Cercador documental amb lematització

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Sistemes

realitzat per

Josep Boix Requesens

i dirigit per

Asier Ibeas

Escola d'Enginyeria

Sabadell, *Juny* de *2012*

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol del projecte: Cercador documental amb lematització.	
Autor[a]: Josep Boix Requesens	**Data:** *Juny 2012*
Tutor[a]/s[es]: Asier Ibeas	
Titulació: Enginyeria tècnica en informàtica de sistemes.	

Paraules clau (mínim 3)

- Català: Lucene, Java, processament de llenguatge natura, lematització.
- Castellà: Lucene, Java, procesamiento del lenguaje natural, lematización.
- Anglès: Lucene, Java, natural language processing, lemmatization.

Resum del projecte (extensió màxima 100 paraules)

- Català: Un cercador documental d'arxius locals, en diferents formats, que permet l'extracció i indexació de lemes en diferents idiomes (principalment català, castellà i anglès) en el que s'habilita la cerca a través d'una pàgina web allotjada en un servidor local.

- Castellà: Un buscador documental de ficheros locales, de distintos formatos, que permite la extracción e indexación de lemas en distintos idiomas (principalmente catalán, castellano e inglés) y es accesible a través de una pàgina web alojada en un servidor local.

- Anglès: A documental search engine for local hosted files, several formats supported, which extracts and indexes lemmas for different languages (mainly catalan, spanish and english) and it's accesible through a web application running on a local server.

Índex

1. Introducció

La gestió de grans quantitats d'informació és una necessitat que ja no només afecta a grans empreses sinó també al usuari. Actualment els motors de cerca (ja siguin per a cercador web generalistes, verticals o propis del sistema operatiu) utilitzen pretractament de la informació continguda en un document per tal de facilitar-ne la cerca posteriorment, transformant les paraules del document a través de diferents tècniques de recuperació d'informació.

Aquestes tècniques varien segons els requeriments de l'aplicació, però estan sempre en estreta relació amb el processament del llenguatge natural, algunes d'aquestes tècniques inclouen:

- *Exact Keyword Matching*: Com el nom indica, la consulta únicament retorna aquells documents que continguin paraules que coincideixen exactament amb la petició de consulta. S'aprecia ràpidament que els resultats obtinguts seran molt rellevants i que no hi haurà falsos positius en la cerca, tot i així una gran quantitat de documents que, amb tota certesa, estan relacionats amb la petició, no seran mai mostrats, dificultant el procés de cara a l'usuari. Aquest mètode de cerca és adient per camps amb un text petit (un autor, el títol d'un llibre o d'una pel·lícula, el nom d'un restaurant, etc.), ja que prèviament hi ha coneixement per part de l'usuari del que vol trobar.

- *Stemming*: El *Stemming* és la reducció d'una paraula a la seva base o arrel. Tot i que existeixen mètodes molt sofisticats de *Stemming*, el problema principal que suposa és que les paraules són reduïdes independentment del seu context o categoria gramatical, així doncs pot augmentar el número de documents que una consulta reduint-ne la rellevància. Per exemple: Suposem que tenim una implementació d'un *stemmer* molt agressiu que redueix el la paraula gateja a l'arrel gat, sembla poc lògic que una consulta sobre documents que parlin de gats retorni documents que continguin el verb gatejar. Tot i així, aquest mètode és una millora sobre el *Exact Keyword Matching*, ja que garanteix l'extracció de tots els documents relacionats amb la cerca – i alguns altres. A nivell de rendiment és una tècnica molt òptima. Requereix el coneixement previ del llenguatge del document i de la consulta.

- *Synonym Search*: L'extracció de sinònims en temps d'indexació ajuda a buscar documents rellevants per a consultes que, d'altra manera, no obtindrien cap resultat. Per exemple, si l'únic document que el nostra cercador ha indexat conté la paraula: saltar no obtindrem cap resultat per la cerca botar. Tot i així, si som capaços de extreure els sinònims, si que obtindrem resultats. S'ha d'anar amb compte amb aquesta tècnica ja que diferents sinònims poden divergir lleugerament de significat per exemple, saltar una pilota no és el mateix que botar una pilota. Aquesta tècnica és força útil com a complement a altres tècniques de pretractament de la informació. Requereix el coneixement previ del llenguatge del document (però no de la consulta).

