és una llibreria per a la recuperació d'informació, que s'utilitza per a la cerca de documents, informació dins dels documents o diferents metadades dels documents. Es tracta d'un projecte dins de l'Apache Software Foundation, escrit completament en Java i de codi lliure. Actualment és la llibreria de recuperació d'informació més popular.

Lucene oculta la complexitat del procés d'indexació i cerca, proporcionant al programador una API senzilla d'utilitzar. Cal destacar que es tracta d'una llibreria, i no d'una aplicació completa. Lucene deixa que la teva aplicació s'encarregui de les regles de negoci específiques al seu domini del problema. Hi ha un gran nombre d'aplicacions de cerca que s'han construït sobre la base de Lucene.

Un altre punt a destacar de Lucene és que permet crear índexs i fer consultes sobre qualsevol tipus de dades de les quals es pugui extreure text. No és important l'origen de les dades, el format o fins i tot el seu idioma, sempre que es pugui transformar en text. Això vol dir que es poden utilitzar pàgines web allotjades en servidors, documents a l'ordinador local, fitxers de text sense format, XML, HTML o fitxers PDF entre d'altres.

5.6 Capa de presentació

5.6.1 JSP

JSP o JavaServer Pages [18, 13] és una tecnologia Java desenvolupada per Sun per a facilitar la creació de pàgines web dinàmiques, que permet generar continguts dinàmics d'una forma ràpida i senzilla.

Aquesta tecnologia permet intercalar codi Java amb contingut estàtic de la pàgina, així com utilitzar algunes etiquetes XML especials que ofereixen diferents funcionalitats i encapsulen la lògica que genera el contingut de la pàgina. Addicionalment, també ofereix la possibilitat de crear noves llibreries d'etiquetes que s'adaptin a les nostres necessitats. A nivell pràctic, JSP s'encarrega d'interpretar el codi contingut en la pàgina JSP en el costat del servidor, per tal de generar la pàgina en format HTML i enviar-se-la al client.

Encara que es pot incloure el codi de l'aplicació directament, únicament s'utilitzarà JSP com a sistema de plantilles per generar de forma dinàmica les diferents pàgines que formen l'aplicació. La lògica de l'aplicació estarà en la capa de domini, mantenint així la separació de la capa de presentació.

5.6.2 HTML

HTML o HiperText Markup Language [19, 13] és el llenguatge de marques més utilitzat en l'elaboració de pàgines web. Un llenguatge de marques (o llenguatge de marcat) està format per un conjunt d'etiquetes que marquen les dades.

HTML utilitza aquestes etiquetes, juntament amb els atributs que puguin tenir, per descriure el contingut d'una pàgina web. Aquestes etiquetes poden utilitzar-se per definir els títols de les seccions de la pàgina, enllaçar a altres pàgines o incloure imatges.

Un navegador web s'encarrega de llegir el contingut del document HTML, interpretar el significat de les etiquetes i mostrar-lo a l'usuari com una pàgina web.

Algunes etiquetes HTML permeten incloure en el mateix document o de forma externa, altres tecnologies complementàries a HTML com CSS o Cascade Style Sheet, que s'utilitza per modificar l'aparença del contingut d'algunes etiquetes i definir l'estil que el navegador web utilitzarà quan mostri la pàgina web, o JavaScript, del que es parlarà en la següent secció.

5.6.3 JavaScript

JavaScript [13] és una implementació de l'estàndard ECMAScript utilitzada en navegadors web per dotar el codi HTML d'interactivitat i millorar la interfície d'usuari. Generalment el codi JavaScript s'adjunta als documents HTML i s'executa en el navegador web del client.

Com que el codi JavaScript s'executa en el navegador del client, s'utilitza una *sandbox* que aïlla l'execució del codi, evitant que pugui accedir al contingut de l'ordinador, però donant-li un accés total al document HTML que forma la pàgina web.

La funcionalitat principal de JavaScript en els navegadors web és modificar l'estructura de la pàgina web, utilitzant les llibreries del navegador per interactuar amb el Document Object Model (DOM), o detectar les accions que realitza l'usuari.

Utilitzant JavaScript es poden ocultar o afegir elements a la pàgina web o validar la informació introduïda en els formularis abans d'enviar les dades. Donat que s'executa en el navegador del client, es pot respondre ràpidament a les seves accions fent que l'aplicació tingui una millor capacitat de resposta.

5.6.4 AJAX

AJAX o Asynchronous JavaScript And XML [13] és, com indiquen les seves sigles, un conjunt de diferents tecnologies que s'uneixen per a permetre crear aplicacions web més interactives.

Permet a les aplicacions web realitzar peticions al servidor en segon pla. D'aquesta manera, es poden obtenir o validar les dades sense necessitat de recarregar la pàgina constantment, millorant així la interacció de l'usuari amb l'aplicació.

Al conjunt de tecnologies que formaven inicialment AJAX se li han anat sumant noves alternatives. Per exemple, encara que en el nom apareix XML, que inicialment és el format que s'utilitza per a l'intercanvi de dades entre el client i el servidor, aquest pot ser substituït per qualsevol altre format que faciliti l'intercanvi de dades.

En el nostre cas, el format que s'ha utilitzat en l'aplicació per realitzar l'intercanvi de dades entre el navegador web i el servidor ha estat JSON. S'ha considerat que en aquest cas és un format més lleuger i senzill i s'integra millor amb la resta de tecnologies utilitzades.

5.7 *Frameworks*

Desenvolupar una aplicació d'aquest tipus des de zero pot ser molt repetitiu, pesat i poc eficient. Per aquest motiu existeixen els *frameworks*, que tenen per objectiu millorar l'eficiència de creació de nou programari.

'*Framework*' és un terme adoptat de l'anglès i es pot traduir com a 'marc de treball'. En l'àmbit del desenvolupament de programari, els *frameworks* són productes de software que proporcionen l'automatització de tasques comunes del domini, així com una solució d'arquitectura integrada que pot ser fàcilment heretada per les aplicacions implementades sobre aquest marc.

S'ha decidit utilitzar *frameworks*, ja que poden augmentar la productivitat del desenvolupador i millorar la qualitat, fiabilitat i robustesa del nou programari. D'una banda, la millora en la productivitat es deu a que els desenvolupadors es poden centrar en les necessitats pròpies de la seva aplicació en lloc de perdre temps ocupant-se dels detalls de baix nivell més comuns. D'altra banda, les millores en el codi es deuen a que es reutilitza codi que ha estat construït i provat prèviament, i que els *frameworks* afavoreixen l'ús de bones pràctiques de programació i patrons de disseny.

5.7.1 *Web development framework*

S'ha decidit utilitzar **Apache Struts 2** [22, 13] com a *framework* per al desenvolupament d'aplicacions web Java. Aquest *framework* està dissenyat per simplificar el cicle de desenvolupament complet, des de la construcció fins al manteniment.

Apache Struts 2 no és tan sols una nova versió del conegut Struts, sinó un *framework* completament nou basat en el *framework* WebWork de OpenSymphony. Struts 2 implementa el patró de disseny Model-Vista-Controlador (MVC) i està construït seguint les millors pràctiques i diversos patrons de disseny acceptats per la comunitat.

A continuació es mostren les característiques que s'han tingut en compte per a triar Struts 2:

- Arquitectura simple i fàcil d'estendre.
- Permet seguir l'arquitectura en tres capes.
- Suport a la tecnologia AJAX.
- Els controladors són simples objectes Java, no requereixen heretar d'una altra classe i, per tant, no estan lligats al *framework*.
- Fàcilment provable, degut a que les classes amb les accions són simples objectes Java.
- Soporta múltiples opcions per a la vista: JSP, Freemarker, Velocity, XSLT.
- Integració amb SiteMesh i Tiles, *frameworks* per a la decoració i/o composició de pàgines web.

5.7.2 *Persistence framework*

Per a gestionar la persistència de dades s'utilitza la llibreria **Hibernate** [23, 13]. Hibernate és una llibreria Java que permet mapejar les taules de la base de dades relacional amb els objectes Java del domini utilitzats en l'aplicació. Ofereix una gestió transparent de la persistència de les entitats de negoci de l'aplicació, encarregant-se de la comunicació amb la base de dades. També disposa de facilitats per a la consulta i la recuperació de les dades i manté l'aplicació independent de la base de dades utilitzada. Amb pocs canvis en el fitxer de configuració de Hibernate, es pot migrar ràpidament de base de dades.

5.7.3 *Web-page layout and decoration framework*

Una pàgina web típica consta d'una capçalera, menú de navegació, peu de pàgina i seccions de contingut, i aquesta estructura s'ha d'aplicar de forma consistent i uniforme en totes les pàgines de l'aplicació. No obstant, implementar aquests elements estructurals de forma independent en un gran nombre de pàgines suposa un cost temporal i és difícil de mantenir si es volen fer canvis.

SiteMesh [21, 13] és un *framework* de decoració i estructuració de pàgines web que ofereix una solució elegant a aquest problema. Aquesta tecnologia permet crear el continguts en simples documents HTML només pensant en el seu contingut i sense tenir en compte com serà decorat. En el moment que es sol·liciti la pàgina HTML, SiteMesh interceptarà aquesta petició, analitzarà la

pàgina, obtindrà les propietats i les dades de tot el contingut, determinarà el decorador a utilitzar i generarà una pàgina apropiada amb les modificacions de l'original. Això permet tenir una separació clara entre el contingut de la vista i la presentació o estructura. En la figura 5.1 es pot veure un exemple del funcionament de SiteMesh.

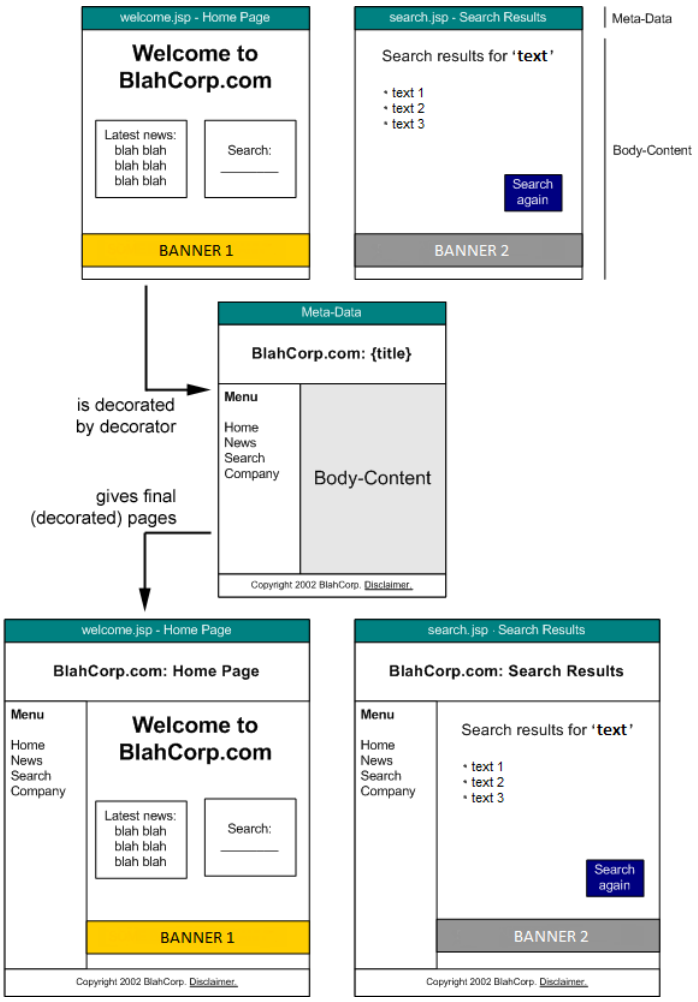


Figura 5.1: Esquema del funcionament de SiteMesh

D'aquesta manera es podrà donar una aparença uniforme a totes les pàgines sense haver-nos de preocupar d'implementar els elements estructurals en cada pàgina, evitant el codi repetit i millorant la mantenibilitat i canviabilitat de la presentació.

SiteMesh es construeix utilitzant Java 2 amb servlets, JSP i XML. Això ho fa ideal per al seu ús amb aplicacions J2EE.

Capítol 6

Disseny del sistema recomanador

Partint de les especificacions i de l'estudi del funcionament dels sistemes de recomanació i les diferents tècniques que es poden utilitzar, es presenten les decisions preses així com la conceptualització del sistema de recomanació que es construirà.

6.1 Definició de les entrades

Abans de procedir, recuperem la imatge vista en el capítol 3 per explicar les decisions preses punt a punt.

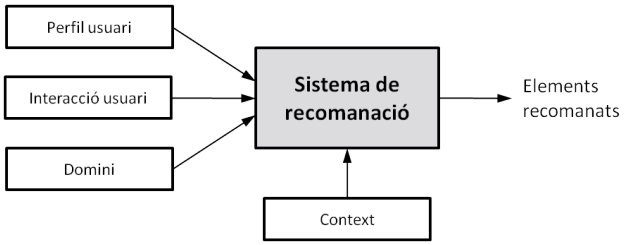


Figura 6.1: Entrades i sortides d'un sistema de recomanació

6.1.1 Domini

Com ja s'ha vist, el domini inclou totes les dades dels elements disponibles pel recomanador. En el cas d'aquest projecte el domini serà sobre conferències i estarà compost principalment per les dades de les seves sessions i presentacions. Un administrador s'haurà d'encarregar d'introduir aquestes dades al sistema per mitjà de l'aplicació.

6.1.2 Entorn

Dins de l'aplicació, el recomanador es trobarà en l'àmbit d'una conferència concreta. D'aquesta manera, a l'hora d'obtenir recomanacions només s'hauran de tenir en compte les dades del domini referents a la conferència donada.

6.1.3 Dades de l'usuari i interaccions

Les dades de l'usuari estaran formades per un conjunt de paraules clau sobre les seves preferències que ell mateix podrà introduir manualment en l'aplicació, així com informació que el sistema extraurà de forma semi-automàtica de recursos externs.

Tenint el nom i cognom que l'usuari haurà proporcionat al registrar-se, el sistema intentarà trobar aquest usuari en repositoris externs, per obtenir informació com per exemple el seu historial de publicacions.

Donat que el nom i cognom no identifiquen unívocament a una persona, aquest procés de identificar l'usuari de l'aplicació en els recursos externs no es pot fer de forma automàtica i fàcil. Es pot fer que sigui l'usuari qui ho seleccioni o es poden utilitzar algoritmes que intentin "endevinar" quins són els correctes basant-se en les dades de la conferència a la que assisteixen, cosa que no és una feina fàcil i no garanteix uns bons resultats.

Si s'encarrega l'usuari d'identificar-se en els recursos externs, té l'inconvenient de ser un procediment en el que es requereix la col·laboració de l'usuari. No obstant, l'avantatge d'aquest mètode és la transparència cap a l'usuari i la qualitat dels resultats obtinguts. Donat que tenir una informació de l'usuari fiable és un factor clau a l'hora d'obtenir els resultats, s'ha triat aquest mètode.

Aquest procediment haurà de ser el més fàcil i atractiu possible per a l'usuari. Per tal de facilitar aquesta tasca, el sistema farà una primera recopilació de possibles candidats trobats en diferents recursos (basant-se en el nom i cognom), dels quals l'usuari podrà seleccionar els correctes i descartar-ne la resta. També es donarà la opció de fer cerques en els recursos modificant el nom i cognom, per tal de resoldre les possibles variacions amb les que pot aparèixer.

En aquest projecte no es tractaran les interaccions de l'usuari, però es proposarà com a possible futura ampliació.

6.2 Plantejament del sistema de recomanació

Després d'estudiar les formes d'encarar un sistema de recomanació, s'han pres les següents decisions sobre el motor del sistema de recomanació.

6.2.1 Tècnica de correspondència

El primer punt important a determinar ha estat quin tipus de correspondència utilitzar en el sistema. S'ha decidit utilitzar una combinació entre l'anàlisi basat en els elements i el basat en els usuaris. D'una banda, es buscaran presentacions de la conferència similars a les publicacions anteriors de l'usuari, assumint que els interessos d'un usuari cap a un element són similars als d'experiències passades. Es donarà prioritat a les publicacions més recents. D'altre banda, assumint que els usuaris similars tenen gustos similars, es buscaran usuaris similars per tal d'obtenir informació extra. Es considerarà que dos usuaris son similars si són coautors en alguna publicació, donant prioritat als que tenen més publicacions en comú. Finalment, la cerca es refinarà utilitzant les paraules clau que l'usuari hagi introduït en el seu perfil.

Per obtenir informació de l'usuari com les seves publicacions i els seus coautors, es consultarà en repositoris bibliogràfics externs.

6.2.2 Tècnica de filtració

La tècnica de filtració utilitzada serà la basada en el contingut. Es buscaran presentacions de la conferència amb un contingut que s'aproximi al contingut de la informació recopilada sobre l'usuari. Per això s'utilitzaran tècniques de recuperació de la informació. D'una banda, es processaran les presentacions de la conferència i, de l'altra, el text de la informació de l'usuari i s'obtindrà una representació adequada en cada cas. A continuació, es farà una consulta sobre el domini fent servir la informació de l'usuari com a *query* i una funció de similitud determinarà la rellevància de cada document per a la consulta donada.

6.2.3 Tècniques de recuperació de la informació

Anàlisi i processat de text

A l'hora de processar el text s'han pres les següents decisions en l'àmbit de la recuperació de la informació:

- La tokenització seguirà les següents regles:
  - Es separaran les paraules en els signes de puntuació i s'eliminaran els signes de puntuació. Excepte els punts que no siguin seguits per un espai, que es consideraran part del *token*.
  - Es dividiran les paraules en els guions, excepte si hi ha un número en el *token* que, en aquest cas, es considera tot el *token* com un producte numèric i no es separa.
  - Les adreces de correu electrònic i els noms de hosts d'Internet es reconeixeran i es consideraran com un sol *token*.
- Es normalitzaran els *tokens* passant-los tots a minúscula.
- S'utilitzarà una llista de paraules buides més comunes que es descartaran a l'hora de processar el text.

Funció de similitud

Tal com s'ha vist en el capítol 5, s'utilitzarà la llibreria **Apache Lucene** per a desenvolupar la indexació i cerca. En el capítol 8 es poden veure més detalls sobre la implementació de la cerca i indexació amb Lucene. En aquest apartat ens centrarem en l'algoritme de *scoring* o funció de similitud que utilitza, ja que influeix en la conceptualització del sistema recomanador.

Lucene combina el model booleà i el model vectorial. Els documents "aprovats" pel model booleà es puntuen bastant-se en el model vectorial.

Tal com s'ha vist en els conceptes preliminars, en el model booleà els documents i les consultes es representen com a vectors de pesos en un espai multidimensional, on cada índex és una dimensió diferent. Lucene calcula els pesos mitjançant la fórmula *tf-idf* també vista anteriorment.

S'ha vist que una possible funció per a determinar la similitud d'un document d per a una consulta q era la mesura del cosinus de l'angle format pels vectors de pesos $V(q)$ i $V(d)$.

$$similaritat(q,d) = \frac{V(q) \cdot V(d)}{\|V(q)\| \cdot \|V(d)\|}$$

on $V(q) \cdot V(d)$ és el producte escalar dels vectors de pesos, i $\|V(q)\|$ i $\|V(d)\|$ són les seves normes euclidianes.

Lucene utilitza utilitza una una versió refinada de la mesura del cosinus per a calcular la similitud d'un document per a una determinada consulta:

- La normalització del vector $V(d)$ al vector unitari pot ser problemàtic ja que elimina tota la informació de la longitud del document. Per evitar aquest problema, Lucene utilitza un factor diferent per a la normalització de la longitud d'un document. Aquest factor és més petit quan la llargada del text és més gran. Així es dona més importància a les coincidències en textos més curts, ja que quan més llarg es el text menys rellevància té la coincidència. Aquest factor normalitza a un vector igual o més gran al vector unitari.
- A l'hora d'indexar, es poden especificar que certs documents són més importants que altres, mitjançant l'assignació d'un valor anomenat *boost*.
- En Lucene, els documents estan compostos per un o més camps. Cada terme de la consulta s'aplica a un sol camp, la normalització de la longitud d'un document es fa per la longitud d'un cert camp i, a part de poder assignar un *boost* als documents també es poden assignar als camps específics.
- A l'hora de cercar, també es poden introduir *boosts* a cada terme de la consulta. La contribució total d'un terme de la consulta a la puntuació del document es multiplica pel *boost* d'aquest terme de la consulta.
- Un document pot coincidir amb una consulta de múltiples termes sense haver de contenir tots els termes de la consulta, però s'afegeix un factor per a poder premiar els documents que continguin més termes de la *query* a través d'un factor de coordinació $coord(q,d)$.

Així, podem reescriure la fórmula de similitud que utilitza Lucene:

$$score(d,q) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t\ in\ q} tf(t,d) \cdot idf(t)^2 \cdot boost(t,q) \cdot norm(t,d))$$

A continuació es descriuen cadascun dels factors:

- **score(d,q)** Rellevància o similitud d'un document d per a una determinada consulta q.
- **coord(q,d)** Factor de la puntuació basat en el nombre de termes de la consulta que apareixen en el document d. Aquest factor es calcula per defecte en Lucene com la fracció entre nombre de termes de la consulta que coincideixen en el document i el nombre total de termes de la consulta.
- **queryNorm(q)** Factor de normalització de la query q.
- **tf(t,d)** Freqüència del terme t en el document d. Cal destacar que $tf(t,q)$, és a dir, la freqüència del terme en la consulta, s'assumeix que és 1 i per això no apareix en l'equació.

- **idf(t)** Freqüència inversa del terme t en tots els documents. Els termes molt comuns tenen un valor baix, mentre que els termes estranys tenen un valor alt.
- **boost(t,q)** Factor de *boost* del terme t en la consulta q.
- **norm(t,d)** Factor de normalització del terme t en el document d. Consisteix en el producte dels factors de *boost* del camp i del document i del factor de normalització de la dimensió del document.

Indexació

Hem vist que els documents de Lucene normalment consten de diferents camps. A més, per a cada camp es pot dir si s'ha d'indexar o no (segons si es volen fer cerques en aquest camp o no) i també si s'ha d'analitzar i processar el seu valor abans d'indexar-lo o no. Generalment no s'analitzen camps que representen dates, URLs, etc. També es pot indicar si vol emmagatzemar es seu valor original per tal de poder-lo recuperar en els resultats de la cerca. Generalment s'emmagatzemen els camps que es volen mostrar en la llista de resultats per tal d'evitar una crida a la base de dades per a cada resultat.

A l'hora de decidir els camps que tindran els documents de Lucene en el nostre projecte s'han tingut en compte les següents consideracions:

- Es volen efectuar cerques sobre el títol, el resum i el fitxer amb el text complet de les presentacions.
- En la llista de resultats voldrem mostrar el títol i el resum.
- S'ha de poder associar el document amb la presentació guardada en la base de dades per mitjà de l'identificador de l'objecte. D'aquesta manera, si l'usuari es mostra interessat, es podrà accedir a la base de dades per recuperar la resta d'informació de la presentació.

A partir d'aquestes consideracions, s'ha deduït que calen camps per a l'identificador de la presentació, el seu títol, el seu resum i el text extret del fitxer que té associat. A continuació es mostra un resum de les característiques de cadascun d'ells:

Camp	**Descripció**	**Analitzat**	**Indexat**	**Emmagatzemat**
id	Identificador de la presentació en la base de dades del sistema.			✓
title	Títol de la presentació	✓	✓	✓
abstract	Resum de la presentació	✓	✓	✓
fullArticle	Text extret del seu *paper*	✓	✓	

En seccions anteriors s'ha vist que es pot donar un pes especial a alguns dels camps del document. En el nostre cas ens interessaria que les coincidències en el títol tinguessin més pes que les coincidències en el resum, i les del resum més pes que les de l'article complet.

No obstant, no s'ha assignat explícitament un *boost* especial als camps, ja que el títol tendeix a ser bastant més curt que el resum i aquest molt més curt que el text complet. D'aquesta manera, com que Lucene normalitza en funció de la dimensió del camp, ja assignarà implícitament un pes especial a les coincidències en els camps més curts.

Consulta

Per a realitzar la consulta sobre l'índex, s'utilitzaran tres grups de dades: les paraules clau introduïdes per l'usuari, el text format per la unió de títols de les seves publicacions i el text format per la unió dels títols de les publicacions dels seus coautors. Aquests dos últims grups de dades els generarà el sistema a partir de la informació que recuperi en les fonts externes.

Per convertir tota aquesta informació en una consulta raonable es seguiran els següents passos. En primer lloc, s'extrauran els termes dels textos generats pel sistema utilitzant un analitzador. D'aquesta manera s'obtindran un conjunt de termes normalitzats i sense paraules buides, que no ens interessen. Agruparem els termes iguals dins de cada grup i, de cada terme, n'obtindrem la seva freqüència i li donarem aquest pes al terme. Donat que hi ha diferents grups de termes i no volem que afecti la seva dimensió al resultat de la consulta, normalitzarem la freqüència del terme dividint-la entre el total de termes del seu grup.

A més, voldrem que les paraules clau tinguin un pes especial ja que és una informació donada explícitament per l'usuari. Així, se li donarà un pes de α (tal que $0 < \alpha < 1$) a la informació introduïda per l'usuari, és a dir, les paraules clau, i l'altre $1 - \alpha$ a repartir entre la informació extreta pel sistema. Dins d'aquest $1 - \alpha$, s'assignarà β a les paraules extretes de les publicacions pròpies i el $1 - \beta$ a les publicacions dels coautors. Així, cada grup tindrà un pes de α, $(1 - \alpha)\beta$ i $(1 - \alpha)(1 - \beta)$, respectivament. El valor de α i β es determinarà experimentalment.

D'aquesta manera, es construirà una consulta amb cadascun dels termes dels grups mencionats i a cadascun se li assignarà un *boost* equivalent al de la seva freqüència dins del grup, normalitzada en funció del total de termes del grup, i multiplicada pel pes del grup. Lucene internament ja s'encarregarà d'aplicar la fórmula del *tf-idf* en funció dels documents sobre els que es faci la cerca.

Per tal d'optimitzar la consulta es limitaran el nombre de termes de cada grup utilitzant heurístiques. Partint de la fórmula *tf-idf*, probablement ens interessi quedar-nos amb els termes que tinguin un valor de tf més alt. Donat que aquesta informació ja la tenim directament i no caldrien càlculs addicionals, es podria establir un llindar en funció de la freqüència del terme en la consulta. Una altra heurística a considerar és que els termes amb un *idf* alt (és a dir, que tenen una baixa freqüència en tota la col·lecció de documents i, per tant, es consideren més representatius) tendeixen a ser més llargs. Així, també podem establir un ordre de preferència en funció del nombre de caràcters a l'hora de limitar els termes.

6.2.4 Fonts de dades externes

En la primera versió d'aquest projecte, el sistema només recuperarà dades dels autors en el repositori DBLP.

DBLP és un servidor web que proporciona dades bibliogràfiques sobre les principals publicacions científiques i conferències en l'àmbit de la informàtica. Inicialment, el servidor es centrava en sistemes de bases de dades i programació lògica, però s'ha anat estenent gradualment a altres camps de la informàtica. Actualment, DBLP indexa més d'un milió d'articles i conté més de 10.000 enllaços a pàgines d'autors.

Tot i que de moment només s'utilitzarà el repositori DBLP, es vol tenir present la possibilitat

d'ampliació a noves fonts d'informació com podria ser Google Scholar, l'ACM Digital Library, etc. D'aquesta manera es dissenyarà un mòdul de gestió dels recursos externs que permeti afegir noves fonts sense repercutir a la resta del sistema.

Capítol 7

Disseny del software

La fase de disseny és l'activitat d'aplicar diferents tècniques i principis amb el propòsit de definir un sistema amb el suficient detall per permetre la seva construcció física, és a dir, la seva implementació.

El disseny pren com a punt de partida l'especificació i les tecnologies que s'ha decidit utilitzar. En aquest apartat, es desenvoluparà l'arquitectura del sistema definint quins patrons arquitectònics i de disseny s'utilitzaran de cara a la implementació del sistema.

7.1 Patrons arquitectònics

Els patrons arquitectònics són patrons de software que ofereixen solucions a problemes recurrents en el disseny de l'arquitectura de software. En comparació amb els patrons de disseny, els patrons arquitectònics tenen una escala més gran i, per tant, afecten a tot el sistema.

7.1.1 Arquitectura en tres capes

L'arquitectura en capes sorgeix de la necessitat de descomposar un sistema gran en grups de subtasques que estiguin a un nivell determinat d'abstracció. Aquest patró arquitectònic consisteix en estructurar el sistema en un nombre apropiat de capes que es col·loquen verticalment i fent que cada capa només es pugui comunicar amb la capa immediatament superior i inferior. Tots els components d'una mateixa capa han de treballar al mateix nivell d'abstracció.

El principal avantatge que s'obté utilitzant aquest patró és la reducció de l'acoblament, fent que sigui més fàcil poder generar canvis en una capa sense que les altres es vegin afectades. Per tant, també permet que el manteniment tingui menor cost i es pugui adaptar més ràpid a les tecnologies que vagin sorgint. A canvi, el model provoca que el flux d'informació perdi eficiència, ja que es veu obligat a travessar per totes les capes intermèdies.

En aquest projecte s'ha decidit fer ús d'una arquitectura en tres capes: presentació, domini i gestió de dades. A continuació es mostra l'estructura i detallen les responsabilitats de cadascuna d'aquestes capes.

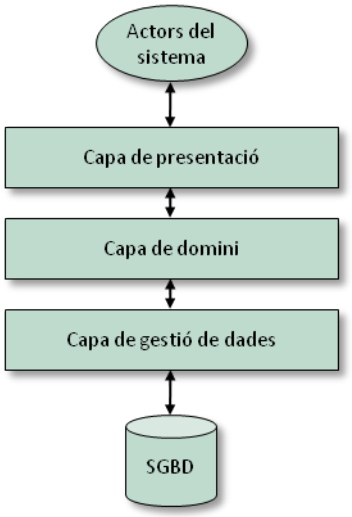


Figura 7.1: Arquitectura en tres capes

Capa de presentació. És la responsable de la interacció amb l'usuari. Conté la interfície necessària per presentar la informació i obtenir dades. S'ocupa d'assabentar-se de les peticions dels usuaris, ordenar l'execució d'accions, comunicar els resultats de les accions als usuaris, així com del tractament dels elements visuals (finestres, diàlegs, botons, llistats). Aquesta capa coneix com presentar les dades, però ignora quines transformacions cal fer per donar resposta a les peticions de l'usuari.

Capa de domini. És la responsable de la implementació de les funcionalitats del sistema. El domini rep l'entrada del nivell de presentació i interactua amb la capa de dades per executar les operacions de domini necessàries. És a dir, s'ocupa de rebre els esdeveniments de presentació, controlar-ne la validesa, canviar l'estat del domini, executar les accions encomanades, fer consultes a la capa de dades, obtenir-ne el resultat i comunicar la resposta. Aquesta capa coneix com satisfer les peticions de l'usuari però ignora on es guarden les dades i com es presenten a l'usuari.

Capa de gestió de dades. Coneix on i com estan emmagatzemades les dades i s'ocupa de gestionar la seva persistència. Les funcions concretes d'aquesta capa dependran de l'estratègia de persistència triada.

7.1.2 Orientació a objectes

El patró arquitectònic d'orientació a objectes consisteix en veure el sistema com una col·lecció d'objectes que interactuen entre si, on cadascun d'aquests objectes té unes responsabilitats assignades. Cada objecte agrupa les seves dades i els mecanismes per a manipular-les, i la comunicació entre objectes es realitza mitjançant la invocació d'operacions oferts per altres objectes.

La orientació a objectes ofereix els següents avantatges:

- Els canvis en el codi no es propaguen a tot el sistema

- Es poden reutilitzar i substituir components per implementacions alternatives.
- L'agrupació de responsabilitats similars afavoreixen la comprensió i el manteniment.

7.2 Patrons de disseny

Un cop determinats els patrons arquitectònics, els patrons de disseny donen un esquema per refinar els seus subsistemes o components, o les relacions entre ells.

En aquest apartat, es presenten els patrons de disseny més significatius que s'han utilitzat.

7.2.1 Model-Vista-Controlador(MVC)

El patró MVC permet separar la lògica de negoci de la part de presentació i entrada de dades, dividint l'aplicació en els següents components:

- **Model.** Representa i manté l'estat del domini i també implementa les operacions que accedeixen i modifiquen aquest estat.
- **Vista.** Obté les dades necessàries del model i mostra la informació a l'usuari d'una determinada manera.
- **Controlador.** S'encarrega de processar les peticions de l'usuari, ordenar l'execució de les operacions del model que calguin i triar una vista per a mostrar el contingut.

Tal com s'ha vist en el capítol 5 , Tecnologies escollides, s'utilitzarà Struts 2 i l'arquitectura bàsica d'aquest *framework* implementa aquest patró, tal com es pot veure en la figura 7.2.

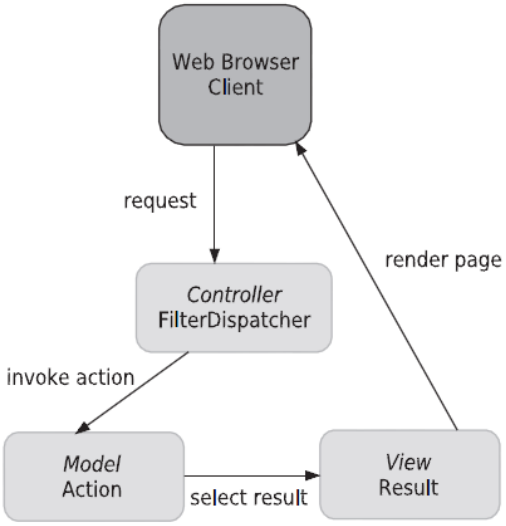


Figura 7.2: Patró MVC implementat per Struts 2

7.2.2 *Front Controller* (Controlador Façana)

L'objectiu d'aquest patró és crear un punt d'entrada comú al sistema per tal de capturar tots els esdeveniments externs relacionats. El *Front Controller* centralitzarà la lògica de control i s'encarregarà de redirigir la petició a algun altre controlador o objecte de domini. Això permet aplicar una lògica de control comuna a múltiples peticions sense tenir codi duplicat.

Struts 2 disposa de l'anomenat *FilterDispatcher*, que és bàsicament un controlador façana que rep totes les peticions HTTP i les redirigeix als controladors propis del sistema segons la petició rebuda.

7.2.3 Patró iterador

El patró iterador defineix una interfície que declara els mètodes necessaris per accedir seqüencialment a un conjunt d'objectes d'una col·lecció. Aquest patró permet recórrer diferents estructures de dades sense preocupar-nos de la seva implementació interna.

7.3 Esquema de disseny

Finalment, es presenta una visó general de com interactuaran els elements dels diferents nivells d'abstracció i es presenta el paper de les tecnologies que s'ha decidit utilitzar. En la figura 7.3 es pot veure l'esquema de disseny resultant.

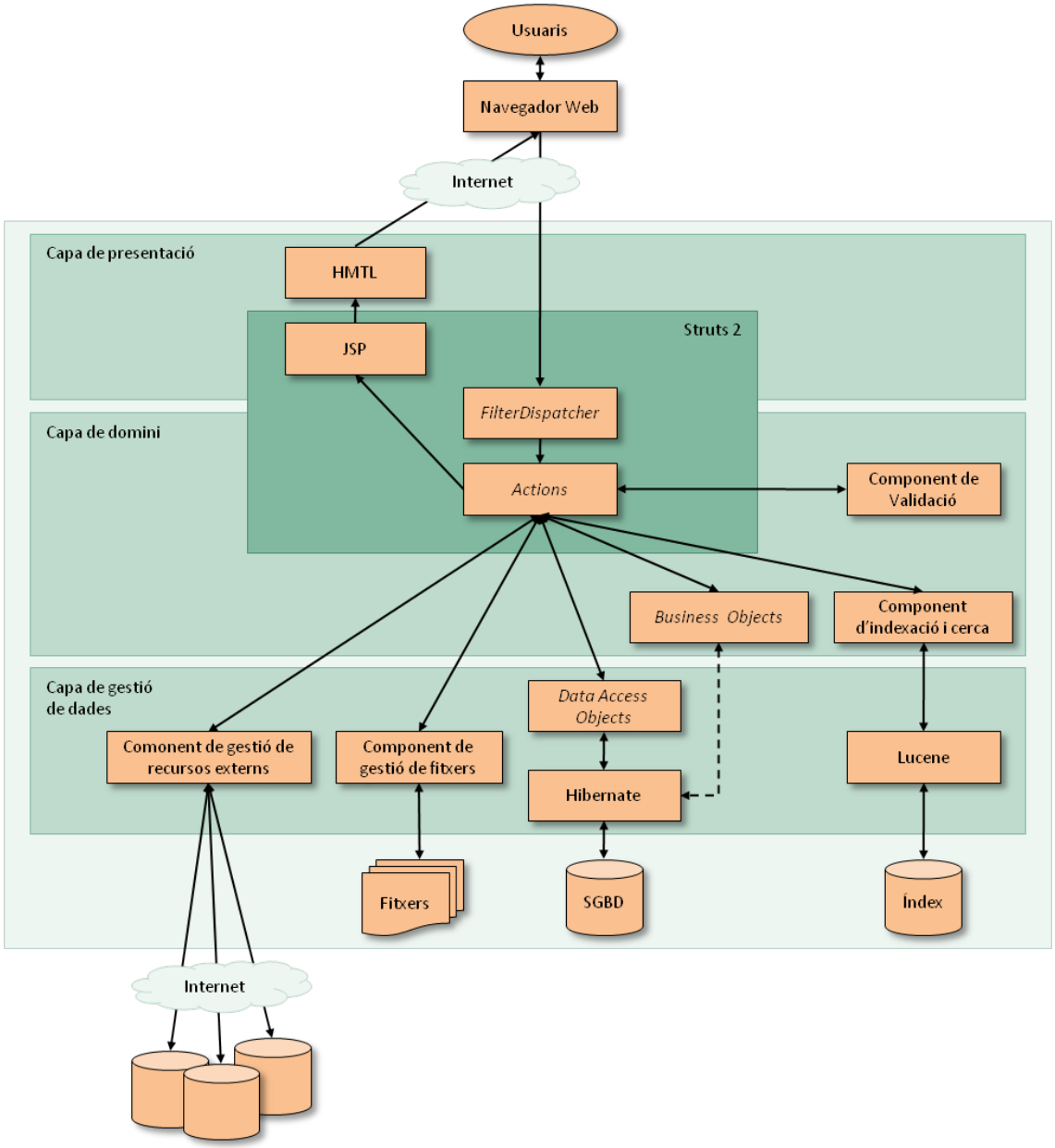


Figura 7.3: Esquema de disseny

L'usuari interacciona, a traves del navegador web, amb les vistes HTML generades a partir dels JSP. Tant les vistes com el *FilterDispatcher* i els *Actions*, formen part de l'arquitectura interna de Struts 2. Tal com ja s'ha vist en apartats anteriors, el *FilterDispatcher* actua com un controlador façana que rep totes les peticions HTTP. Un cop rep la petició, s'encarrega de consultar les seves assignacions (*mappings*) per tal de determinar quina *Action* ha de resoldre aquesta petició i l'invoca. En Struts 2, un *Action* encapsula un conjunt de funcionalitats que representen un cas d'ús del sistema. Quan l'*Action* acaba la seva tasca, tria la vista que cal mostrar a l'usuari que va enviar la petició. Un *Action* pot portar a diferents vistes segons el resultat de l'execució. Per exemple, si es tracta del registre d'un usuari, pot portar a una vista determinada si tot ha anat correctament, o mostrar una pàgina d'error si s'han introduït dades incorrectes.

Els *Actions* es comunicaran amb el mòdul de validació per tal de comprovar les dades entrades pels usuaris a través dels diferents formularis de l'aplicació.

L'aplicació contindrà nombrosos formularis HTML a través dels quals els usuaris podran entrar dades en el sistema. En la majoria dels casos, aquestes dades hauran de complir unes condicions i, per tant, caldrà validar-les. Les validacions es poden dur a terme en el servidor a l'enviar el formulari o en el client per mitjà de JavaScript en el navegador. No obstant, per seguretat, sempre cal validar com a mínim en el servidor, per tal de protegir l'aplicació d'usuaris malintencionats. Tal es veurà en el capítol 8, existeixen aplicacions com Firebug que permeten modificar fàcilment el codi JavaScript i, per tant, es podrien eludir fàcilment les comprovacions i posar en perill la integritat de les dades del sistema. En el nostre projecte, el component de validació és el que s'encarregarà dur a terme aquestes validacions en la part del servidor.

Els *Business Objects* són les classes del domini de l'aplicació que encapsulen totes les dades i els comportaments associats a la entitat que representen. Aquests objectes seran mapejats a partir de tècniques ORM (*Object-Relational Mapping*) que en el nostre cas implementa Hibernate. D'altre banda, els *Data Acess Objects* s'implementen com a classes que s'encarreguen de comunicar-se amb Hibernate per realitzar consultes i modificacions a la base de dades.

A part de les dades guardades en la base de dades, també tenim un directori amb els fitxers de les presentacions que es gestionen a través del mòdul corresponent.

El component d'indexació i cerca s'encarrega de realitzar totes les tasques relacionades amb aquests processos així com d'utilitzar l'API de Lucene per a la creació i lectura de l'índex.

Finalment, tenim el component de gestió de recursos externs, que s'encarrega de gestionar la recopilació i processat d'informació de diferents repositoris externs. Al ser un component independent, s'aconsegueix que si es volen afegir o modificar els repositoris utilitzats no afecti a la resta del sistema.