- *Lemmatization*: La lematització és una millora significant sobre el *Stemming*, ja que redueix les paraules a la seva arrel gramatical. En el cas que hem vist en el *Stemming*, la paraula gateja es redueix a l'arrel gatejar de manera que una consulta sobre gats ja no retorna documents sobre gatejar. Un pas previ a lematitzar una paraula és conèixer la seva categoria gramatical. Requereix el coneixement previ del llenguatge del document i de la consulta.

Altres tècniques existeixen – per exemple es poden indexar paraules utilitzant l'algoritme *Metaphone* per buscar paraules per sonoritat en comptes de text – però en el cas que ens ocupa ens centrarem en la lematització.

Les avantatges de la lematització en termes de rellevàncies són molt grans respecte altres mètodes d'anàlisis de la informació. Tot i així, les solucions que existeixen actualment són propietàries o comercials; la majoria de solucions lliures es centren en la lematització en l'Anglès. L'objectiu principal d'aquest projecte és superar aquesta barrera i crear un cercador capaç de lematitzar en idiomes diferents de l'anglès (amb prioritat pel Català i el Castellà).

1.2. Descripció de la memòria

2. Estudi de Viabilitat: L'objectiu d'aquest apartat és descobrir els punts forts i febles de la solució proposada a la present problemàtica, juntament amb una descripció comparativa dels recursos disponibles i un anàlisis dels costos envers els beneficis.

3. Anàlisi: L'objectiu d'aquest apartat és donar forma a la solució presentada en l'estudi de viabilitat, mitjançant: diagrames de cassos d'ús, diagrames de flux i diagrames funcionals de cada mòdul del projecte.

4. Tecnologies: Descripció de les tecnologies utilitzades durant el projecte. Cada apartat engloba una única tecnologia. Aquest apartat inclou tant tecnologies que s'han utilitzat en l'entorn de treball (eines per la gestió i construcció del projecte) com tecnologies que s'han utilitzat en el codi del projecte (llibreries). Es reserva un apartat per tecnologies desenvolupades dins del marc del projecte i una breu descripció de perquè les tecnologies existents no es podien adoptar per resoldre la problemàtica presentada.

5. Implementació: Descripció de la metodologia de desenvolupament del projecte i la seva relació amb les tecnologies utilitzades. S'inclou aquí una descripció del desenvolupament de les proves unitàries i proves d'integració juntament amb els resultats obtinguts.

6. Conclusions: Conclusions finals sobre el projecte exploren si s'han assolit els objectius i els requisits descrits en l'estudi de viabilitat, quines dificultats s'han trobat durant el desenvolupament del projecte i quines experiències i coneixements ha aportat la seva realització.

2. Estudi de Viabilitat

Aquest apartat exposa i explora les diferents solucions a la problemàtica del present projecte amb l'objectiu de planificar-ne la seva implementació a través de la definició de requisits i objectius concrets i l'enumeració de les diferents tasques que s'han d'acomplir per tal d'assolir el resultat desitjat.

2.1. Descripció

En aquest projecte es desenvoluparà una cercador local que permetrà el pretractament de documents, en diferents formats, a través de la tècnica coneguda com a lematització amb l'objectiu de millorar la rellevància dels resultats en diferents idiomes. Aquest cercador explorarà el sistema de fitxers, ja sigui d'una màquina personal o d'un servidor, i n'habilitarà la cerca a través d'una pàgina web allotjada en la mateixa màquina.

Tipologia i paraules claus:

- Tipologia del projecte: Desenvolupament i investigació.
- Paraules claus: Lucene, Java, processament del llenguatge natural.

Definicions, acrònims i abreviacions:

- *Processament del llenguatge natural* (PLN).
- *Information retrieval* (IR).
- *Java Native Interface* (JNI).
- *Integrated development environment*(IDE).