7.4 Disseny de la capa de domini

Partint del diagrama conceptual de l'especificació s'ha elaborat el diagrama conceptual dels objectes de negoci del domini. S'ha normalitzat el diagrama i s'han fet algunes modificacions tenint present la implementació.

Al normalitzar l'esquema conceptual han desaparegut les classes associatives. S'ha creat una nova classe `User Field` relacionada amb `User`, que conté l'identificador del Field i els atributs propis de l'antiga classe associativa.

També s'han eliminat les especialitzacions d'usuari, agrupant-les en una mateixa taula. S'ha afegit un nou atribut `admin` en la taula `User` per tal de poder diferenciar de quin perfil es tracta en cada cas, i s'han fet opcionals els atributs no compartits.

S'ha cregut convenient afegir atributs identificadors numèrics autogenerats estables que no variïn al llarg del temps per tal de simplificar la implementació, ja que en la majoria de casos la resta d'atributs poden ser modificats pels usuaris.

També s'ha afegit una nova classe per a representar l'estat de l'aplicació, segons si ja ha estat configurada o no per tal de poder actuar en conseqüència en cadascun dels casos.

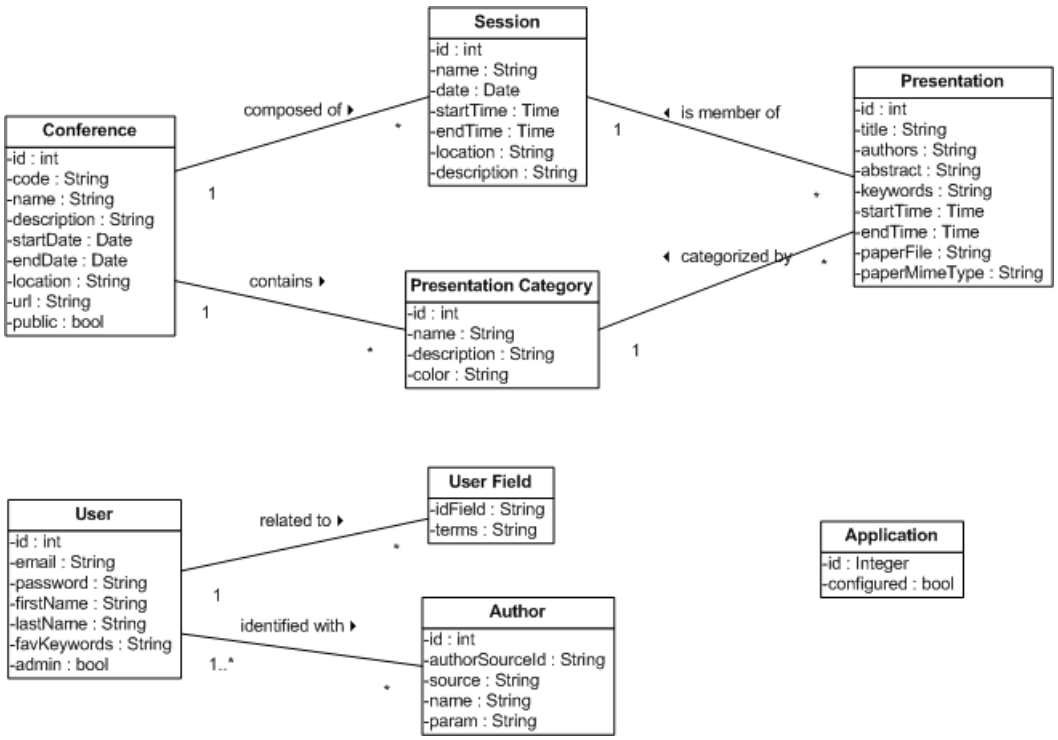


Figura 7.4: Diagrama de classes del domini

7.5 Disseny de la capa de gestió de dades

7.5.1 Disseny lògic de la base de dades

Com que es disposa d'una base de dades relacional, l'esquema es crea directament de la transformació del diagrama de classes del domini. A continuació, en la figura 7.5 es mostra l'esquema del model de dades que s'utilitzarà.

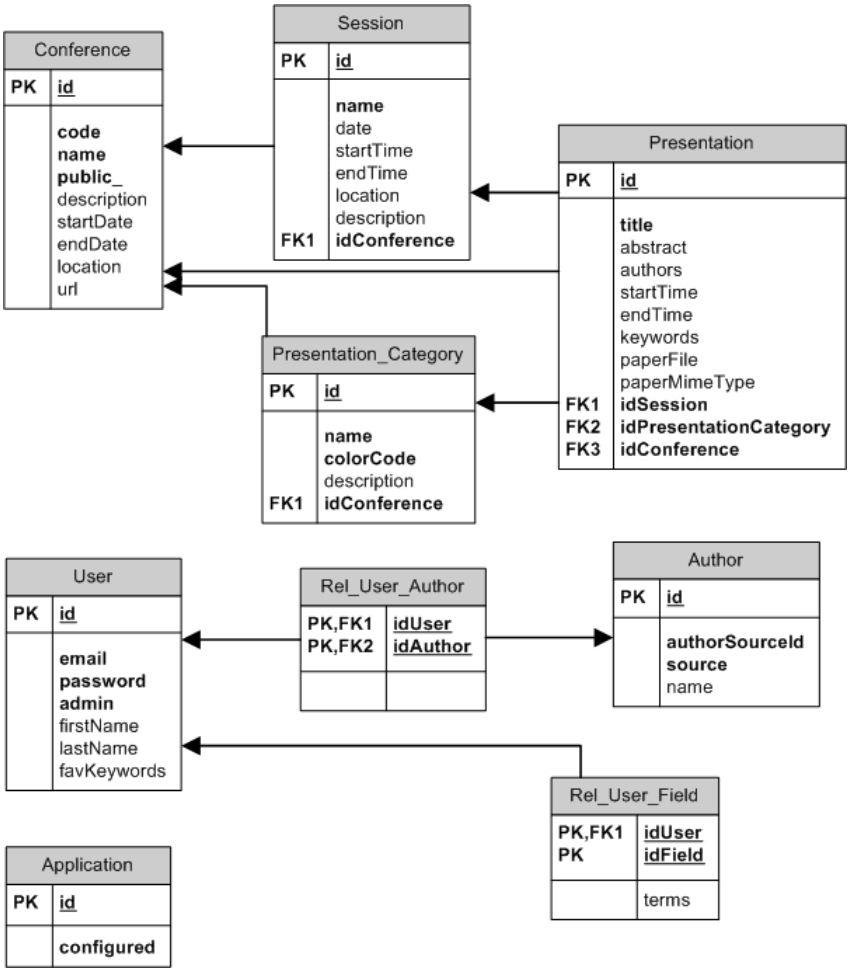


Figura 7.5: Model de dades

7.5.2 Estratègia de gestió de la persistència

Podem trobar diferents estratègies de gestió de la persistència: generació automàtica i disseny directe.

La generació automàtica de la persistència es basa en un traduir les actualitzacions fetes en el model conceptual del domini en actualitzacions de la base de dades d'una forma gairebé automàtica. Aquesta estratègia requereix especificar la correspondència entre diagrama de classes i l'esquema de la base de dades. D'aquesta manera, la gestió de persistència és transparent al dissenyador.

Si s'utilitza disseny directe, la capa de gestió de dades és qui s'encarrega de fer la feina de gestionar la informació que s'obté o es guarda en la base de dades. El disseny de la gestió de dades s'ha de fer tenint en compte el Sistema de Gestió de Bases de Dades (SGBD) que s'utilitzarà en la implementació, ja que la capa de gestió de dades haurà de treballar de manera diferent si es tracta d'un SGBD relacional o orientat a objectes, per exemple. En aquest cas, el dissenyador ha de conèixer bé el SGBD.

En canvi, si s'utilitza la generació automàtica, el dissenyador només ha de definir la correspondència entre l'estructura de classes del domini i l'estructura de taules de la base de dades. El traductor

de dades s'encarregarà de mantenir la correspondència entre els objectes i les taules. D'aquesta manera, quan es facin actualitzacions sobre les classes del domini, es traduiran de forma (gairebé) automàtica com a actualitzacions sobre les taules de la base de dades. En aquest cas, la gestió de la persistència és transparent al dissenyador.

Per a aquest projecte s'ha escollit l'estratègia de generació automàtica, ja que permet operar directament sobre objectes del domini, evitant-nos la definició de cadascuna de les operacions SQL que caldrien per accedir/modificar dades de la base de dades, i dota al sistema de portabilitat entre diferents SGBDs.

Tal com s'ha vist en el capítol 5, s'utilitza Hibernate per a aquest propòsit.

7.6 Disseny de la capa de presentació

A l'hora de dissenyar l'estructura de la interfície d'usuari s'han tingut en compte les següents decisions:

1. Des de qualsevol pàgina s'ha de poder veure el logotip o nom de l'aplicació.
2. L'usuari ha de poder tornar a la pàgina principal amb un sol clic des de qualsevol lloc de la web.
3. L'usuari ha de poder tancar sessió des de qualsevol lloc de la web.
4. L'usuari ha de poder navegar a través de la web d'una forma senzilla i intuïtiva.
5. L'usuari ha de poder accedir a qualsevol secció en un màxim de tres passos.
6. L'usuari ha de poder veure de forma ràpida en quina pàgina es troba actualment.

Per tal de complir aquests requeriments, totes les pàgines de la web seguiran la mateixa estructura. A continuació es detalla l'esquema bàsic.

Capçalera. Contindrà el nom corporatiu així com informació sobre la sessió de l'usuari. Si ha iniciat sessió, es mostrarà el seu identificador i un enllaç per a poder tancar sessió i, si no ha iniciat sessió, es mostraran enllaços per a iniciar sessió i registrar-se. També es mostrarà el codi de la conferència en la que s'ha entrat i un enllaç per sortir d'aquesta i tornar a l'índex. La capçalera estarà present en qualsevol pàgina que visiti l'usuari.

Menú de navegació. Contindrà enllaços a les principals seccions de la web. El menú sempre serà visible però, depenent de la situació i els permisos de l'usuari, podrà contenir diferents enllaços. Aquests enllaços permetran millorar la navegabilitat i reduir el nombre de clics per arribar d'una secció a una altra. A l'entrar en una secció determinada, es mostrarà ressaltat el seu enllaç en el menú per tal d'ajudar a l'usuari a ubicar-se en la web.

Peu de pàgina. De moment no conté cap informació, però s'ha inclòs en el disseny per mantenir l'estructura a la que l'usuari està acostumat i també per reservar l'espai per a possibles modificacions en futures versions.

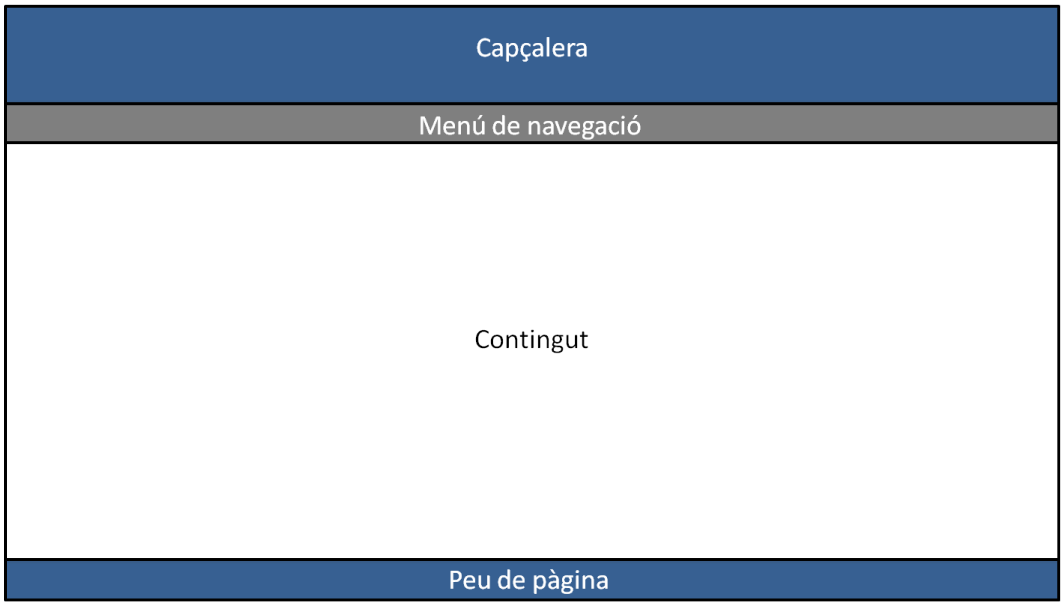


Figura 7.6: Estructura de les pàgines

Contingut. En aquest espai es mostraran els continguts de les diferents seccions.

D'altre banda, també és important que el disseny de la interfície faciliti la tasca d'introduir dades a l'usuari, per mitjà de diferents tècniques, com per exemple:

- Selectors de dates a través d'un calendari.
- Selectors de color a través d'una paleta visual.
- *Watermarks* o marques d'aigua en els camps de formulari que requereixin especificar un format determinat, com els camps de dates i hores.
- Editors de tags per a la gestió de paraules clau.

En el cas de les seccions que es carreguin o s'actualitzin de forma asíncrona per AJAX, es mostrarà una imatge animada que indiqui a l'usuari que s'està processant la petició.

Un altre punt és que tots els missatges d'error de l'aplicació hauran de tenir un estil uniforme. S'ha decidit utilitzar una tonalitat vermella que típicament s'associa als errors i permet identificar-los més fàcilment. En el cas que es tracti d'un error de validació d'un formulari, s'haurà de mostrar l'error sota del camp que l'ha provocat, per tal que l'usuari pugui localitzar fàcilment les dades invàlides o no acceptades.

Capítol 8

Implementació

L'objectiu d'aquest capítol és presentar les eines utilitzades durant el desenvolupament (entorn de treball), així com els detalls i problemes trobats durant la fase d'implementació.

8.1 Entorn de treball

8.1.1 Eclipse

Eclipse [29, 13] és un entorn integrat de desenvolupament (IDE)[1] multiplataforma de codi obert, molt estès en la comunitat informàtica, que permet tècniques avançades de refactorització i anàlisi de codi.

Per aquest projecte s'ha utilitzat el paquet per a Java EE de l'Eclipse Galileo (versió 3.5) que conté entre altres, un IDE específic per a aplicacions Java, un compilador de Java intern i eines per al desenvolupament Java EE (Java, HTML, CSS, JavaScript...). Inclou, també, per defecte la integració del servidor web. D'aquesta manera, dins del propi Eclipse es pot arrancar, parar i configurar, en aquest cas, el servidor Tomcat.

Eclipse també permet ampliar les seves funcionalitats segons les necessitats del projecte mitjançant la incorporació de plug-ins[2]. En aquest projecte s'ha utilitzat el plugin de Subclipse.

Subclipse es un mòdul que integra Subversion (SVN) a l'entorn Eclipse. En el següent apartat s'explicarà amb més detall en que consisteix i per a què s'utilitza.

8.1.2 Subversion

Subversion o SVN [30, 13] és un sistema de control de versions de codi obert. Guarda un historial dels canvis fets en els arxius i carpetes al llarg del temps i, per això, permet consultar en qualsevol moment versions antigues d'un fitxer o, fins i tot, retrocedir a una versió determinada.

[1] Un entorn integrat de desenvolupament o IDE és una aplicació informàtica que conté eines que faciliten el desenvolupament de programari.

[2] Un plug-in és una aplicació informàtica que interactua amb una altra aplicació per aportar-li una funcionalitat addicional

Subversion pot funcionar a través d'una xarxa i gestionar situacions en les que més d'un usuari estiguin modificant els mateixos fitxers, ajudant a resoldre els conflictes que puguin aparèixer.

Alguns dels conceptes principals de Subversion són:

- Repositori (*repository*). És el magatzem on es guarden tots els fitxers i les seves versions. Totes les operacions situen el punt de referència en el repositori: per exemple, fent un *check-out* (extreure un fitxer del repositori) o un *commit* (incorporar un fitxer dins del repositori).
- Còpia de treball (*working copy*). És el directori que conté els fitxers sobre els que estem treballant. Es poden tenir tantes còpies de treball com es vulguin per un repositori.
- Sincronització. És la clau per portar el control de versions i l'objectiu que es vol aconseguir: tenir la nostra còpia de treball en sincronia amb el repositori. Si es modifica la còpia de treball, caldrà actualitzar el repositori. I si el repositori ha estat modificat per un altre membre, en el cas que hi hagi un grup de programadors, caldrà que ens descarreguem els canvis a la nostra còpia de treball.

En aquest projecte, tot i que només hi ha un programador, s'ha considerat interessant l'ús d'un sistema de control de versions per tal de guardar un historial de cadascun dels canvis fets sobre el projecte i poder consultar versions antigues en cas necessari.

S'ha utilitzat el repositori Subversion ubicat al servidor svn.fib.upc.es que la Facultat d'Informàtica de Barcelona posa a l'abast de cada alumne. El fet de tenir el repositori en un servidor apart també fa la funció de còpia de seguretat en cas de tenir algun problema amb la còpia o, de forma més general, amb l'ordinador sobre el que es treballa.

8.1.3 Firebug

Firebug [31, 13] és un plug-in o extensió de codi obert pel navegador Mozilla Firefox. Posa a l'abast funcionalitats útils per al desenvolupament web com editar, depurar i monitoritzar CSS, HTML i JavaScript en qualsevol pàgina web i veure'n els resultats de forma immediata. També esdevé una eina molt útil per detectar i solucionar errors, ja que dona informació detallada dels problemes que es troba en JavaScript, CSS i XML.

Cal tenir en compte que aquesta eina la pot utilitzar tothom i que, per tant, qualsevol usuari pot modificar el que vulgui del CSS, HTML o JavaScript d'una pàgina i enviar-ho a servidor per mitjà d'un *submit* d'un formulari. Si a servidor no es fessin les suficients comprovacions, es podrien tenir problemes importants de seguretat.

8.1.4 Luke

Luke [32, 13] és una eina de desenvolupament i diagnòstic de codi obert que permet accedir a índexs de Lucene ja existents, i mostrar-los i modificar el seu contingut de diverses maneres:

- Buscar per número de document o per terme
- Veure els documents

- Recuperar una llista ordenada dels termes més freqüents
- Executar una consulta i veure'n els resultats
- Analitzar els resultats de la cerca
- Eliminar documents concrets de l'índex

Aquesta eina ha sigut útil per a poder inspeccionar els índexs que genera l'aplicació per mitjà de Lucene i treballar sobre ells.

8.1.5 Log4j

Els *logs* poden proporcionar informació sobre el context precís d'una execució de l'aplicació. Un dels avantatges de la generació dels *logs* és que, una vegada inserit el codi necessari, no requereix intervenció humana. A més, es poden guardar *logs* de sortida en mitjans persistents per tal d'estudiar-los posteriorment. Apart de ser útil al llarg del cicle de desenvolupament, també pot ser vist com una eina d'auditoria.

Log4j[33, 13] es una llibreria *open source* que ofereix funcionalitats per a la generació de *logs*. Amb log4j és possible habilitar la generació de *logs*, així com escollir la sortida i el nivell de granularitat, en temps d'execució sense necessitat de modificar l'aplicació binària. El paquet log4j està dissenyat perquè aquestes instruccions puguin romandre en el codi final sense incórrer en un cost de rendiment pesat. El comportament de la generació de *logs* es pot controlar fàcilment mitjançant l'edició d'un arxiu de configuració.

8.2 Detalls d'implementació

8.2.1 Indexació i cerca amb Lucene

En primer lloc veurem les etapes del procés d'indexació amb Lucene. En la figura 8.1 es pot veure un esquema d'aquest procés.

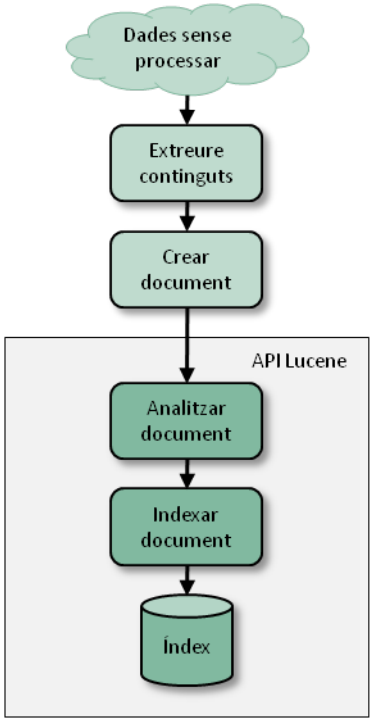


Figura 8.1: Procés d'indexació amb Lucene

El primer pas és obtenir les dades que es volen indexar i crear documents de Lucene a partir dels seus continguts (possiblement calgui extreure el text de documents binaris, ja que Lucene només treballa amb text). En el nostre cas obtenim el títol, resum i article complet de cada presentació i creem un document de Lucene per a cada una.