Parts interessades:

1. **Nom:** Josep Boix Requesens.
   Descripció: Tècnic.
   Responsabilitat: Definició de les tasques, desenvolupament de l'aplicació.

2. **Nom:** Asier Ibeas.
   Descripció: Director del projecte.
   Responsabilitat: Supervisar i avaluar l'alumne.

Producte i documentació del projecte:

1. Aplicació informàtica.
2. Memòria de projecte.

2.2. Objectius

1. **Crear un motor de recol·lecció de documents**: Ha de ser capaç de descobrir nous documents i mantenir els documents que ja estan indexats.
2. **Crear un motor de indexació de documents**: Ha de ser capaç de rebre el contingut d'un document i indexar-lo per la posterior cerca.
3. **Crear un motor de cerca sobre els índexs**: Ha de ser capaç de rebre peticions i retornar una llista resultats per ordre de rellevància.
4. **Potenciar les peticions de cerca**: A través de filtres i opcions de cerca avançades.
5. **Pretractament dels documents amb PLN**: Extracció dels lemes de cada paraula en relació a la seva categoria gramatical.
6. **Desenvolupar una demostració gràfica que integri el motor**: Una aplicació web que permeti realitzar cerques sobre els documents indexats.
7. **Compatibilitat amb una extensa base de formats**: Microsoft Office, PDF, etc.

Esquema de prioritats:

	Crític	Prioritari	Secundari
1	X		
2	X		
3	X		
4	X		
5	X		
6		X	
7		X	

2.3. Requisits del sistema

Requisits funcionals

1. Recol·lecció de documents.
2. Indexació de documents
3. Gestió i manteniment dels documents indexats.
4. Habilitar la cerca sobre els índexs obtinguts.
5. Crear un llenguatge de peticions.
6. Lematització de textos plans.
7. Detecció automàtica d'idiomes en els fitxers de entrada.
8. Exportació dels resultats en XML i JSON.
9. Control de l'aplicació a través de fitxers de configuració.

Requisits no funcionals

1. Usabilitat en la demostració gràfica.
2. Codificació intuïtiva de l'aplicació utilitzant patrons de disseny i bones pràctiques.
3. Documentar el codi.

Restriccions del sistema:

1. La comunicació i configuració de l'aplicació ha de donar-se en temps d'execució i ha de ser independent del llenguatge utilitzat.
2. El projecte s'ha de finalitzar abans del 24 de juny.
3. L'aplicació s'ha de desenvolupar sobre programari lliure.

Catalogació i priorització dels requisits:

	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RF9	RNF1	RNF2	RNF3
Essencial	X	X		X		X					X	X
Condicional			X		X			X	X	X		
Opcional							X					

Relació entre els objectius i els requisits:

	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RF9	RNF1	RNF2	RNF3
O1	X		X						X		X	X
O2		X	X						X		X	X
O3				X				X	X		X	X
O4					X						X	X
O5						X	X				X	X
O6					X			X	X	X	X	X
O7	X	X	X								X	X
O8											X	X

2.4. Alternatives i selecció de la solució

El present projecte combina diferents tecnologies per tal de assolir els objectius. En la proposició d'alternatives ens centrem en aquelles que suposen un canvi més gran en l'arquitectura de l'aplicació:

- El motor de recuperació d'informació.
- El motor de lematització.

Alternatives sobre el motor de recuperació d'informació:

- **Alternativa 1:** Egothor.
  Característiques: Cercador complet, programari lliure, ready-to-use, documentació pobre (només en Txec).
  Costos: cap.
- **Alternativa 2:** Xapian.
  Característiques: Programari lliure.
  Costos: cap.
- **Alternativa 3:** Lucene.
  Característiques: Programari lliure, comunitat d'usuaris extensa, flexible, Java.
  Costos: cap.