Cal destacar que per a obtenir el text de l'article complet s'ha hagut d'extreure de l'arxiu de la presentació tenint en compte el seu format. Si el contingut és textual amb una codificació estàndard coneguda, aquest treball és simple. No obstant, avui en dia molts document solen ser de tipus binari (PDF, Microsoft Office, Open Office, etc.) o contenen etiquetes o *markups* que cal eliminar abans d'indexar (XML, HTML, etc.). Aquest tema es tractarà amb més detall en la secció 8.2.2

Finalment, s'indexen els documents de Lucene. Durant el procés d'indexació, Lucene processa els documents i analitza el seus continguts per extreure termes i emmagatzemar-los en l'estructura de dades anomenada índex invertit.

D'altre banda tenim el procés de cerca amb Lucene. En la figura 8.2 es pot veure un esquema de les etapes d'aquest procés.

L'usuari interactua amb la interfície d'usuari, a través de la qual envia les peticions de cerca. Com que en aquest cas no estem implementant una cerca de l'usuari, sinó un recomanador, el text de la consulta es construeix programàticament després en rebre la petició de l'usuari per obtenir recomanacions. A continuació es processa la consulta textual i es tradueix a un format comprensible per Lucene. Finalment, s'executa la consulta i es mostren els resultats a l'usuari.

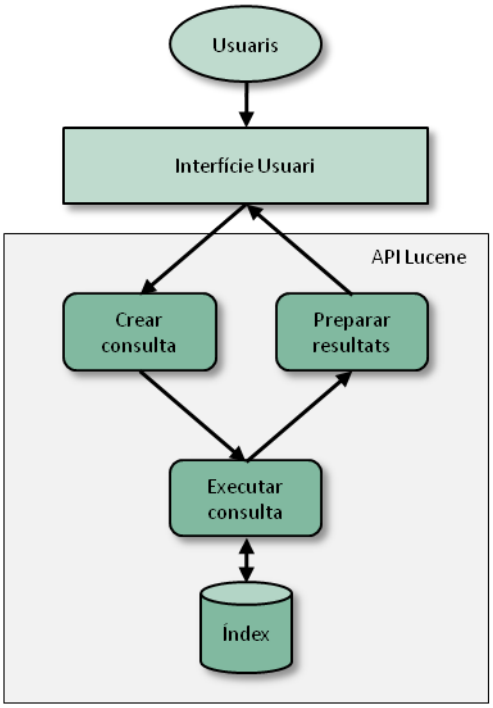


Figura 8.2: Procés de cerca amb Lucene

En l'annex B Implementació es poden veure detalls addicionals sobre la implementació d'aquests processos.

8.2.2 Extracció de text amb Tika

Els requisits del sistema especificaven que s'han d'admetre diferents formats en els arxius de les presentacions però, tal com s'ha vist anteriorment, Lucene només treballa amb text pla. Per aquest motiu cal extreure el text dels arxius prèviament. A nivell d'implementació, això implica reunir un conjunt de descodificadors per a poder tractar cadascun dels formats. A més, cal detectar el format de cada arxiu per tal d'utilitzar el descodificador pertinent.

Per a solventar aquest problema s'ha utilitzat **Apache Tika**. Tika és una eina per a la detecció de format i extracció de continguts de documents. És un projecte de la Apache Software Foundation que néixer com a subprojecte d'Apache Lucene.

Tika proporciona una interfície de programació d'aplicacions (API) que es pot utilitzar per a detectar el tipus de contingut d'un document i també analitzar el contingut textual i metadades de varis formats de documents. No tracta tota la varietat de formats de documents diferents ell mateix, sinó que delega la feina a diverses llibreries de *parsing* o anàlisi sintàctic existents com per exemple Apache POI, PDFBox o Neko HTML. Actualment, Tika suporta uns trenta formats de documents d'entre els quals es destaquen:

- HyperText Markup Language (HTML).
- XML i formats derivats.
- Documents de Microsoft Office: MS Word, MS Excel, MS PowerPoint.

- Open Document Format (ODF), format per defecte de la suite ofimàtica OpenOffice.org.
- Portable Document Format (PDF).
- Electronic Publication Format (EPUB), utilitzat per a molts llibres digitals.
- Rich Text Format (RTF).
- Formats de compressió i arxivament (ar, cpio, tar, zip, gzip, bzip2).
- Formats de text

La gran idea darrera de Tika és que oferix una interfície genèrica per analitzar múltiples formats. La API de Tika oculta les diferències tècniques de les implementacions dels diversos *parsers*. Internament Tika delega el treball a la llibreria del format pertinent i adapta els resultats obtinguts. Si no es coneix el format del document prèviament, Tika ofereix la possibilitat de detectar el format del document d'entrada basant-se en heurístiques i utilitzar aquesta informació per a delegar el treball a la llibreria corresponent.

D'aquesta manera, Tika permet resoldre l'extracció de text de documents de diferents formats d'una forma senzilla i eficaç, complint amb els requisits del sistema.

8.2.3 Xifrat de la contrasenya d'usuari

Quan un usuari es registra, se li demana una contrasenya mitjançant la qual podrà autenticar-se en el sistema. Al crear-se l'usuari en la base de dades, per motius de seguretat no es guarda la contrasenya directament ja que, en cas que la seguretat del sistema es vegi compromesa, usuaris malintencionats podrien obtenir les contrasenyes dels altres usuaris. Encara que es confiés plenament de la seguretat de la base de dades, tampoc seria bona idea guardar la contrasenya explícitament, ja que els administradors de la base de dades podrien accedir-hi. Per aquest motiu, cal utilitzar algun sistema per guardar la contrasenya de forma encriptada.

Aquest escenari és un candidat perfecte per a la "one-way hash encryption" o funció de hash criptogràfica també anomenada *message digest* . És conegut com "one-way" ja que es pot calcular la clau o *hash* que representa un text donat, però a partir de la clau no es pot esbrinar el text que l'ha generat. Aquest també és un mecanisme lliure de col·lisions que garanteix que no hi ha dos valors diferents que produiran una mateixa clau.

Actualment, hi ha diversos algorismes de hash criptogràfic utilitzats àmpliament, com MD5 i SHA-1. SHA-1 (Secure Hash Algorithm 1) és més lent que MD5, però el *hash* del missatge és més gran, la qual cosa el fa més resistent a atacs de força bruta. Per això, s'ha preferit Secure Hash Algorithm (160 bits) per sobre de MD5 (128 bits). Hi ha altres variants més potents, però no s'ha cregut necessari per aquest projecte.

A l'hora registrar un nou usuari i guardar-lo a la base de dades, no es guarda amb la seva contrasenya original, sinó amb el valor del seu hash amb SHA-1. Així, quan un usuari registrat decideix identificar-se en el sistema amb el seu identificador i contrasenya, l'aplicació obté el *hash* de la contrasenya introduïda i el compara amb el valor emmagatzemat amb la base de dades. El *login* és vàlid només si els valors coincideixen.

8.2.4 Validacions

Gràcies al mòdul de validacions en el servidor, es poden validar diversos criteris sobre una determinada variable d'una manera fàcil. Aquest mòdul de validació es fàcilment extensible per afegir nous tipus de validacions i, al ser un mòdul independent, es podrà reutilitzar en el futur ja que la validació és una tasca comuna en moltes de les aplicacions.

En el llistat de codi 8.1 es mostra un exemple de la validació que es duu a terme en el formulari d'introducció de dades d'una presentació. En concret, es tracta de la validació del camp del títol i es comprova que no estigui buit, ja que s'ha marcat com a obligatori, i que la dimensió de la cadena no excedeix el màxim permès. De la mateixa manera es podrien concatenar altres criteris a validar, com per exemple comprobar si té una dimensió mínima, si compleix una determinada expressió regular, etc.

Llistat de codi 8.1: Exemple de validació

```
StringValidator val = new StringValidator("title",title, REQUIRED,
    maxLength(Presentation.TITLE_LENGTH));
if(!val.parseAndValidate()) {
    //Validation error
    //val.getError() returns the validation error key
}
```

Aquests errors es retornaran al navegador en format JSON amb la següent estructura:
`[{‘‘camp 1’’: ‘‘missatge d’error 1’’},{‘‘camp 2’’: ‘‘missatge d’error 2’’}]`

Posteriorment, per mitjà de JavaScript, es mostraran els missatges d'error en els camps del formulari que es corresponguin. En la figura 8.3 es poden veure exemples d'errors de validació en el formulari de *login*.

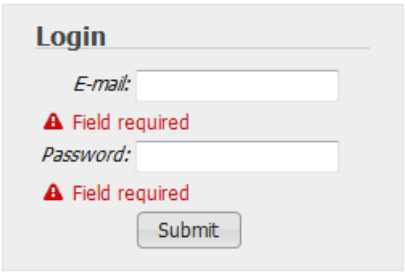


Figura 8.3: Exemple de formulari amb missatges d'error de validació

8.2.5 Localització

Tots els textos del software s'han definit en un fitxer `.properties` de Java, independent de la resta del codi que implementa les vistes. Aquest fitxer és bàsicament un fitxer de text en el que en cada línia es guarda un atribut amb el format `key=value`. D'aquesta manera, si cal modificar el vocabulari utilitzat o traduir-ho a altres idiomes no caldria revisar i modificar tot el codi.

En el llistat 8.2 és pot veure un petit extracte del fitxer `language.properties` que s'ha utilitzat com a diccionari de l'aplicació.

Llistat de codi 8.2: Extracte de `language.properties`

```
project.name = Conference RecSys
setup = Setup wizard
setup.appProperties = Application properties
setup.adminAccount = Adminsitrator account
```

8.2.6 Components de la interfície gràfica

La llibrera JavaScript de **jQuery**, a part de simplificar el desenvolupament gràcies a les seves funcionalitats, ens ha permès enriquir visualment l'aplicació. Concretament, gràcies als *widgets* personalitzables de **jQuery UI** s'han generat les alertes, diàlegs, selectors de dates, botons, icones, etc. amb un tema comú i fàcilment canviable. A les figures 8.4 i 8.5 es poden veure alguns dels resultats. En el primer cas es tracta d'una finestra de diàleg modal que evita que l'usuari interactuï amb la resta de la pàgina fins es tanqui i, en el segon cas, podem veure un selector de dates per tal d'ajudar a l'usuari en la introducció de dades.

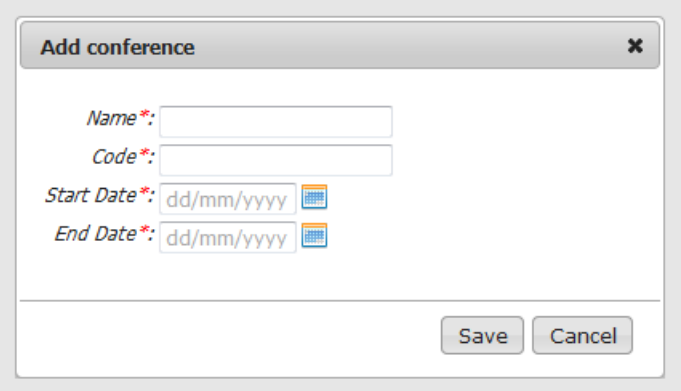


Figura 8.4: Diàleg modal

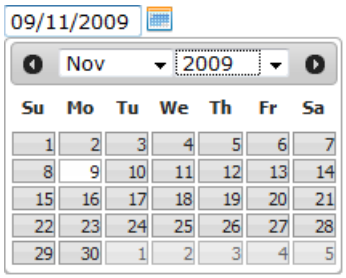


Figura 8.5: Selector de dates

A continuació es mostren altres components o detalls de la interfície gràfica implementats amb jQuery i plugins complementaris.

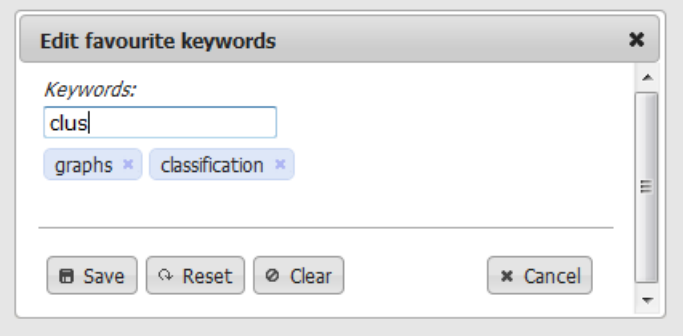


Figura 8.6: Editor de tags

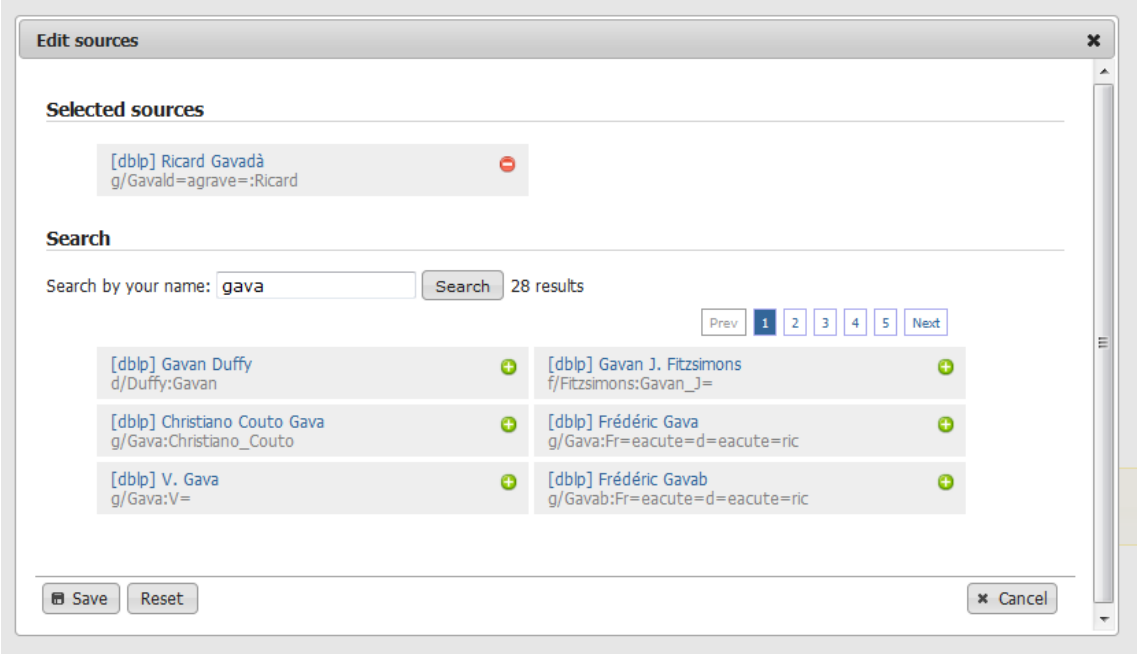


Figura 8.7: Panell d'edició de recursos externs amb funcionalitat de cerca asíncrona i paginació.

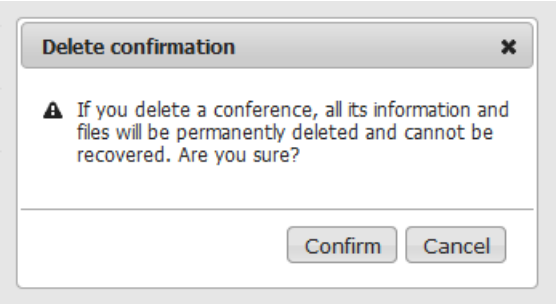


Figura 8.8: Exemple de diàleg amb missatge d'alerta

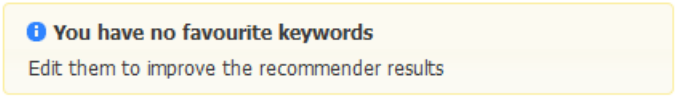


Figura 8.9: Exemple de missatge informatiu

8.2.7 Problemes amb els navegadors

Volíem que l'aplicació fos *multi-browser* i compatible amb els navegadors més utilitzats actualment. No obstant, és totalment conegut el fet que una maquetació web es pot veure molt diferent en diferents navegadors, ja que no tots donen suport als estàndards web de la mateixa manera.

S'han tingut alguns problemes per tal d'aconseguir que l'aplicació web es veies bé en en Firefox, Internet Explorer i Chrome, que són els navegadors més utilitzats [27, 28] i els que s'han tingut en compte. Finalment s'ha aconseguit, tot i que en Internet Explorer es segueixen mostrant algunes diferències de visualització. Per exemple en Internet Explorer els botons no es veuen amb les cantonades arrodonides.

Capítol 9

Proves i experiments

9.1 Especificació del pla de proves

Es duran a terme una sèrie de proves per tal de verificar el correcte funcionament del software i, determinar així, si compleixen amb els requisits fixats. Les proves no només es faran un cop estigui tot finalitzat, sinó que al llarg del desenvolupament s'aniran fent les proves pertinents de les noves funcionalitats implementades.

Les proves s'agrupen en dos blocs:

- Proves generals del software
- Proves específiques del sistema de recomanació

9.1.1 Proves generals del software

En aquest cas, es provarà que totes les funcionalitats de l'aplicació actuïn correctament i amb un temps de càrrega acceptable. Aquestes proves no s'especificaran a la memòria ja que simplement es seguiran els casos d'ús del sistema, tractant de passar per tots els cursos alternatius possible per tal de garantir que es tracta d'una aplicació robusta i que compleix els requisits funcionals i no funcionals.

9.1.2 Proves específiques del sistema de recomanació

D'altra banda, en aquest segon bloc de proves, es tractarà de forma més específica les funcionalitats més directament relacionades amb el sistema de recomanació, en les que cal realitzar experiments per tal d'acabar de prendre decisions i refinar els paràmetres del sistema:

- Indexació de presentacions
  - Anàlisi en l'indexació. Experiments amb diferents analitzadors.
  - Duració del procés d'indexació
- Obtenció de dades dels usuaris en recursos externs

  - Revisió del procés d'obtenció de dades.

- Cerca
  - Consulta generada per l'usuari. Determinació dels paràmetres α i β.
  - Duració del procés de cerca.
- Avaluació dels resultats

Tot i que s'han fet proves amb diferents dades, s'analitzaran de forma detallada algunes de les proves realitzades amb les dades de la conferència real ECML PKDD 2009. A continuació es presenten les observacions dels resultats obtinguts amb les proves i experiments. Per a més informació, en l'annex C es poden veure aquestes proves detallades així com alguns exemples de recomanacions per a usuaris concrets.

Observacions

Dintre dels analitzadors que Lucene porta incorporats, s'ha vist que el *StandardAnalyzer* és el que més s'ajusta a les nostres necessitats ja que no separa només per espais o signes de puntuació sinó que utilitza una mica de lògica per a identificar certs tipus de tokens que no s'haurien de separar com ara sigles o acrònims separats per punts, correus electrònics, direccions IP, URLs, etc. No obstant s'ha vist que la llista de paraules buides que ve per defecte segueix deixant passar masses paraules que no ens aporten informació sobre els continguts i que no ens interessen pel tipus de consulta que volem fer. D'aquesta manera, s'ha considerat necessari l'ús d'una llista de paraules buides més completa.

D'altra banda, s'ha observat que el procés d'indexació es duu a terme en un temps raonable. Una conferència que es considera gran, amb 136 presentacions en total, ha tardat menys de dos segons en indexar els continguts de totes les presentacions, incloent el títol, resum i text extret del fitxer adjunt. Aquest temps també inclou el temps de *parsing* dels fitxers per extreure el contingut textual.

Un dels problemes que s'ha trobat és el temps que es tarda en recopilar la informació de l'usuari en els repositoris externs amb els que s'ha identificat i, per això, s'han introduït els següents límits:

- Si l'usuari té més de 20 publicacions, només es recuperen les últimes 20.
- Si l'usuari té més de 20 coautors, només es recuperaran els 20 que comparteixin més publicacions.
- De cada coautor es recuperaran un màxim de 5 publicacions (que no es repeteixin amb les del propi autor, ni amb altres coautors recuperats anteriorment).

Tal com s'ha vist en el capítol de disseny del sistema de recomanació, la consulta es genera a partir de tres grups d'informació: paraules clau introduïdes per l'usuari, paraules extretes de les seves publicacions i paraules extretes de les publicacions dels seus coautors. També hem vist que se li vol donar un pes diferent a cadascun d'aquests grups en funció dels paràmetres α i β. De tal manera que el pes de les paraules clau introduïdes per l'usuari sigui α, el pes de les seves publicacions sigui $(1-\alpha)\beta$ i el de les publicacions dels coautors $(1-\alpha)(1-\beta)$. Després de realitzar algunes proves s'ha decidit utilitzar els valors $\alpha = 50\%$ i $\beta = 70\%$.

Capítol 10

Documentació

En el CD del projecte s'adjunta documentació addicional del projecte. Hi ha documentació destinada a l'usuari final (*Installation guide*, *Administrator's guide* i *User's guide*), així com documentació tècnica. Es pot trobar en la carpeta `documentation`, fitxer `index.html`.