Comparativa entre les alternatives:

	Cost d'adquisició	Cost d'adaptació	Nous recursos	Suport	Nivell integració	Complexitat	Formació
1)	0€	Alt	No cal	Web	Baix	Alta	Desconegut
2)	0€	Alt	Adaptable	Web	Alt	Mitjana	Desconegut
3)	0€	Baix	Adaptable	Web	Alt	Mitjana	Desconegut

Alternatives sobre el motor de lematització:

- **Alternativa 1:** Freeling.
  Característiques: Lematització, català, castellà, anglès, portuguès, italià, gallec, asturià, gal·lès, c++, Java API.
  Costos: cap.
- **Alternativa 2:** NLTK
  Característiques: Lematització, anglès, python.
  Costos: cap.

Comparativa entre les alternatives:

	Cost d'adquisició	Cost d'adaptació	Nous recursos	Suport	Nivell integració	Complexitat	Formació
1)	0€	Mig	No cal	Web	Mig	Alta	Desconegut
2)	0€	Alt	Adaptable	Web	Mig	Alta	Desconegut

Solució proposada

En relació a les alternatives per llibreries IR hem seleccionat Lucene per ser la llibreria IR més extensa i flexible de les tres. Tot i que Egothor cumpleix molts dels requeriments del sistema, el cost d'adaptació és massa gran.

En relació al motor PLN hem seleccionat Freeling perquè és dels pocs motors que inclou llenguatges molt específics (com el català o el gal·lès) que aporten molt de valor a l'aplicació.

2.5. Pla del projecte

2.5.1. WBS (Work Breakdown Structure)

Fases i activitats del projecte:

Fases	Descripció
Definició	Definició del projecte i presentació de la proposta.
Planificació	Estudi de viabilitat, document del pla del projecte.
Anàlisis	Anàlisis dels requeriments i les restriccions.
Disseny	Disseny de l'arquitectura del software.
Desenvolupament	Codificació.
Test	Test en entorn real.
Implantació	L'aplicació s'instal·la en un entorn real.
Documentació	Desenvolupament de la memòria.
Defensa	Defensa del projecte davant del tribunal.

Milestones:

Nom	Descripció	Data
Definició	Matriculació	01.10.2011
Planificació	Aprovació	01.11.2011
Anàlisis	Aprovació	01.11.2011
Disseny	Aprovació	15.11.2011
Desenvolupament	Aprovació	03.01.2012*
Test	Aprovació	28.01.2012*
Documentació	Aprovació	11.02.2012*
Defensa	Avaluació	Per definir*

* Dates orientatives

2.5.2. Recursos

Recursos del projecte:

Recursos humans	Valoració
Cap de projecte	100€/h
Tècnic	20€/h

Recursos materials: Com a recursos materials s'utilitzaran els recursos materials disponibles a l'entitat. El desenvolupament es farà utilitzant únicament software lliure sobre una computadora personal.

Calendari dels recursos:

Cap de projecte: Definició, Planificació, Anàlisis, Defensa.
Tècnic: Definició, Planificació, Anàlisis, Disseny, Desenvolupament, Test, Documentació, Defensa.

Els recursos materials s'utilitzaran en les fases de desenvolupament, test, documentació i defensa.

2.5.3. Calendari del projecte

Dependències:

El desenvolupament del projecte segueix un model de prototipatge. La finalització de una fase no implica el començament de una altre. Es crearà una base tècnica ampliable del projecte sobre la que s'afegiran progressivament tots els requeriments definits en l'estudi de viabilitat.

La fase de test en entorn real no es portarà a terme fins que no s'hagin satisfet tots els requeriments, tot i així el model preveu test unitaris de les diferents mòduls durant el desenvolupament.

La creació de la documentació es durà a terme en paral·lel al projecte, tot i que es reserva un període de temps després del tancament de codi per acabar de revisar la memòria.

Quadre de tasques del projecte:

Nom	D.	Com.	Fi	Prec.
1. Planificació i Anàlisis	6	25.10.2011	01.11.2011	
2. Estudi de viabilitat	3	25.10.2011	27.10.2011	
3. Document del plà de projecte	3	28.10.2011	01.11.2011	**2**
4. Disseny	10	02.11.2011	15.11.2011	**1**
5. Definició dels mòduls	1	02.11.2011	02.11.2011	
6. Definició dels fitxers de configuració	3	03.11.2011	05.11.2011	**5**
7. Definició del llenguatge de peticions	3	08.11.2011	10.11.2011	**6**
8. Definició de la demostració web	3	11.11.2011	15.11.2011	**7**
9. Desenvolupament	35	16.11.2011	03.01.2012	**4**
10. Col·lector	11	16.11.2011	30.11.2011	
11. Implementació del motor de col·lecció	7	16.11.2011	24.11.2011	
12. Recol·lecció de documents	2	25.11.2011	26.11.2011	**11**
13. Recursiva sobre un sistema de fitxers	2	25.11.2011	26.11.2011	
14. Simple sobre un directori	2	25.11.2011	26.11.2011	
15. Analitzador	2	29.11.2011	30.11.2011	**12**
16. Integrar Tika	2	29.11.2011	30.11.2011	
17. Indexador	19	25.11.2011	14.12.2011	
18. Implementació del motor d'indexació	7	25.11.2011	03.12.2011	**11**
19. Processament del llenguatge natural	7	06.12.2011	14.12.2011	**18**
20. Integrar FreeLing	7	06.12.2011	14.12.2011	
21. Detecció automàtica del llenguatge	3	06.12.2011	08.12.2011	
22. Cercador	14	06.12.2011	23.12.2011	
23. Implementació del motor de cerca	7	06.12.2011	14.12.2011	
24. Analitzador de peticions	7	15.12.2011	23.12.2011	**23**
25. Exportació XML	2	15.12.2011	16.12.2011	**23**
26. Exportació JSON	2	15.12.2011	16.12.2011	**23**
27. Demostració Web	7	24.12.2011	03.01.2012	**10,17,22**
28. Cerca general	7	24.12.2011	03.01.2012	
29. Cerca de documents	7	24.12.2011	03.01.2012	
30. Proves	19	04.01.2012	28.01.2012	**9**
31. Període de proves	19	04.01.2012	28.01.2012	
32. Documentació	64	16.11.2011	11.02.2012	**4**
33. Documentació del codi	35	16.11.2011	03.01.2012	
34. Memòria	10	31.01.2012	11.02.2012	**30**