Figura 10.1: Índex de la documentació

La documentació tècnica és un enllaç a la documentació del codi font del projecte generada amb Doxygen. Doxygen és un generador de documentació per C++, C, Java, Objective-C, Python, IDL, entre d'altres. La documentació s'extreu directament a partir del codi font i els comentaris dins d'aquest, cosa que permet mantenir fàcilment la consistència entre la documentació i el codi. També permet utilitzar l'eina Graphviz per generar automàticament diagrames en la documentació. A continuació es mostren dos captures de la documentació que s'ha generat en format HTML.

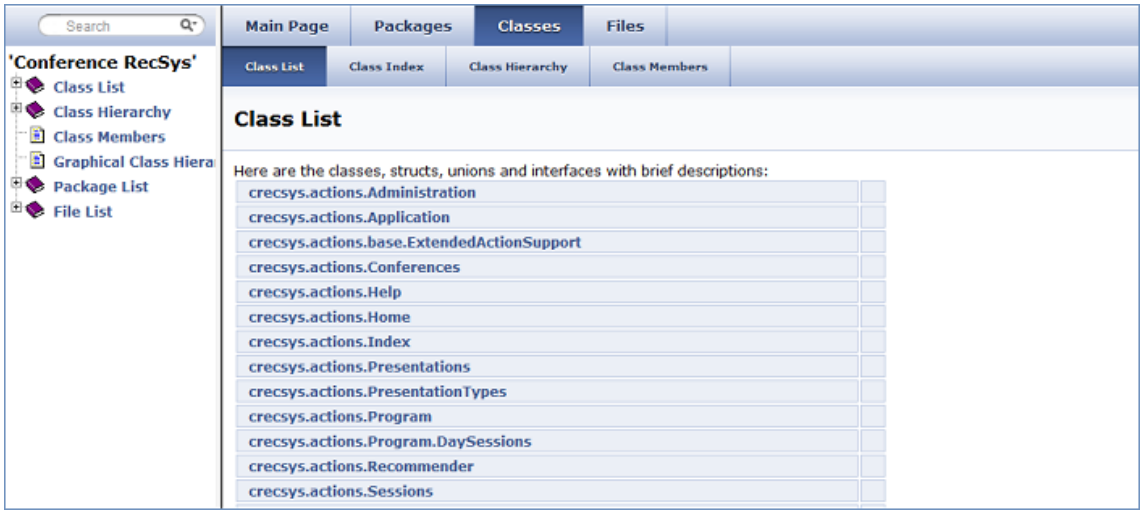


Figura 10.2: Documentació del codi. Llista de classes

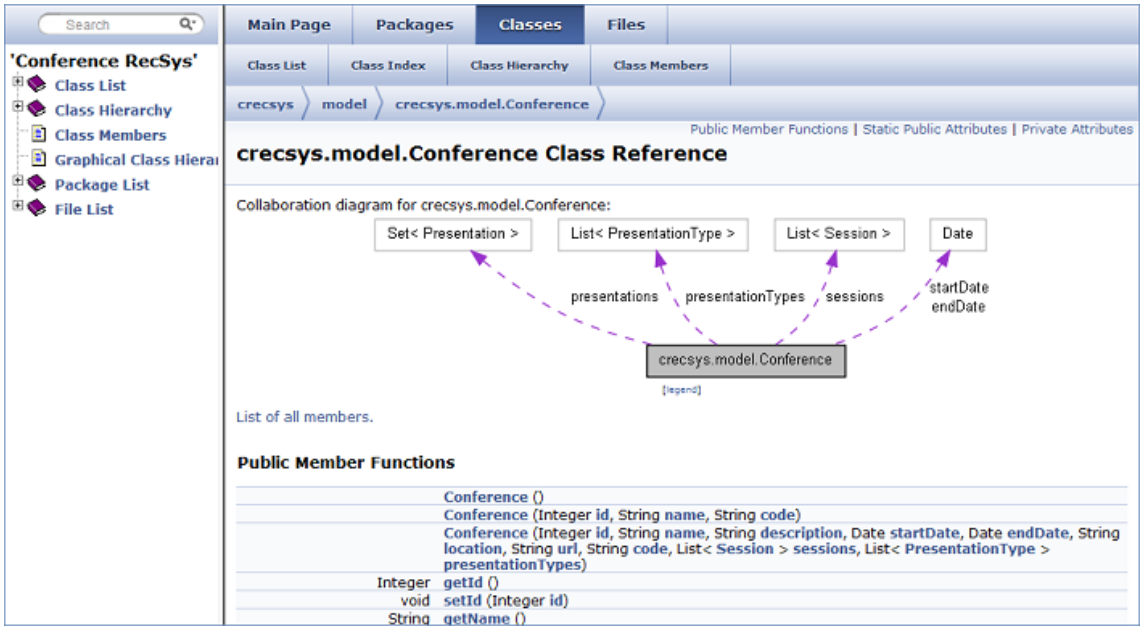


Figura 10.3: Documentació del codi. Referència d'una classe determinada

Capítol 11

Conclusió

11.1 Objectius assolits

Es considera que s'han assolit els objectius inicials del projecte. D'una banda, l'aplicació resultant compleix els requisits funcionals que es van establir en la fase d'anàlisi del projecte. S'ha obtingut una aplicació que permet gestionar conferències i els seus continguts per part dels administradors i que, d'altra banda, permet que els usuaris externs puguin consultar-les, així com omplir el seu perfil per tal d'obtenir recomanacions. També s'ha elaborat la documentació corresponent de l'aplicació de cara a l'usuari final.

Pel que fa al recomanador, cal destacar que es tracta d'un prototip, tal i com es va explicar en l'abast del projecte. Es va fer un estudi de diferents tècniques que es poden utilitzar en l'àmbit dels sistemes recomanadors, així com de l'àmbit de la recuperació de la informació. A partir dels coneixements que es van obtenir d'aquest estudi, es va conceptualitzar el sistema recomanador de ponències que s'ha acabat desenvolupant. No obstant, caldria fer-ne proves addicionals i estudiar-ne més detingudament els resultats per tal de refinar l'algoritme.

Pel que fa als requisits no funcionals, també s'han complert la majoria dels objectius. S'ha aconseguit dissenyar una interfície força intuïtiva i agradable pels usuaris. La configuració de l'aplicació per part de l'organitzador és prou senzilla, tret d'algunes configuracions manuals necessàries. També s'ha aconseguit l'objectiu de dissenyar una aplicació modular, per tal que sigui fàcilment extensible i reutilitzable.

La resposta a l'usuari extern és prou ràpida, tret d'una excepció. S'han detectat problemes amb el temps que tarda a l'hora de recuperar informació externa de l'usuari, donat que això es tradueix a un gran nombre consultes en repositoris externs i el conseqüent processat dels resultats. En l'apartat de millores es proposa una solució que permetria reduir el temps d'algunes d'aquestes consultes, millorant el temps total de resposta en aquest cas.

Tot i que estem satisfets amb el producte resultant, al llarg del desenvolupament s'han anat observant diversos aspectes que ajudarien a millorar l'aplicació, alguns de les quals ja s'han anat comentant. En la següent secció es veuran amb més detall.

11.2 Revisió de la planificació

Finalment hi ha hagut una lleugera desviació en les hores planificades del projecte. A continuació es detallen les variacions:

- L'especificació es va finalitzar en 2 dies (10 hores) menys del que es va planificar.
- Es va ampliar el temps dedicat a la conceptualització i disseny del sistema recomanador en 6 dies (30 hores) més per tal de refinar el procés.
- L'implementació va durar 10 dies (50 hores) més degut a la corba d'aprenentatge de les tecnologies.

En conclusió, s'han dedicat 14 dies més del que s'havia planificat que, sobre el total de 224 dies estimats (contant també les tasques pròpies del PFC), suposa una desviació d'un 6% aproximadament.

A continuació es mostra la revisió dels costos estimats amb les noves dades temporals.

Fase	**Perfil**	**Hores**	**Preu**	**Total**
Definició del projecte	Director	25	35 €/hora	875 €
	Analista	15	25 €/hora	375 €
Formació i aprenentatge en sistemes de recomanació	Analista	20	25 €/hora	500 €
	Arquitecte	30	25 €/hora	750 €
Especificació	Analista	40	25 €/hora	1000 €
Disseny	Arquitecte	280	25 €/hora	7000 €
Implementació	Programador	460	15 €/hora	6900 €
	Dissenyador web	15	15 €/hora	225 €
Proves	Programador	50	15 €/hora	750 €
Documentació	Programador	50	15 €/hora	750 €
TOTAL	-	**985**	-	**19125€**

Taula 11.1: Revisió de costos

11.3 Possibles ampliacions i millores

A continuació es mostren les funcionalitats que, des de l'inici, ja han quedat fora de l'abast del projecte, així com algunes de les possibles ampliacions i millores que han anat sorgint al llarg del projecte però que, per limitacions temporals, no s'han inclòs en el desenvolupament i queden com a propostes de treball futur. S'han volgut exposar totes les idees ja que aquest projecte ofereix moltes possibilitats de ser ampliat i créixer.

11.3.1 Recomanacions

Tal com ja s'ha explicat en la secció anterior, el recomanador és un prototip i no és definitiu sinó que caldria fer-ne proves addicionals i estudiar-ne més detingudament els resultats per tal de refinar-lo. Caldria acabar d'ajustar els pesos que s'assignen a cada conjunt de termes ja sigui a

l'hora d'indexar les presentacions, com a l'hora de fer la consulta amb la informació extreta de l'usuari.

D'altre banda, de moment només s'utilitza un repositori extern, DBLP, del que s'extreu informació de les publicacions de l'usuari. Així que una de les millores seria afegir noves fonts de les que extreure informació en el mòdul de tractament de recursos externs, així com afegir les classes per processar la informació de les noves fonts. Gràcies a la modularitat, només caldrà tocar el mòdul pertinent sense que la resta del sistema es vegi afectat.

Donat que en les recomanacions poden intervindre dades obtingudes automàticament pel sistema, pot ser que la consulta per obtenir els resultats sigui bastant àmplia i retorni resultats diferents entre si. D'aquesta manera, un usuari podria estar interessat especialment en algun dels resultats i podria interessar-li obtenir recomanacions més concretes de presentacions similars a la seleccionada. Això es podria considerar per a una altra ampliació del sistema. Per exemple, es podria afegir un enllaç al costat de cada resultat del recomanador per tal de fer una cerca del tipus "More like this". També es podria incloure aquest enllaç en la vista detallada d'una presentació.

Una altra proposta de millora seria tenir en compte, també, les interaccions dels usuaris a l'hora de fer les recomanacions. Per exemple, enregistrar sobre quines presentacions ha clicat per veure'n la informació detallada, quins resultats de les recomanacions ha obert o afegir la funcionalitat de poder marcar favorits per tal d'intentar extreure els seus gustos amb la màxima informació possible. En el cas de que s'implementés la funcionalitat de cerca "More like this", també suposaria una nova font de dades sobre les interaccions de l'usuari que podria aportar informació útil sobre els seus gustos.

També queda com a ampliació la funcionalitat de generar un horari coherent temporalment amb un conjunt de presentacions recomanades, intentant maximitzar la puntuació del conjunt i que compleixin una sèrie de restriccions temporals com podrien ser un número màxim d'hores o un número màxim de presentacions o dies prohibits, entre d'altres. Donat que en la versió actual del sistema s'ha intentat generalitzar per tal d'adaptar-se a les màximes situacions, hi ha dades, com per exemple l'hora de fi d'una presentació, que no són obligatòries però que per aquesta funcionalitat serien necessàries. Per a resoldre aquest problema es podria afegir un camp de control a la conferència per tal que l'administrador pugui escollir si vol aquesta funcionalitat del recomanador estigui disponible. En aquest cas, s'haurien de fer obligatoris tots els camps sobre les dates i hores de les sessions i presentacions, i validar que s'omplen bé.

11.3.2 Seguretat

La seguretat de l'aplicació és un tema millorable. La contrassenyes es guarden xifrades en la base de dades i en el servidor es fan moltes comprovacions per evitar problemes de seguretat, sobretot de les dades enviades des dels navegadors, però es podria considerar emprar connexions més segures per al *login* dels usuaris, per exemple mitjançant una connexió xifrada amb SSL.

11.3.3 Temps de resposta

Un dels problemes que s'ha trobat és el temps que es tarda en recopilar la informació de l'usuari en els repositoris externs ja que cada consulta es tradueix en un gran nombre de peticions a una

font externa, seguit del seu processat per extreure'n les dades.

En el cas de la font externa DBLP, s'utilitzaven crides a la seva API que retornaven respostes en format XML. Donat que ens interessa reduir el temps de resposta, una solució seria descarregar inicialment el fitxer XML sencer que conté tots els nodes bibliogràfics de DBLP, analitzar el fitxer, extreure totes les dades que ens poden interessar i guardar-les en el format que interessi, com per exemple com a noves taules en la base de dades del nostre sistema. D'aquesta manera, podríem estalviar-nos la connexió a aquest recurs extern i el processat del resultat de cada petició. En aquest cas, caldria afegir la funcionalitat de poder actualitzar la informació dels recursos externs dins del sistema.

11.3.4 Usabilitat

Una de les ampliacions que es podrien fer per millorar notablement la usabilitat seria afegir una nova funcionalitat al sistema per tal de poder fer una càrrega inicial dels continguts d'una conferència a través d'un fitxer de text. D'aquesta manera, podria estalviar-se l'haver d'afegir cada element un a un. Caldria estudiar quin format establir per tal que fos el més pràctic possible per a l'usuari final. Un cop feta la primera càrrega de dades, podria utilitzar les funcionalitats ja desenvolupades per fer les modificacions necessàries.

També queda com a futura ampliació, afegir funcionalitats relacionades amb la identificació de l'usuari en el sistema, com per exemple no tancar sessió, o un mètode per recordar o restablir la contrasenya, en cas d'haver-la oblidat.

També queda pendent com a millora, afegir la funcionalitat de poder filtrar les diferents llistes que es mostren en l'aplicació segons diferents paràmetres de cerca per tal de poder trobar el que es busca d'una forma més ràpida i també la de poder seleccionar en funció de quina columna es volen ordenar els resultats. Actualment només es permet filtrar les llistes de les presentacions i només en funció de la sessió a la que pertanyen. Es podria ampliar aquest filtre per poder cercar, per exemple, sobre el nom, la data, el tipus de presentació, paraules clau i si té fitxer associat o no. De la mateixa manera es podrien afegir filtres també en la llista de conferències, en la llista de sessions, etc.

El sistema permet tenir diverses conferències en la mateixa aplicació. No obstant, en els casos que l'aplicació fos utilitzada per conferències bastant espaiades temporalment, podria ser interessant que l'administrador pogués determinar quina es la conferència activa en aquest moment per tal que en accedir a l'aplicació es seleccionés aquesta automàticament. D'aquesta manera, un usuari extern s'estalviaria l'accés a l'índex de conferències.

11.3.5 Configuració

En aquesta primera versió del sistema s'ha utilitzat una base de dades incorporada en la pròpia aplicació però, donades les característiques del sistema, es podria canviar fàcilment la base de dades a utilitzar. Per això, es proposa com a ampliació documentar els passos per a poder connectar l'aplicació a una base de dades externa com podria ser MySQL així com traduir el disseny físic de la base de dades per ajustar-la a la nova tecnologia. D'aquesta manera els usuaris avançats tindran la opció de connectar-se a una base de dades en un servidor independent.

11.3.6 Altres

Una altra proposta és fer el sistema més amigable a un usuari identificat afegint una petita secció personalitzada dins la pàgina principal que podria incloure un missatge de benvinguda, així com enllaços recomanats segons el perfil i el seu estat. Per exemple, en el cas d'un usuari extern, el podria animar a omplir el perfil en cas que no ho hagués fet, en cas que el sistema tingués suficients dades podria mostrar un número reduït de presentacions recomanades i un enllaç a la secció del recomanador, també podria mostrar *quick tips* o consells ràpids, notícies, etc. En el disseny de la pàgina principal ja s'ha tingut en compte l'espai per aquesta secció.

També es podria afegir la funcionalitat de que els administradors poguessin enviar notícies o notificacions als usuaris que ho desitgessin per mitjà del correu electrònic. Un usuari es podria subscriure en el seu perfil a una o vàries conferències per tal de poder rebre els missatges d'aquesta conferència particular.

11.4 Conclusions personals

En aquest projecte s'han pogut posar en pràctica molts dels coneixements adquirits en diverses assignatures cursades al llarg de la carrera d'Enginyeria Informàtica. Aquí s'han pogut utilitzar conjuntament coneixements que, fins ara, només s'havien posat en pràctica per separat en cadascuna de les assignatures. Es destaquen els coneixements d'enginyeria del software, coneixements de bases de dades, coneixements de programació i coneixements de l'àmbit de la recuperació de la informació. L'assignatura optativa Recuperació de la Informació va ser una de les assignatures cursades que em va semblar més interessant i va ser el motiu que va desencadenar que acabés triant aquest projecte.

En un principi vaig valorar la possibilitat de realitzar el Projecte de Final de Carrera en l'empresa on treballo, cosa que hagués resultat més còmode i, segurament, m'hagués permès acabar molt més ràpid. No obstant, vaig decidir prendre'm el projecte com una possibilitat per aprendre sobre algun tema que em motivés i m'interessés especialment. Aquesta decisió va ser la que em va portar a escollir un projecte on pogués aplicar algun coneixement relacionat amb la recuperació de la informació. Tot i així, no em volia centrar només en l'estudi o implementació de tècniques o algoritmes d'aquest àmbit, sinó que m'interessava també el fet de poder incloure-ho en una aplicació sencera utilitzable per un usuari final. Per aquests motius vaig triar el projecte que al final he dut a terme. Tot i que m'ha suposat més hores i dedicació, és un projecte que m'he pres amb molta il·lusió i valoro molt positivament haver-me decantat per aquesta opció.

El fet de dur a terme el projecte fora d'una empresa m'ha permès tenir molta llibertat a l'hora d'escollir, comparar i prendre les decisions que m'han semblat més oportunes pel projecte, com per exemple a l'hora de decidir les tecnologies.

Gràcies a aquest projecte, també he pogut aprofundir en l'aprenentatge d'unes tecnologies en les que fins ara els meus coneixements eren molt bàsics (com per exemple Hibernate, JavaScript, AJAX i Lucene) o no dels que no tenia coneixements (com el *framework* Struts 2 i Sitemesh).

Finalment, el balanç personal que n'obtinc en acabar aquest projecte és que ha estat una experiència molt positiva i enriquidora.

Bibliografia

[1] Satnam Alag.
Collective Intelligence in Action.
Manning. 2009.

[2] Donald Brown, Chad Michael Davis, Scott Stanlick.
Struts 2 in Action.
Manning. 2008.

[3] Dolors Costal, Xavier Franch, M. Ribera Sancho, Ernest Teniente.
Enginyeria del Software. Especificació.
Edicions UPC. 2005.

[4] C. Gòmez, E. Mayol, A. Olivé, E. Teniente
Enginyeria del Software. Disseny I..
Edicions UPC. 2005.

[5] Dietmar Jannach, Markus Zanker, Alexander Felfernig, Gerhard Friedrich.
Recommender Systems: An Introduction.
Cambridge University Press. 2010.

[6] Michael Ley.
DBLP - DBLP XML Requests. Appendix to the paper "DBLP - Some Lessons Learned"
2009.
http://dblp.uni-trier.de/xml/docu/dblpxmlreq.pdf

[7] Michael Ley.
DBLP - Some Lessons Learned.
Proceedings of the VLDB Endowment. 2009.
http://dblp.uni-trier.de/xml/docu/dblpxml.pdf

[8] C.D. Manning, P. Raghavan, H. Schütze.
An Introduction to Information Retrieval, online edition.
Cambridge UP. 2009.

[9] Raymond J. Mooney, Loriene Roy.
Content-Based Book Recommending Using Learning for Text Categorization.
Proceedings of the SIGIR-99 Workshop on Recommender Systems:Algorithms and Evaluation.
1999.

[10] Otis Gospodnetic, Erik Hatcher.
Lucene in Action. Second Edition.
Manning. 2010.

[11] Apunts de l'assignatura *Recuperació de la Informació.*
FIB. UPC. 2009.