Calendari temporal

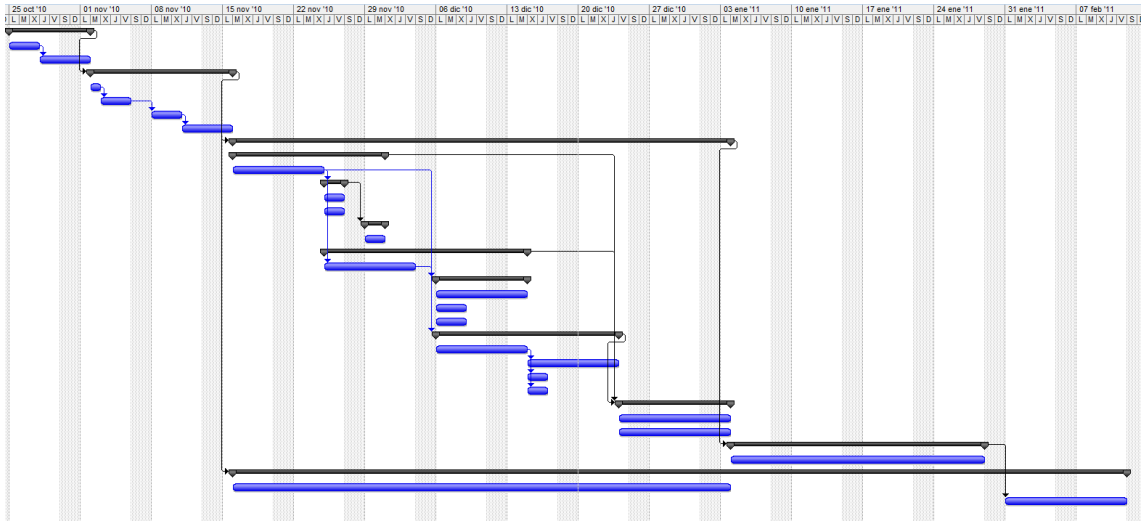


fig. 1 Calendari temporal

2.5.4. Avaluació de riscos

Llista de riscos

1. Planificació temporal optimista: estudi de viabilitat. No s'acaba en la data prevista, augmenten els recursos.
2. Manca d'alguna tasca necessària: estudi de viabilitat. No es compleixen els objectius del projecte.
3. Canvi de requisits: estudi de viabilitat, anàlisi. Endarreriment en el desenvolupament i resultat.
4. Eines de desenvolupament inadequades: implementació. Endarreriment en la finalització del projecte, menys qualitat.
5. No es fa correctament la fase de test: desenvolupament, implementació. Manca de qualitat, deficiències en l'operativa, insatisfacció del usuari.
6. Inviabilitat d'alguns dels requisits: estudi de viabilitat, implementació. Endarreriment en el desenvolupament, no es compleixen els objectius.

Catalogació de riscos

	Probabilitat	**Impacte**
R1	Alta	Crític
R2	Alta	Crític
R3	Mitjana	Crític
R4	Baixa	Crític
R5	Baixa	Crític
R6	Baixa	Crític

Pla de contingència

	Solució que cal adoptar
R1	Ajornar alguna funcionalitat prioritària o secundaria.
R2	Revisar l'estudi de viabilitat, modificar la planificació.
R3	Modificar la planificació.
R4	Ajornar funcionalitats, modificar la planificació.
R5	Dissenyar tests amb antelació, realitzar tests automàtics.
R6	Descartar funcionalitats, modificar la planificació.

2.5.5. Pressupost

Estimació cost de personal

Costos de personal imputables directament al projecte.

	Temps	Cost total
Cap de projecte	10h	1000€
Tècnic	200h	4000€

Total: 5000€

Estimació cost dels recursos

Amortització dels recursos propis del projecte.

	Cost amortització	Cost unitari	Període amortització	Període utilització
Amortització PC tècnic	66.6€	800€	36 mesos	3 mesos
Amortització MS Office	20.8€	250€	36 mesos	3 mesos
Amortització MS Project	30€	360€	36 mesos	3 mesos

Total: 117.46€

Resum i anàlisis cost/benefici

Cost de desenvolupament del projecte: 5000€
Cost d'amortització del material: 150.8€

Total: 5150.8€

Per aquest projecte no hi ha beneficis econòmics previstos.

2.6. Conclusions

A mode de resum de l'estudi de viabilitat, entenem que els beneficis i els inconvenients de la realització del present projecte són:

- **Beneficis:**
  - Millora de la indexació utilitzant PLN.
  - Cerques més potents.
  - Programari fàcilment ampliable (multi idioma, multi format...).
  - Millora de l'accessibilitat.
- **Inconvenients:**
  - Desconeixement de les possibles complicacions durant el desenvolupament de mòduls PLN.

Es conclou que el projecte és viable ja que la inversió econòmica és mínima i els beneficis que suposa, a nivell d'investigació sobre tecnologies de lematització integrades en un motor de cerca, són molt positius.

3. Anàlisis

En aquest apartat es poden veure els diferents cassos d'ús de l'aplicació juntament amb un recull de diagrames de flux i del sistema. Per entendre la relació entre aquests diagrames i la implementació final de l'aplicació cal consultar l'apartat *5.1. Implementació*.

3.1. Cassos d'ús

Existeixen dos actors principals en el programari aquí descrit:

- Usuari: qui interactua amb l'aplicació per buscar i recuperar documents de l'índex.
- Administrador de sistemes: qui s'encarrega del bon funcionament de l'aplicació i d'establir els paràmetres de configuració segons els requeriments del client. És també la seva responsabilitat desplegar la interfície d'usuari; en el context d'aquest projecte, el desplegament d'un servidor i de la demostració web.

3.1.1. Cassos d'ús de l'usuari

L'usuari ha de ser capaç de:

- Buscar documents: introduir una petició en un formulari i obtenir els documents que coincideixen amb aquests paràmetres. En aquest mateix context ha de ser capaç de: filtrar la cerca per rangs de dates o per antiguitat, filtrar la cerca pel tipus de document, filtrar la cerca pel llenguatge del document o ordenar la cerca segons la data de la última edició del document.
- Obtenir documents: L'usuari ha de ser capaç d'obtenir el document original, això significa que l'aplicació ha de poder redirigir l'usuari a la ubicació d'aquest document o proporcionar el document en cas que l'usuari no pugui ser redirigit (per exemple, si el document esta emmagatzemat en local).

Totes aquestes operacions les ha de poder realitzar des de la mateixa interfície – renderitzar el document es realitza en la màquina de l'usuari sobre la seva aplicació preferida.

La següent figura il·lustra els cassos d'ús aquí descrits:

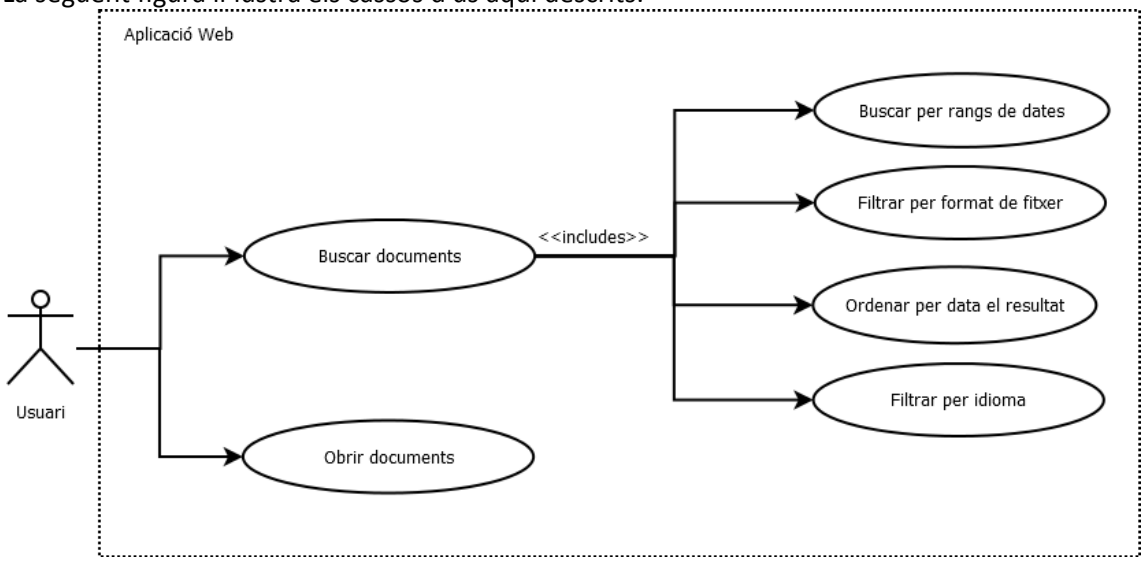


fig. 2 Casos d'ús de l'usuari

3.1.2. Cassos d'ús de l'administrador

L'administrador ha de ser capaç de:

- Afegir, eliminar i modificar les fonts del buscador: segons els requeriments dels producte és possible que les fonts definides durant el procés de desplegament no siguin les definitives, és per això que l'administrador ha de ser capaç d'afegir i eliminar fonts fàcilment des del fitxer de configuració de l'aplicació.
- Filtrar les fonts el llenguatge: és possible que l'aplicació només requereixi indexar documents en un idioma o que el número d'idiomes d'una font sigui limitat. En aquests cassos l'administrador ha de ser capaç de indiciar a través dels fitxers de configuració que els documents indexats només han de ser analitzats en un o un subgrup dels idiomes disponibles.
- Filtrar fonts per format del fitxer: de la mateixa manera que l'idioma pot ser un opció de filtrat també ho pot ser el format del fitxer: indexar només documents en format *Microsoft Office*, *PDF*, *Open Office*, etc. o qualsevol subgrup d'aquests.
- Canviar l'algoritme d'anàlisis: L'algoritme d'anàlisis del contingut del document per defecte utilitza *FreeLing*, tot i així és possible modificar l'algoritme d'anàlisis per un desenvolupat ad hoc o pels algoritmes d'anàlisis que ofereix *Lucene*.
- Programar el procés d'anàlisis: L'administrador ha de ser capaç de programar el procés d'anàlisis: Si s'ha d'executar en un horari concret i cada quan s'ha d'executar.
- Accedir al registre de l'aplicació: El registre de l'aplicació ajuda al l'administrador a buscar i solucionar possibles problemes que sorgeixin en el marc de l'aplicació (ja siguin relacionats amb la indexació o amb la cerca), l'administrador ha de ser capaç d'accedir i llegir aquests registres – és important que els registres siguin clars i que només es registri informació rellevant.