[12] Unified Modeling Language (UML).
http://www.uml.org/

[13] Wikipedia.
Seccions tecnològiques.
http://www.wikipedia.org

[14] Java Technology.
http://www.oracle.com/us/technologies/java/index.html

[15] Apache Tomcat.
http://tomcat.apache.org/

[16] Apache Derby.
http://db.apache.org/derby/

[17] Apache Lucene.
http://lucene.apache.org/java/docs/index.html

[18] JavaServer Pages Technology.
http://java.sun.com/products/jsp/

[19] W3C HTML.
http://www.w3.org/html/

[20] Apache Tika.
http://tika.apache.org/

[21] SiteMesh.
http://www.opensymphony.com/sitemesh/

[22] Struts 2. Apache Software Foundation.
http://struts.apache.org/2.x/index.html

[23] Hibernate - Jboss Comunity.
http://www.hibernate.org/

[24] The DBLP Computer Science Bibliography.
http://www.informatik.uni-trier.de/ ley/db/

[25] jReform.
http://jreform.sourceforge.net/

[26] Doxygen.
http://www.stack.nl/ dimitri/doxygen/index.html

[27] W3Counter. Global Stats - Market Share Trends.
http://www.w3counter.com/trends

[28] W3Schools. Browser Statistics.
http://www.w3schools.com/browsers/browsers_stats.asp

[29] Eclipse.
http://www.eclipse.org/

[30] Apache Subversion.
http://subversion.apache.org/

[31] Firebug.
http://getfirebug.com

[32] Luke - Lucene Index Toolbox.
http://www.getopt.org/luke/

[33] Apache log4j.
http://logging.apache.org/log4j/index.html

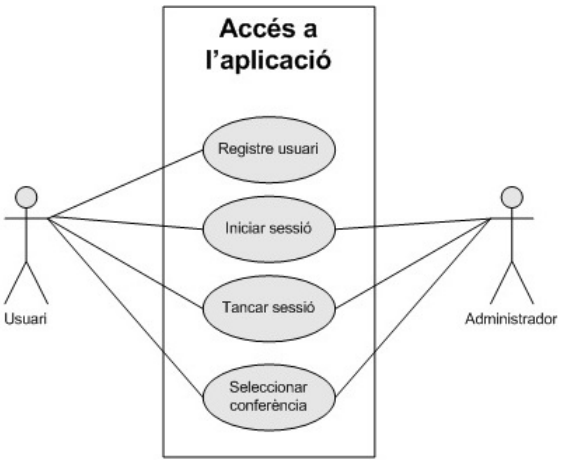
Annex A

Especificació

A.1 Model de casos d'ús

En aquest apartat es mostra l'especificació del model de casos d'ús del sistema. Per tal de simplificar la lectura, s'han agrupat alguns casos d'us similars com per exemple Afegir sessió, Afegir presentació i Afegir categoria de presentació.

A.1.1 Accés a l'aplicació



Cas d'ús: Registre usuari
Actors: Usuari
Descripció: Permet registrar un usuari anònim en l'aplicació.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari prem el botó de registrar	
	2. El sistema mostra un formulari per omplir els camps: correu electrònic, contrasenya, confirmació de contrasenya, nom i cognoms.
3. L'usuari omple els camps i prem el botó de guardar.	
	4. El sistema comprova les dades introduïdes per l'usuari.
	5. El sistema enregistra el nou usuari.

Cursos alternatius:

- Línia 4: Si hi ha dades invàlides, el sistema mostra un error i demana que introdueixi de nou les dades errònies. Torna al pas 3.

Cas d'ús: Iniciar sessió
Actors: Usuari, Administrador
Descripció: Permet que un usuari s'identifiqui en el sistema.
Precondició: L'usuari ha d'estar registrat en el sistema.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari es connecta a l'aplicació	
	2. El sistema mostra el formulari de *login*.
3. L'usuari introdueix el seu identificador i la contrasenya.	
	4. El sistema comprova les dades introduïdes per l'usuari.
	5. El sistema inicia la sessió de l'usuari

Cursos alternatius:

- Línia 4: Si hi ha dades invàlides, el sistema mostra un error i demana que introdueixi de nou les dades errònies. Torna al pas 3.

Cas d'ús: Tancar sessió
Actors: Usuari, Administrador
Descripció: Permet que un usuari tanqui sessió en el sistema.
Precondició: L'usuari està identificat en el sistema.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari prem el botó de sortir del sistema	
	2. Es tanca la sessió de l'usuari en el sistema.

Cas d'ús: Seleccionar conferència
Actors: Usuari, Administrador
Descripció: Permet que un usuari seleccioni una conferència.
Precondició: -
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari va a l'índex de conferències	
	2. El sistema retorna el conjunt de conferències disponibles segons si es tracta d'un administrador o no.
3. L'usuari selecciona la conferència que desitja.	
	4. El sistema entra en el context de la conferència triada

A.1.2 Administració de conferències

Cas d'ús: Afegir conferència
Actors: Administrador
Descripció: Permet afegir una conferència.
Precondició: L'administrador està identificat en el sistema.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'administrador prem el botó d'afegir conferència	
	2. El sistema mostra un formulari per introduir les dades de la conferència.
3. L'administrador omple els camps i envia el formulari.	
	4. El sistema comprova les dades introduïdes per l'administrador. 5. El sistema emmagatzema la nova conferència.

Cursos alternatius:

- Línia 4: Si hi ha dades invàlides, el sistema mostra un error i demana que introdueixi de nou les dades errònies. Torna al pas 3.

Cas d'ús: Editar conferència
Actors: Administrador
Descripció: Permet editar una conferència.
Precondició: L'administrador està identificat en el sistema.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'administrador selecciona la conferència a editar	
	2. El sistema mostra un formulari amb les dades de la conferència.
3. L'administrador modifica les dades que desitja i envia el formulari.	
	4. El sistema comprova les dades introduïdes per l'administrador. 5. El sistema actualitza la conferència.

Cursos alternatius:

- Línia 4: Si hi ha dades invàlides, el sistema mostra un error i demana que introdueixi de nou les dades errònies. Torna al pas 3.

Cas d'ús: Afegir sessió/presentació/categoria de presentació
Actors: Administrador
Descripció: Permet afegir una nova sessió, presentació o categoria de presentació.
Precondició: L'administrador està identificat en el sistema i ha seleccionat una conferència.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'administrador decideix afegir una nova sessió, presentació o categoria de presentació	
	2. El sistema mostra un formulari per introduir les dades de l'ítem.
3. L'administrador omple les dades i envia el formulari.	
	4. El sistema comprova les dades introduïdes per l'administrador. 5. El sistema emmagatzema la nova sessió, presentació o categoria de presentació.

Cursos alternatius:

- Línia 4: Si hi ha dades invàlides, el sistema mostra un error i demana que introdueixi de nou les dades errònies. Torna al pas 3.

Cas d'ús: Editar sessió/presentació/categoria de presentació
Actors: Administrador
Descripció: Permet editar una sessió, presentació o categoria de presentació.
Precondició: L'administrador està identificat en el sistema i ha seleccionat una conferència.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'administrador decideix editar una sessió, presentació o categoria de presentació	
	2. El sistema mostra un formulari amb les dades de l'ítem seleccionat.
3. L'administrador modifica les dades que desitja i envia el formulari.	
	4. El sistema comprova les dades introduïdes per l'administrador. 5. El sistema actualitza les dades de la sessió, presentació o categoria de presentació seleccionada.

Cursos alternatius:

- Línia 4: Si hi ha dades invàlides, el sistema mostra un error i demana que introdueixi de nou les dades errònies. Torna al pas 3.

Cas d'ús: Eliminar sessió/presentació/categoria de presentació
Actors: Administrador
Descripció: Permet eliminar una sessió, presentació o categoria de presentació.
Precondició: L'administrador està identificat en el sistema i ha seleccionat una conferència.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'administrador prem l'opció d'eliminar una sessió, presentació o categoria de presentació. 3. L'administrador confirma l'acció.	2. El sistema demana confirmació. 4. El sistema fa les comprovacions necessàries. 5. El sistema elimina la sessió, presentació o categoria de presentació.

Cursos alternatius:

- Línia 4: Si no es pot eliminar la sessió, presentació o categoria seleccionada, s'interromp el curs d'esdeveniments i es mostra un missatge a l'administrador.
- Línia 3: Si l'administrador cancel·la l'acció, s'interromp el curs d'esdeveniments.

Cas d'ús: Publicar conferència
Actors: Administrador
Descripció: Permet canviar la visibilitat d'una conferència a pública, per tal que pugui ser visible per a la resta d'usuaris.
Precondició: L'administrador està identificat en el sistema i ha seleccionat una conferència.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'administrador prem l'opció de publicar conferència.	2. El sistema fa les comprovacions necessàries. 3. El sistema indexa les presentacions de la conferència. 4. El sistema canvia la visibilitat de la conferència.

Cursos alternatius:

- Línia 2: Si es produeix algun error, s'interromp el curs d'esdeveniments i es mostra un missatge a l'usuari.

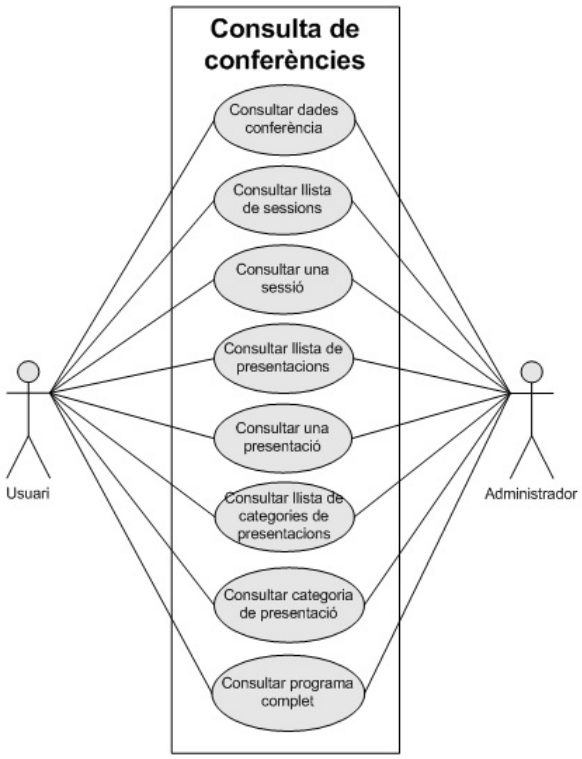
Cas d'ús: Passar conferència a privada
Actors: Administrador
Descripció: Permet canviar la visibilitat d'una conferència a privada.
Precondició: L'administrador està identificat en el sistema i ha seleccionat una conferència.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'administrador prem l'opció de passar conferència a privada.	2. El sistema fa les comprovacions necessàries. 3. El sistema canvia la visibilitat de la conferència.

Cursos alternatius:

- Línia 2: Si es produeix algun error, s'interromp el curs d'esdeveniments i es mostra un missatge a l'usuari.

A.1.3 Consulta de conferències



Cas d'ús: Consultar dades conferència
Actors: Usuari
Descripció: Permet consultar la informació bàsica conferència.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari decideix consultar la informació bàsica conferència	2. El sistema mostra les dades de la conferència.

Cas d'ús: Consultar llista de sessions/presentacions/categories de presentacions
Actors: Usuari
Descripció: Permet consultar la llista de sessions ,presentacions o categories de presentacions.
Precondició: L'usuari ha seleccionat una conferència (pública).
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari decideix consultar la llista de sessions ,presentacions o categories de presentacions.	2. El sistema mostra la llista demanada.

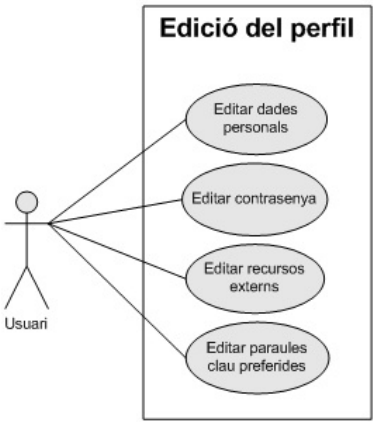
Cas d'ús: Consultar una sessió/presentació/categoria de presentació
Actors: Usuari
Descripció: Permet consultar la informació d'una determinada de sessió, presentació o categoria de presentació.
Precondició: L'usuari ha seleccionat una conferència (pública).
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari decideix consultar una determinada sessió, presentació o categoria de presentació.	2. El sistema mostra la informació detallada de l'ítem consultat.

Cas d'ús: Consultar programa complet
Actors: Usuari
Descripció: Permet consultar el programa complet d'una conferència amb les seves sessions i presentacions.
Precondició: L'usuari ha seleccionat una conferència (pública).
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari decideix consultar el programa d'una conferència.	2. El sistema mostra el programa de la conferència a l'usuari.

A.1.4 Edició del perfil

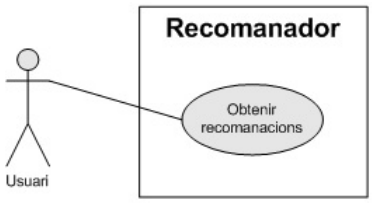


D'aquest conjunt només es descriu el cas d'us EDITAR RECURSOS EXTERNS, ja que la resta no tenen cap complexitat especial i són molt semblants a d'altres descrits anteriorment.

Cas d'ús: EDITAR RECURSOS EXTERNS
Actors: Usuari
Descripció: Permet a l'usuari identificar-se en recursos externs per tal que el sistema en pugui extreure informació i utilitzar-la en el recomanador.
Precondició: L'usuari està identificat en el sistema.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari entra al seu perfil	
	2. El sistema mostra la informació del seu perfil.
3. L'usuari prem l'opció d'editar els recursos externs	
	4. El sistema busca informació sobre l'usuari en recursos externs i mostra una llista de possibles resultats.
5. L'usuari selecciona els autors o usuaris dels recursos amb els que s'identifica i accepta.	
	6. El sistema guarda la llista d'autors o usuaris amb els que s'ha identificat. 7. El sistema obté les publicacions dels autors seleccionats, processa la informació i l'emmagatzema en el perfil de l'usuari. 8. El sistema obté els coautors de l'usuari en les publicacions trobades. 9. El sistema obté les publicacions dels coautors trobats, processa la informació i l'emmagatzema en el perfil de l'usuari.

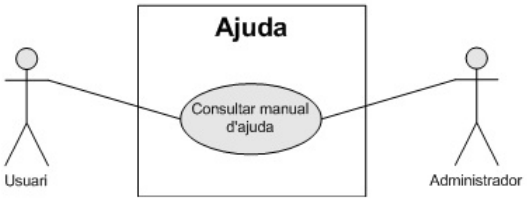
A.1.5 Recomanador



Cas d'ús: Obtenir recomanacions
Actors: Usuari
Descripció: Permet a l'usuari obtenir recomanacions personalitzades.
Precondició: L'usuari està identificat en el sistema i ha seleccionat una conferència.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari demana recomanacions	2. El sistema utilitza el motor recomanador amb la informació extreta del perfil de l'usuari. 3. El sistema mostra les recomanacions obtingudes ordenades segons el grau d'interès estimat.

A.1.6 Ajuda



Cas d'ús: Consultar manual d'ajuda
Actors: Usuari, Administrador
Descripció: Permet consultar manuals de configuració, manuals d'ús per a l'usuari i per a l'administrador.
Curs típic d'esdeveniments:

Accions dels actors	Resposta del sistema
1. L'usuari decideix consultar algun dels manuals del projecte.	2. El sistema retorna una llista dels diferents manuals disponibles
3. L'usuari selecciona el manual que li interessa	4. El sistema mostra el manual seleccionat.

Annex B

Implementació

B.1 Detalls d'implementació

B.1.1 Indexació amb Lucene

El procés d'indexació es pot dividir en diferents etapes: obtenir les dades, crear documents a partir del seu contingut (possiblement calgui extreure el text de documents binaris, ja que Lucene només treballa amb text) i, finalment, indexar els documents. Durant el procés d'indexació, Lucene processa els documents i analitza el seus continguts per extreure termes i emmagatzemar-los en l'estructura de dades anomenada índex invertit. En la figura B.1 es pot veure un esquema del procés d'indexació.

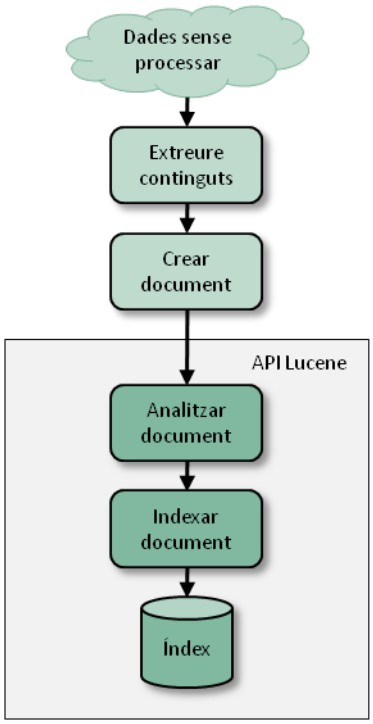


Figura B.1: Procés d'indexació amb Lucene

En els següents subapartats es farà un breu resum de les classes principals de la API de Lucene per a indexar i, a continuació, es comentarà el codi creat per indexar les presentacions d'una conferència.

API de Lucene per a indexar

En aquesta secció, es descriuen les classes principals de Lucene per a crear un índex.

- `IndexWriter`
- `Directory`
- `Analyzer`
- `Document`
- `Field`

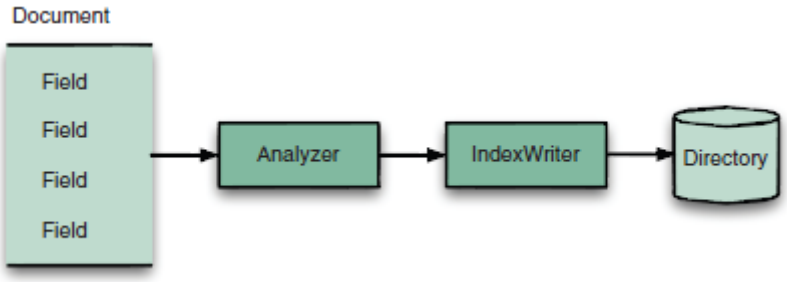


Figura B.2: Principals classes de Lucene per a indexar

`IndexWriter` és el component principal en el procés d'indexació. S'encarrega de crear un nou índex i afegir documents a un índex existent. Un `IndexWriter` utilitza un objecte `Directory` i un `Analyzer`.

La classe `Directory` representa la ubicació d'un índex de Lucene. És una classe abstracta que permet que les seves subclasses guardin l'índex segons les seves necessitats. Lucene inclou dues implementacions d'aquesta classe: `FSDirectory`, que emmagatzema l'índex en el disc, i `RAMDirectory`, que manté l'índex en memòria.

Abans d'indexar el text, es passa a traves d'un `Analyzer` que s'encarrega d'extreure els tokens del text. Un `Analyzer` aplica al text primer un `Tokenizer` i després un o més `TokenFilter`s. De forma simplificada es pot dir que un `Tokenizer` llegeix el fitxer de text i el fragmenta en `Token`s, generant una `TokenStream`. Un `TokenFilter` llegeix una `TokenStream` i en produeix una altra després d'eliminar tokens indesitjats o fer les transformacions necessàries. Això permet encadenar un `Tokenizer` amb un nombre indefinit de `TokenFilter`s i permet construir analitzadors complexos a partir de blocs simples. A l'hora d'integrar Lucene en una aplicació, l'elecció de l'analitzador és un element crític en el disseny de l'aplicació.

Lucene inclou varis `Analyzer`s predefinits, entre els quals es troben:

- `WhitespaceAnalyzer`, que divideix els tokens en els espais en blanc.

- `SimpleAnalyzer`, que divideix el text en els caràcters no alfabètics i espais en blanc. Posteriorment converteix els tokens en minúscula.
- `StopAnalyzer`, que divideix el text en caràcters no alfabètics, espais en blanc, converteix els caràcters en minúscula i elimina paraules buides.
- `StandardAnalyzer`, que tokenitza basant-se en una sofisticada gramàtica que reconeix alfanumèrics, adreces e-mail, acrònims, noms de hosts d'Internet, els caràcters xinesos, japonesos i coreans, posteriorment converteix els tokens en minúscula i elimina paraules buides.

Els `Document`s són les entitats atòmiques que s'agreguen a un índex i que es poden recuperar a través de cerques sobre aquest índex. Un `Document` està compost per un conjunt camps amb un nom i un valor textual, anomenats `Field`s.

Un `Field` correspon a informació sobre la que es vol fer cerques o mostrar en els resultats. Per a cada `Field` es pot dir si s'ha d'indexar o no (segons si es volen fer cerques en aquest camp o no) i també si s'ha analitzar el seu valor abans d'indexar-lo o no. Generalment no s'analitzen camps que representen dates, URLs, etc.

A l'hora de crear un Field, Lucene disposa de les següents constants per definir-lo:

- `Field.Index.ANALYZED` Indexa els tokens produïts després de processar el valor del camp amb un `Analyzer`.
- `Field.Index.NO` No indexa el valor d'aquest camp.
- `Field.Index.NOT_ANALYZED` Indexa el valor del camp sense ser processat per un `Analyzer`
- `Field.Stored.YES` Emmagatzema el valor original del camp en l'índex i permet recuperar-lo en els resultats de la cerca. Això és útil per a textos curts com el títol dels documents que es volen mostrar en els resultats.
- `Field.Stored.NO` No emmagatzema el valor del camp en l'índex.

Detalls del codi desenvolupat

En aquesta secció es s'explicaran els detalls del codi que s'encarrega d'indexar les presentacions d'una conferència.

Llistat de codi B.1: Indexació de les presentacions d'una conferència

```
//Get conference index file.
String indexFilePath = AppProperties.getHomeSubdirectory(INDEXES_SUBDIR, c.getCode());
File indexFile = new File(indexFilePath);
//Create IndexWriter
IndexWriter writer = new IndexWriter(
       FSDirectory.open(indexFile), //Index directory
       getAnalyzer(), //Analyzer to use
       true, //'true' to create the index or overwrite the existing one
       IndexWriter.MaxFieldLength.LIMITED); //Maximum field length

//Add presentations to index
for(Session s: c.getSessions()){
```

```
    for (Presentation p: s.getPresentations()){
        Document doc = getDocument(p);
        writer.addDocument(doc);
    }
}

//Optimize and close index
writer.optimize();
writer.close();
return true;
```

En primer lloc s'ha creat una instància de l'`IndexWriter` passant com a paràmetre un `Directory` i un `Analyzer` entre altres. Pel que fa al directori, s'ha utilitzat un `FSDirectory` per tal de guardar els índexs físicament en directoris del sistema d'arxius.

Pel que fa a l'analitzador, s'ha extret la configuració en un fitxer a part per tal de millorar la modularitat i canviabilitat. Això permet fer proves amb diferents analitzadors sense haver de modificar totes les classes on s'utilitza, ja que s'ha de fer servir el mateix analitzador en la indexació i en la cerca.

Llistat de codi B.2: Analitzador

```
public static Analyzer getAnalyzer() {
    return new StandardAnalyzer(Version.LUCENE_30);
}
```

S'ha utilitzat un dels `Analyzer`s que ve per defecte a Lucene, el `StandardAnalyzer`, que s'ajusta al que s'ha decidit en la conceptualització del sistema.

Després de crear l'`IndexWriter` amb les seus corresponents paràmetres, el següent pas es afegir les presentacions en l'índex. Seguint amb el llistat de codi B.1, es pot veure com es recorren les presentacions de la conferència, creant un document i afegint-lo a l'índex en cada cas. El mètode `getDocument` s'encarrega de crear un `Document` i omplir els seus camps a partir d'un objecte `Presentation`.

Llistat de codi B.3: Creació d'un `Document`

```
private static Document getDocument(Presentation p) {
    //Create Document and add Fields
    Document doc = new Document();
    doc.add(new Field("id", String.valueOf(p.getId()), Field.Store.YES, Field.Index.NO));
    doc.add(new Field("title", p.getTitle(),Field.Store.YES,Field.Index.ANALYZED));
    if(p.getAbstract_()!=null)
        doc.add(new Field("abstract", p.getAbstract_(),Field.Store.YES,Field.Index.ANALYZED));
    if(p.getPaperFile()!=null){
        String pathName = p.getPaperFilePath();
        Reader reader = ContentAnalysis.extractText(new File(pathName));
        if(reader!=null)
            doc.add(new Field("fullArticle", reader));
    }
    return doc;
}
```

En el fragment de codi anterior es pot veure com s'afegeixen fins a quatre `Fields` al `Document`. A continuació es defineixen les característiques de cadascun d'ells.

Field	**Descripció**	**Analitzat**	**Indexat**	**Emmagatzemat**
id	Identificador de la presentació en la base de dades del sistema.			✓
title	Títol de la presentació	✓	✓	✓
abstract	Resum de la presentació	✓	✓	✓
fullArticle	Text extret del seu *paper*	✓	✓	

L'identificador s'emmagatzema per tal de poder associar el document amb la presentació guardada en la base de dades. No cal ni analitzar ni indexar aquest camp, només ens interessa poder recuperar el seu valor original. El títol, el resum i l'article complet són camps que s'analitzen i s'indexen per tal de poder efectuar cerques sobre ells. Addicionalment, el títol i el resum també s'emmagatzemen ja que els volem mostrar a l'usuari en la llista de resultats. Generalment s'emmagatzemen els camps que es volen mostrar en la llista de resultats per tal d'evitar una crida a la base de dades per a cada resultat.

Cal destacar que per a obtenir el text de l'article complet s'ha hagut d'extreure de l'arxiu de la presentació tenint en compte el seu format. Si el contingut és textual amb una codificació estàndard coneguda, aquest treball és simple. No obstant, avui en dia molts document solen ser de tipus binari (PDF, Microsoft Office, Open Office, etc.) o contenen etiquetes o *markups* que cal eliminar abans d'indexar (XML, HTML, etc.).

Tal i com es mostra al llistat de codi B.1, un cop s'han afegit totes les presentacions a l'index, s'optimitza i es tanca l'índex. L'optimització d'un índex és un procés que agrupa varis arxius que formen l'índex per tal de minimitzar el temps de resposta en les cerques. Es recomana optimitzar l'índex quan es sap que l'índex es mantindrà sense canvis durant un temps. En aquest cas, s'ha optimitzat l'índex un cop s'han acabat les operacions d'indexació, just abans de tancar-lo.

B.1.2 Cerca amb Lucene

En la figura B.3 es pot veure el procés de cerca amb Lucene. L'usuari interactua amb la interfície d'usuari, a través de la quan envia les peticions de cerca. Com que en aquest cas no estem implementant una cerca de l'usuari, sinó un recomanador, el text de la consulta es construeix programàticament després en rebre la petició de l'usuari per obtenir recomanacions. A continuació es processa la consulta textual i es tradueix a un format comprensible per Lucene. Finalment, s'executa la consulta i es mostren els resultats a l'usuari.

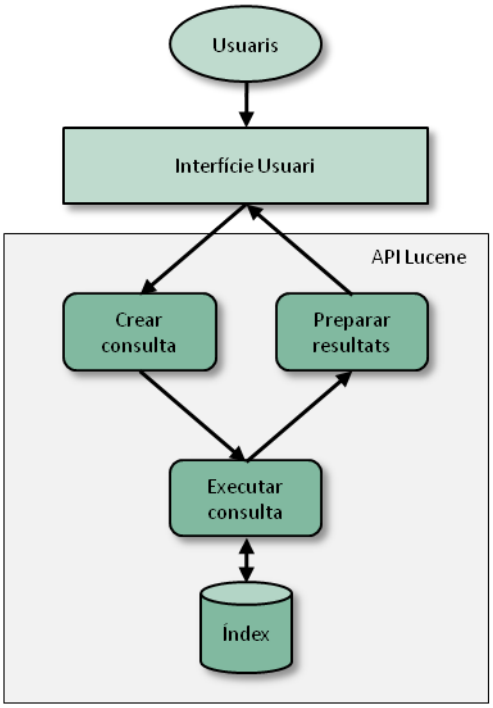


Figura B.3: Procés de cerca amb Lucene

API de Lucene per a cercar

En aquesta secció, es descriuen les classes principals de Lucene per a cercar en un índex.

- `IndexSearcher`
- `Query`
- `QueryParser`
- `TopDocs`

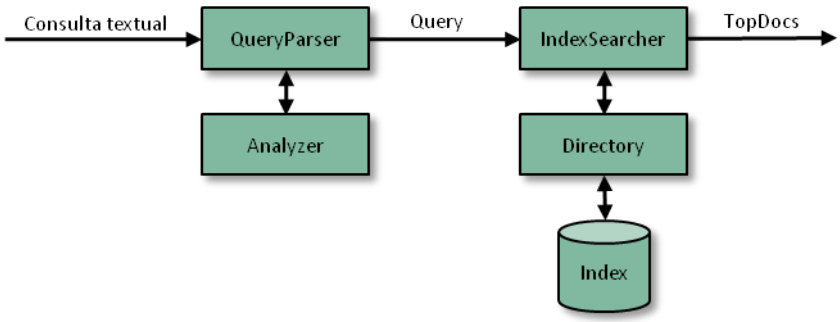


Figura B.4: Relació entre les classes principals de cerca

`IndexSearcher` és la classe principal per a cercar documents d'un índex. La ubicació de l'índex s'especifica mitjançant un `Directory`, de la mateixa manera que en l'`IndexWriter`. La cerca es fa efectiva passant una `Query` a qualsevol dels mètodes `search` de l'objecte `IndexSearcher`. La cerca

més simple parteix d'una `Query` i el número de resultats màxim que volem i retorna un objecte `TopDocs`, que es descriu més endavant.

Un `QueryParser` serveix per generar instàncies de `Query` a partir de consultes textuals, en el format definit per Lucene. `QueryParser` utilitza un `Analyzer` per a processar el text de la consulta de la mateixa forma que es processen els documents.

La classe `TopDocs` és un simple contenidor de punters als primers N millors resultats de la cerca. Per a cadascun dels resultats, `TopDocs` emmagatzema el `docID` (que permetrà recuperar el document) i la seva puntuació.

Detalls del codi desenvolupat

En primer lloc s'han utilitzat les classes de `Directory` i `IndexSearcher` per obrir l'índex desitjat en mode de cerca. El `Directory` és el mateix en el que es va dur a terme la indexació.

Llistat de codi B.4: IndexSearcher

```
//Get conference index file
String indexFilePath = AppProperties.getHomeSubdirectory(INDEXES_SUBDIR,c.getCode());
File indexFile = new File(indexFilePath);

//Create instance of IndexSearcher. Open index
final Searcher searcher = new IndexSearcher(FSDirectory.open(indexFile));
```

A continuació es crea una instància de `MultiFieldQueryParser` amb el mateix analitzador que es va usar en la indexació. `MultiFieldQueryParser`, com qualsevol altre subclasse de `QueryParser`, permet convertir el text de cerca en un objecte `Query` de Lucene.

Llistat de codi B.5: QueryParser

```
//Create array with field names
String[] fields = {"title", "abstract","fullArticle"};
//Create parser
MultiFieldQueryParser parser = new MultiFieldQueryParser(
               Version.LUCENE_30, fields, getAnalyzer());

//Parse query
Query query = parser.parse(queryString);
```

Concretament, `MultiFieldQueryParser` permet crear una `Query` que cerca en diversos camps. En aquest cas ens interessa cercar en els camps "title", "abstract" i"fullArticle".

Per a cercar en l'índex, s'invoca el mètode `search` de `IndexSearcher` i, finalment, tanquem el `IndexSearcher`.

Annex C

Proves

C.1 Dades utilitzades i entorn

Tot i que s'han fet proves amb diferents dades, en aquest apartat s'analitzen de forma detallada algunes de les proves realitzades amb les dades de la conferència real ECML PKDD 2009.

Les proves s'han realitzat l'ordinador personal que s'ha utilitzat per desenvolupar. A continuació es detallen les seves especificacions.

Entorn hardware

- CPU: Intel(R) Core(TM)2 Duo CPU P7450, 2133GHz, 2 processadors
- RAM: 4GB ($\sim$ 2GB disponibles), 512MB assignats a la màquina virtual de Java.
- Disc dur: 320GB, 5400rpm, IDE

Entorn software

- Versió Java: 1.6
- Versió Lucene: 3.0.1
- Sistema Operatiu: Microsoft Windows 7 Home Premium
- Ubicació de l'índex: local

C.2 Anàlisi en l'indexació de presentacions

C.2.1 Proves amb diferents analitzadors

En aquest apartat es mostrarà l'efecte de varis analitzadors sobre diferents textos. Per a fer les proves de forma automatitzada s'ha creat un programa que rep el text a analitzar com a paràmetre d'entrada i imprimeix el resultats després d'aplicar els diferents analitzadors. En aquest cas es proven quatre dels analitzadors predefinits en Lucene:

- `WhitespaceAnalyzer`, que divideix els *tokens* en els espais en blanc.
- `SimpleAnalyzer`, que divideix el text en els caràcters no alfabètics i espais en blanc. Posteriorment converteix els *tokens* en minúscula.
- `StopAnalyzer`, que divideix el text en caràcters no alfabètics, espais en blanc, converteix els caràcters en minúscula i elimina paraules buides.
- `StandardAnalyzer`, que tokenitza basant-se en una sofisticada gramàtica que reconeix alfanumèrics, adreces e-mail, acrònims, noms de hosts d'Internet, els caràcters xinesos, japonesos i coreans, posteriorment converteix els *tokens* en minúscula i elimina paraules buides.

A l'hora de mostrar els resultats, cada *token* es s'escriu entre claudàtors per tal de fer visibles les divisions.

S'han realitzat proves sobre varis textos extrets de presentacions reals de la conferència ECML PKDD 2009 triats de forma aleatòria. Degut a la seva extensió, en la memòria només inclourem l'anàlisi d'un dels títols i un dels resums de les presentacions. S'han triat textos que contenen diferents casos en els que cada analitzador actua de forma diferent per tal de visualitzar el comportament de cadascun d'ells.

En primer lloc veurem el resultat d'analitzar un dels títols de les presentacions.

Llistat de codi C.1: Test `Analyzers 1`

```
Analyzing "Communication-Efficient Classification in P2P Networks"
 WhitespaceAnalyzer:
    [Communication-Efficient] [Classification] [in] [P2P] [Networks]

 SimpleAnalyzer:
    [communication] [efficient] [classification] [in] [p] [p] [networks]

 StopAnalyzer:
    [communication] [efficient] [classification] [p] [p] [networks]

 StandardAnalyzer:
    [communication] [efficient] [classification] [p2p] [networks]
```

Podem observar que el resultat amb el `WhitespaceAnalyzer` és el mateix text d'entrada separat per espais. Amb el `SimpleAnalyzer`, no només els espais actuen com a separadors, sinó que també qualsevol altre caràcter no alfabètic, com en aquest cas '-' i '2'. Una altra diferència respecte l'anterior és que normalitza els *tokens* passant els caràcters a minúscula. També veiem que el resultat amb el `StopAnalyzer`, és el mateix que l'anterior però eliminant les paraules buides com 'in'. Finalment, el resultat amb el `StandardAnalyzer` veiem que també passa els caràcters a minúscula i elimina la paraula buida 'in' però, a diferència dels altres, no ha separat el mot 'p2p'.

A continuació es mostra el resultat d'analitzar el resum de la presentació amb títol 'The Growing Semantic Web' per tal de veure més casos de comportament diferent dels analitzadors en un text més llarg.

Llistat de codi C.2: Test `Analyzers 2`

```
Analyzing "From its beginnings in 2004, the data available on the web in Semantic Web formats has
    typically been both eclectic and relatively small, and closely linked the interests of
    particular researchers. In the past year, however, the quantity and scope of data published on
    the public semantic web has exploded, and the size of the semantic web is now measured in the
    billions of assertions. It is a significant and growing resource for applications which depend
    on web-based resources for some or all of their knowledge. With this massive increase in
    quantity and scope come many opportunities, as well as the usual issues of scale on the web:
    inconsistency, mapping problems, incompleteness and data variability. This talk will cover the
    history and current state of the Semantic Web and the Linked Data Cloud, describe some of the
    uses to which web-based semantic data is currently put, and discuss prospects for the ECML/PKDD
    community to leverage "

WhitespaceAnalyzer:
    [From] [its] [beginnings] [in] [2004,] [the] [data] [available] [on] [the] [web] [in] [Semantic]
    [Web] [formats] [has] [typically] [been] [both] [eclectic] [and] [relatively] [small,] [and]
    [closely] [linked] [the] [interests] [of] [particular] [researchers.] [In] [the] [past] [year,]
    [however,] [the] [quantity] [and] [scope] [of] [data] [published] [on] [the] [public] [semantic]
    [web] [has] [exploded,] [and] [the] [size] [of] [the] [semantic] [web] [is] [now] [measured] [in]
    [the] [billions] [of] [assertions.] [It] [is] [a] [significant] [and] [growing] [resource] [for]
    [applications] [which] [depend] [on] [web-based] [resources] [for] [some] [or] [all] [of] [their]
    [knowledge.] [With] [this] [massive] [increase] [in] [quantity] [and] [scope] [come] [many]
    [opportunities,] [as] [well] [as] [the] [usual] [issues] [of] [scale] [on] [the] [web:]
    [inconsistency,] [mapping] [problems,] [incompleteness] [and] [data] [variability.] [This] [talk]
    [will] [cover] [the] [history] [and] [current] [state] [of] [the] [Semantic] [Web] [and] [the]
    [Linked] [Data] [Cloud,] [describe] [some] [of] [the] [uses] [to] [which] [web-based] [semantic]
    [data] [is] [currently] [put,] [and] [discuss] [prospects] [for] [the] [ECML/PKDD] [community]
    [to] [leverage]

SimpleAnalyzer:
    [from] [its] [beginnings] [in] [the] [data] [available] [on] [the] [web] [in] [semantic] [web]
    [formats] [has] [typically] [been] [both] [eclectic] [and] [relatively] [small] [and] [closely]
    [linked] [the] [interests] [of] [particular] [researchers] [in] [the] [past] [year] [however]
    [the] [quantity] [and] [scope] [of] [data] [published] [on] [the] [public] [semantic] [web] [has]
    [exploded] [and] [the] [size] [of] [the] [semantic] [web] [is] [now] [measured] [in] [the]
    [billions] [of] [assertions] [it] [is] [a] [significant] [and] [growing] [resource] [for]
    [applications] [which] [depend] [on] [web] [based] [resources] [for] [some] [or] [all] [of]
    [their] [knowledge] [with] [this] [massive] [increase] [in] [quantity] [and] [scope] [come]
    [many] [opportunities] [as] [well] [as] [the] [usual] [issues] [of] [scale] [on] [the] [web]
    [inconsistency] [mapping] [problems] [incompleteness] [and] [data] [variability] [this] [talk]
    [will] [cover] [the] [history] [and] [current] [state] [of] [the] [semantic] [web] [and] [the]
    [linked] [data] [cloud] [describe] [some] [of] [the] [uses] [to] [which] [web] [based] [semantic]
    [data] [is] [currently] [put] [and] [discuss] [prospects] [for] [the] [ecml] [pkdd] [community]
    [to] [leverage]

StopAnalyzer:
    [from] [its] [beginnings] [data] [available] [web] [semantic] [web] [formats] [has] [typically]
    [been] [both] [eclectic] [relatively] [small] [closely] [linked] [interests] [particular]
    [researchers] [past] [year] [however] [quantity] [scope] [data] [published] [public] [semantic]
    [web] [has] [exploded] [size] [semantic] [web] [now] [measured] [billions] [assertions]
    [significant] [growing] [resource] [applications] [which] [depend] [web] [based] [resources]
    [some] [all] [knowledge] [massive] [increase] [quantity] [scope] [come] [many] [opportunities]
    [well] [usual] [issues] [scale] [web] [inconsistency] [mapping] [problems] [incompleteness]
    [data] [variability] [talk] [cover] [history] [current] [state] [semantic] [web] [linked] [data]
    [cloud] [describe] [some] [uses] [which] [web] [based] [semantic] [data] [currently] [put]
    [discuss] [prospects] [ecml] [pkdd] [community] [leverage]
```

```
StandardAnalyzer:
   [from] [its] [beginnings] [2004] [data] [available] [web] [semantic] [web] [formats] [has]
   [typically] [been] [both] [eclectic] [relatively] [small] [closely] [linked] [interests]
   [particular] [researchers] [past] [year] [however] [quantity] [scope] [data] [published] [public]
   [semantic] [web] [has] [exploded] [size] [semantic] [web] [now] [measured] [billions] [assertions]
   [significant] [growing] [resource] [applications] [which] [depend] [web] [based] [resources]
   [some] [all] [knowledge] [massive] [increase] [quantity] [scope] [come] [many] [opportunities]
   [well] [usual] [issues] [scale] [web] [inconsistency] [mapping] [problems] [incompleteness] [data]
   [variability] [talk] [cover] [history] [current] [state] [semantic] [web] [linked] [data] [cloud]
   [describe] [some] [uses] [which] [web] [based] [semantic] [data] [currently] [put] [discuss]
   [prospects] [ecml] [pkdd] [community] [leverage]
```

A continuació es comenten algunes de les diferències que es poden apreciar.

- Com que el `WhitespaceAnalyzer` únicament separa per espais, podem trobar *tokens* com [researchers.] i [inconsistency,]. Això no es reprodueix en la resta ja que els signes de puntuació també actuen com a separadors i es descarten.
- `WhitespaceAnalyzer` és l'únic dels quatre que no normalitza els caràcters passant-los a minúscula.
- El token numèric '2004' s'ha descartat amb el `SimpleAnalyzer` i el `StopAnalyzer`.
- Amb el `StopAnalyzer` i `StandardAnalyzer` s'han descartat paraules buides com [in], [the], [on], [and], [of], [as], [will] i [to].

Finalment veiem com es comportarien cadascun dels analitzadors en altres casos que es podrien donar:

Llistat de codi C.3: Test `Analyzers` 3

```
Analyzing "Other special cases: EE.UU., 2x-1, name@example.com, 142.32.48.231, www.google.com"
 WhitespaceAnalyzer:
    [Other] [special] [cases:] [EE.UU.,] [2x-1,] [name@example.com,] [142.32.48.231,] [www.google.com]

 SimpleAnalyzer:
    [other] [special] [cases] [ee] [uu] [x] [name] [example] [com] [www] [google] [com]

 StopAnalyzer:
    [other] [special] [cases] [ee] [uu] [x] [name] [example] [com] [www] [google] [com]

 StandardAnalyzer:
    [other] [special] [cases] [ee.uu] [2x-1] [name@example.com] [142.32.48.231] [www.google.com]
```

En conclusió veiem que el `StandardAnalyzer` és l'analitzador més sofisticat que separa les paraules en els espais i signes de puntuació sense descartar els *tokens* numèrics, i té una mica de lògica per a identificar certs tipus de *tokens* que no s'haurien de separar com ara sigles o acrònims separats per punts, correus electrònics, direccions IP, URLs, etc. També passa els caràcters a minúscula i descarta una llista de paraules buides.

Lucene no fa que aquests resultats del procés d'anàlisi siguin visibles a l'usuari final. Els termes resultants de l'anàlisi són afegits directament a l'índex. Són aquests termes, i només aquests termes, els que es mirarà si coincideixen amb els de la cerca.

C.2.2 Visualització dels continguts de l'índex

Després de dur a terme la indexació de les presentacions d'una conferència amb l'analitzador `StandardAnalyzer`, s'ha utilitzat l'eina Luke, descrita en el capítol 8, per tal d'inspeccionar els continguts de l'índex. En les figures C.1, C.2 i C.3 es mostren la llistes de termes més freqüents en l'índex per a cada camp.

Es pot veure que els termes del camp de l'índex són prou acceptables, en el sentit que són termes que aporten certa informació sobre els continguts, tot i així apareix alguna paraula buida com 'which'. No obstant els títols solen contenir poques paraules buides originalment, així que ens interessa veure el resultat els camps dels resums i, especialment, els textos extrets dels fitxers associats. Sobre els termes indexats dels resums, veiem que comencen a aparèixer més paraules buides com 'we', 'our', 'can', 'has'. I, per últim, també es pot observar que en vint termes més freqüents del camp *fullArticle* pràcticament no hi han paraules que aportin informació, tret d'algunes excepcions.

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	33	title	learning
2	13	title	based
3	13	title	data
4	11	title	multi
5	11	title	classification
6	10	title	feature
7	10	title	from
8	10	title	using
9	9	title	mining
10	9	title	models
11	8	title	graph
12	8	title	efficient
13	7	title	semi
14	7	title	networks
15	7	title	algorithms
16	6	title	text
17	6	title	selection
18	6	title	analysis
19	6	title	language
20	5	title	kernel

Figura C.1: Termes indexats més freqüents del camp `title`

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	105	abstract	we
2	65	abstract	data
3	64	abstract	learning
4	63	abstract	our
5	60	abstract	from
6	60	abstract	which
7	60	abstract	paper
8	54	abstract	can
9	52	abstract	methods
10	49	abstract	based
11	46	abstract	problem
12	45	abstract	show
13	44	abstract	approach
14	44	abstract	algorithm
15	44	abstract	propose
16	41	abstract	algorithms
17	39	abstract	results
18	38	abstract	new
19	38	abstract	using
20	36	abstract	has

Figura C.2: Termes indexats més freqüents del camp `abstract`

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	136	fullArticle	2009
2	135	fullArticle	et
3	135	fullArticle	i
4	135	fullArticle	p
5	135	fullArticle	w
6	135	fullArticle	al
7	134	fullArticle	ecml
8	134	fullArticle	pkdd
9	133	fullArticle	s
10	133	fullArticle	abstract
11	132	fullArticle	part
12	131	fullArticle	springer
13	131	fullArticle	1
14	131	fullArticle	we
15	130	fullArticle	e
16	129	fullArticle	l
17	129	fullArticle	c
18	129	fullArticle	t
19	128	fullArticle	m
20	128	fullArticle	can

Figura C.3: Termes indexats més freqüents del camp `fullArticle`

D'aquesta manera s'ha vist que, tot i haver eliminat una llista de paraules buides més freqüents, seguien apareixent molts termes que no aportaven informació sobre els continguts. L'analitzador `StandardAnalyzer`, si no s'especifica el contrari, utilitza per defecte la següent llista de paraules buides:

```
a, an, and, are, as, at, be, but, by, for, if, in, into,
is, it, no, not, of, on, or, such, that, the, their, then,
there, these, they, this, to, was, will, with
```

Vistos els resultats, s'ha decidit utilitzar una llista de paraules buides més completa. Algunes de

les paraules que es poden trobar en aquesta llista són les següents:

```
a, a's, able, about, above, according, accordingly, across,
actually, after, afterwards, again, against, ain't, all,
allow, allows, almost, alone, along, already, also,
although, always, am, among, amongst, an, and, another,
any, anybody, anyhow, anyone, anything, anyway, anyways,
anywhere, apart, appear, appreciate, appropriate, are,
aren't, around, as, aside, ask, asking, associated, at,
available, away, awfully, b, be, became, because, become,
becomes, becoming, been, before, beforehand, behind, being,
believe, below, beside, besides, best, better, between,
beyond, both, brief, but, by, c, c's, came, can ...
```

A continuació es tornen a mostrar les llistes de termes més freqüents per a cada camp, havent tornat a indexar les presentacions utilitzant l'analitzador `StandardAnalyzer` amb la nova llista de paraules buides.

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	33	title	learning
2	13	title	data
3	13	title	based
4	11	title	multi
5	11	title	classification
6	10	title	feature
7	9	title	models
8	9	title	mining
9	8	title	efficient
10	8	title	graph
11	7	title	networks
12	7	title	semi
13	7	title	algorithms
14	6	title	language
15	6	title	analysis
16	6	title	text
17	6	title	selection
18	5	title	discovery
19	5	title	clustering
20	5	title	kernel

Figura C.4: Termes indexats més freqüents del camp `title`

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	65	abstract	data
2	64	abstract	learning
3	60	abstract	paper
4	52	abstract	methods
5	49	abstract	based
6	46	abstract	problem
7	45	abstract	show
8	44	abstract	propose
9	44	abstract	algorithm
10	44	abstract	approach
11	41	abstract	algorithms
12	39	abstract	results
13	35	abstract	proposed
14	33	abstract	present
15	32	abstract	classification
16	31	abstract	problems
17	30	abstract	method
18	30	abstract	model
19	29	abstract	information
20	28	abstract	framework

Figura C.5: Termes indexats més freqüents del camp `abstract`

Top ranking terms. (Right-click for more options)

No	Rank	Field	Text
1	136	fullArticle	2009
2	135	fullArticle	al
3	134	fullArticle	pkdd
4	134	fullArticle	ecml
5	133	fullArticle	abstract
6	132	fullArticle	part
7	132	fullArticle	springer
8	131	fullArticle	1
9	128	fullArticle	verlag
10	127	fullArticle	eds
11	126	fullArticle	data
12	126	fullArticle	buntine
13	126	fullArticle	berlin
14	126	fullArticle	lnai
15	124	fullArticle	based
16	123	fullArticle	heidelberg
17	122	fullArticle	2
18	122	fullArticle	3
19	120	fullArticle	circlecopyrt
20	120	fullArticle	number

Figura C.6: Termes indexats més freqüents del camp `fullArticle`

C.3 Duració del procés d'indexació

En aquest apartat es mostrarà el temps que ha tardat l'aplicació en indexar la conferència que s'esta fent servir d'exemple en aquestes proves.

A continuació s'especifiquen les variables d'indexació de Lucene en aquest cas concret:

- Nombre de documents: 136

- Mida total dels documents originals: Uns 7.635.000 caràcters en total. Continguts de la base de dades + fitxers (total $\sim$ 120 MB)
- Mida mitjana dels documents originals: Uns 56.140 caràcters. Continguts de la base de dades + fitxer ($\sim$ 903KB).
- Ubicació dels documents originals: base de dades i sistema de fitxers (*filesystem*)
- Tipus de dades: Text i fitxers en PDF
- *Parser*: Apache Tika
- Analitzador: *StandardAnalyzer*
- Nombre de camps per document: 4
- Tipus de camps: id (*stored*), títol (*stored*, *analyzed*, *indexed*), resum (*stored*, *analyzed*, *indexed*), text extret del fitxer (*analyzed*, *indexed*)
- Persistència de l'índex: FSDirectory (*filesystem*)
- Mida de l'índex: 1,90 MB

Per a 136 documents, l'aplicació ha invertit una mitjana de 95224 milisegons ($\simeq$ 1,6 minuts) en 6 execucions independents. Els temps obtinguts són raonables, tenint en compte que s'han realitzat les proves en un ordinador personal i no en un entorn de producció amb màquines més potents.

C.4 Obtenció de dades d'un usuari en recursos externs

Per a dur a terme les proves, s'ha utilitzat l'usuari "Ricard Gavaldà". S'han fet les proves amb un usuari real (no inventat) per tal de poder recuperar informació dels recursos externs.

Posem per cas que l'usuari s'identifica amb l'autor "Ricard Gavaldà" de DBLP que té per codi "g/Gavald=agrave=:Ricard". En aquest cas l'aplicació recupera dades del recurs:

http://dblp.uni-trier.de/rec/pers/g/Gavald=agrave=:Ricard/xk.

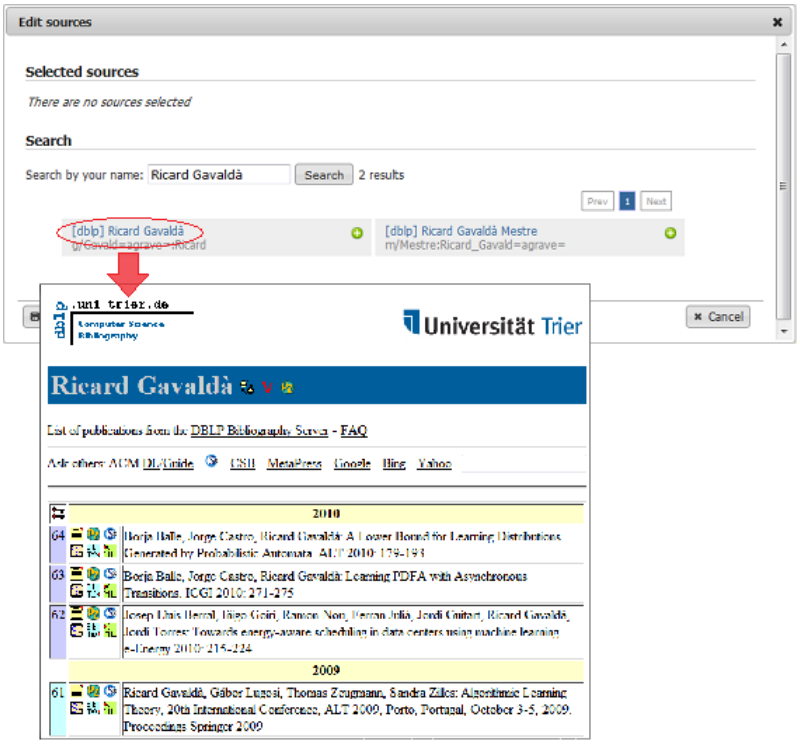


Figura C.7: DBLP

D'aquest recurs s'obtenen els títols de les seves publicacions. A continuació es mostren les primeres publicacions recuperades per l'autor 'Ricard Gavaldà'.

- A Lower Bound for Learning Distributions Generated by Probabilistic Automata.
- Learning PDFA with Asynchronous Transitions.
- Towards energy-aware scheduling in data centers using machine learning.
- Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings
- Improving Adaptive Bagging Methods for Evolving Data Streams.
- An Algebraic Perspective on Boolean Function Learning.
- Adaptive XML Tree Classification on Evolving Data Streams.
- Predicting Web Server Crashes: A Case Study in Comparing Prediction Algorithms.
- Adaptive Learning from Evolving Data Streams.
- New ensemble methods for evolving data streams.

En el següent pas, s'obtenen els coautors prioritzant els més pròxims de l'autor en qüestió. Es consideren que són més pròxims els que tenen més publicacions en comú. En la següent llista es mostren alguns coautors amb més prioritat en el cas del Ricard Gavaldà i el número de publicacions que comparteixen:

- Osamu Watanabe (14)

- Jordi Torres (7)
- Denis Thérien (6)
- Nicolás Poggi (5)
- Toni Moreno (3)
- Hava T. Siegelmann (3)

I, de cadascun dels coautors obtinguts, s'obtenen les seves publicacions.

Un dels problemes que s'ha trobat és el temps que tarda en recopilar aquesta informació. Tenint en compte que s'han utilitzat crides a la API de DBLP que retornen respostes en format XML, aquest procés es tradueix a un gran nombre de peticions.

En primer lloc, es fa una petició per recuperar els codis de les seves publicacions, i després cal fer una petició per a cada publicació per obtenir-ne els seus títols. Amb una altra petició es poden obtenir els seus coautors, i amb els coautors es repeteixen els passos anteriors per obtenir els seus títols. D'aquesta manera, si es tractés d'un usuari amb moltes publicacions i coautors, aquest procés es traduiria a un gran nombre de peticions a externes seguides del seu processat per extraure'n les dades.

Per tal de reduir el temps d'espera s'han dut a terme els introduït els següents límits:

- Si l'usuari té més de 20 publicacions, només es recuperen les últimes 20.
- Si l'usuari té més de 20 coautors, només es recuperaran els 20 que comparteixin més publicacions.
- De cada coautor es recuperaran un màxim de 5 publicacions (que no es repeteixin amb les del propi autor, ni amb altres coautors recuperats anteriorment).

Amb aquestes restriccions s'han conseguit uns temps acceptables. A continuació es mostren un parell d'exemples.

Prova 1: Jordi Delgado (identificat a DBLP)

- 7 publicacions
- 6 coautors
- 26 publicacions de coautors
- Mitjana de temps del procés: 5 segons

En aquesta primera prova només ha actuat la restricció de màxim de publicacions per coautor.

Prova 2: Ricard Gavaldà (identificat a DBLP)

- 20 publicacions
- 20 coautors
- 80 publicacions de coautors
- Mitjana de temps del procés: 16.5 segons

En aquesta segona prova han actuat totes les restriccions: màxim de publicacions pròpies, màxim de coautors i màxim de publicacions per coautor.

C.5 Consulta generada per a un usuari

La consulta es genera a partir de tres grups d'informació: paraules clau introduïdes per l'usuari, paraules extretes de les seves publicacions i paraules extretes de les publicacions dels seus coautors. No obstant, aquesta prova s'ha realitzat sense paraules clau introduïdes per l'usuari, així que la consulta es basarà només en les dades obtingudes pel sistema.

Tal com s'ha vist en el capítol de disseny del sistema de recomanació, se li vol donar un pes diferent a cadascun d'aquests grups en funció dels paràmetres α i β. De tal manera que el pes de les paraules clau introduïdes per l'usuari sigui α, el pes de les seves publicacions sigui $(1-\alpha)\beta$ i el de les publicacions dels coautors $(1-\alpha)(1-\beta)$. S'han fet proves amb varis valors d'aquestes variables i s'ha decidit utilitzar $\alpha = 50\%$ i $\beta = 70\%$

A continuació es revisarà el procés de generació de la consulta amb els valors de α i β triats, utilitzant l'usuari "Ricard Gavaldà" per seguir amb l'exemple de l'apartat anterior.

En primer lloc, el text format per la successió dels títols de l'usuari de obtinguts es passa per l'analitzador, s'agrupen els termes coincidents i s'obtenen les seves freqüències. Per tal d'optimitzar la consulta, es redueix la llista a un màxim de 100 termes diferents. Per establir la prioritat a l'hora d'escollir els termes es basa en heurístiques tal com s'ha vist en el capítol 6. Per això, prèviament s'ordena la llista de termes descendentment per la freqüència i, en segon lloc, descendentment pel número de caràcters dels termes. A continuació es mostra una taula amb alguns dels primers termes d'aquesta llista, la seva freqüència i el *boost* final que s'assignarà al terme. El *boost* es calcula dividint la freqüència de cada terme entre el número total de termes, que en aquest cas és 144, i multiplicat pel pes del camp, que és 35 (ja que se li volia donar un $50\% \cdot 70\% = 35\%$ d'importància sobre la consulta total i s'ha fet sobre 100 per tal que no surtin xifres insignificants).

Terme	**Freqüència**	***Boost* final**
learning	10	2.43
data	9	2.19
adaptive	7	1.7
streams	6	1.46
evolving	4	0.97
web	3	0.73
probabilistic	2	0.49
resources	2	0.49
automata	2	0.49
session	2	0.49
centers	2	0.49
methods	2	0.49
machine	2	0.49
finite	2	0.49
energy	2	0.49
based	2	0.49
2009	2	0.49
prioritization	1	0.24
virtualization	1	0.24
classification	1	0.24

Taula C.1: Termes obtinguts de les publicacions de l'autor Ricard Gavaldà, la seva freqüència i el *boost* final assignat.

En el següent pas, s'uneixen tots els títols obtinguts dels coautors i, amb el text resultant, es repeteix el procés anterior. A continuació es mostren alguns dels termes obtinguts en aquest pas. El *boost* es calcula igual que abans, dividint la freqüència entre el número total de termes i multiplicat pel pes del camps. En aquest cas, el número total de termes és 285 i el pes del camp, 15.

Terme	**Freqüència**	***Boost* final**
complexity	13	0.68
2010	9	0.47
learning	7	0.37
query	7	0.37
logic	6	0.32
applications	5	0.26
performance	5	0.26
discovery	5	0.26
knowledge	5	0.26
proceedings	4	0.21
conference	4	0.21
algorithms	4	0.21
resolution	4	0.21
analysis	4	0.21
optimal	4	0.21
games	4	0.21
ecml	4	0.21
pkdd	4	0.21
reconstructing	3	0.16
distributed	3	0.16

Taula C.2: Termes obtinguts de les publicacions dels coautors del Ricard Gavaldà, la seva freqüència i el *boost* final assignat.

Finalment, la consulta es construeix amb cadascun d'aquests termes amb el pes que s'ha calculat per a cadascun. En la següent secció es podran observar els resultats que s'han obtingut amb aquesta consulta, entre d'altres.

C.6 Recomanacions obtingudes

Finalment veiem les cinc primeres recomanacions resultants per a l'usuari Ricard Gavaldà (identificat amb el seu perfil a DBLP i sense paraules clau afegides explícitament) en la conferència que estem utilitzant en aquest conjunt de proves.

- Adaptive XML Tree Classification on Evolving Data Streams
  Albert Bifet, Ricard Gavaldà
- Learning from Multi-Label Data
  Grigorios Tsoumakas, Min-Ling Zhang, Zhi-Hua Zhou
- Learning Monotone Models from Data
  Rob Potharst, Ad Feelders
- Music and Machine Learning
  Rafael Ramirez, Darrell Conklin, Christina Anagnostopoulou, Jose Manuel Iniesta

- Learning and data Mining for Robotics
  Einoshin Suzuki, Michele Sebag

Si ho provem amb un altre usuari del que recupera informació diferent, veiem que retorna unes recomanacions diferents. A continuació es mostra un altre exemple de resultats de recomanació, però aquest per a l'usuari Jordi Delgado (identificat amb el seu perfil a DBLP i sense paraules clau afegides explícitament).

- Explorative Analytics of Information Networks
  Andreas Nuernberger, Michael Berthold
- Communication-efficient classification in P2P networks
  Hock Hee Ang, Vivekanand Gopalkrishnan, Wee Keong Ng, Hoi Chu Hong
- Inference and Validation of Networks
  Ilias Flaounas, Marco Turchi, Tijl De Bie, Nello Cristianini
- Debt Detection in Social Security by Sequence Classification Using Both Positive and Negative Patterns
  Yanchang Zhao, Huaifeng Zhang, Shanshan Wu, Jian Pei, Longbing Cao, Chengqi Zhang, Hans Bohlscheid
- Neural Networks for State Evaluation in General Game Playing
  Daniel Michulke, Michael Thielscher

Finalment, mostrem un altre exemple dels resultats obtinguts en el mateix cas que la prova anterior, però afegint les següents paraules clau en les preferències del seu perfil: *complex neworks, power laws, zipf, multiagents, cooperation, statistical physics, autoorganization, self-organization, small world, ants, network topology, self-referential.*

- Explorative Analytics of Information Networks
  Andreas Nuernberger, Michael Berthold
- Statistical Relational Learning with Formal Ontologies
  Achim Rettinger, Matthias Nickles,Volker Tresp
- Inference and Validation of Networks
  Ilias Flaounas, Marco Turchi, Tijl De Bie, Nello Cristianini
- Methods for Large Network Analysis
  Vladimir Batagelj
- Communication-efficient classification in P2P networks
  Hock Hee Ang, Vivekanand Gopalkrishnan, Wee Keong Ng, Hoi Chu Hong

Veiem que tres de les recomanacions es conserven respecte a la prova anterior, encara que algunes en posicions diferents, mentre que les altres dos han canviat.