La següent figura il·lustra els cassos d'ús aquí descrits:

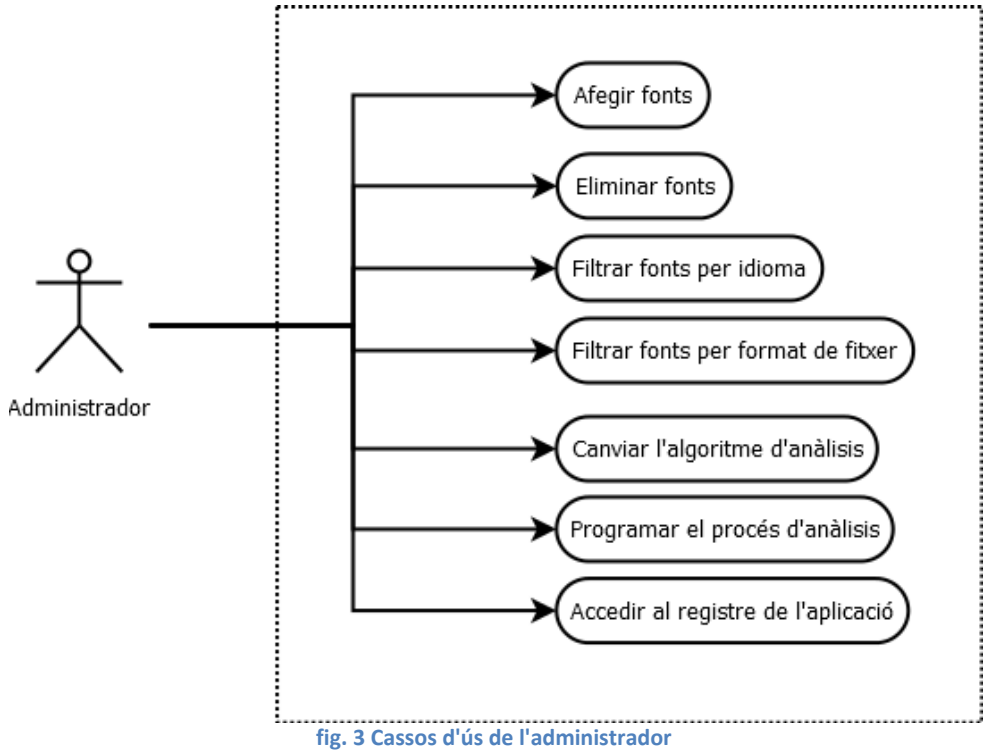


fig. 3 Cassos d'ús de l'administrador

3.2. Diagrama del sistema

Un cop entesos els casos d'ús i els requisits del sistema el següent pas és dissenyar una aplicació que els compleixi. El sistema que presentem no és en absolut trivial; es compon de diferents mòduls i processos independents que s'entrellacen sutilment. És doncs de vital importància entendre quina és la visió general del sistema: De quins mòduls es compon? Quins processos interactuen en ell? I quins són els recursos que utilitza? Per tal de respondre a aquestes qüestions, i abans d'entrar en detalls, hem dissenyat el següent diagrama que ajuda a comprendre les diferents parts del programari[1]:

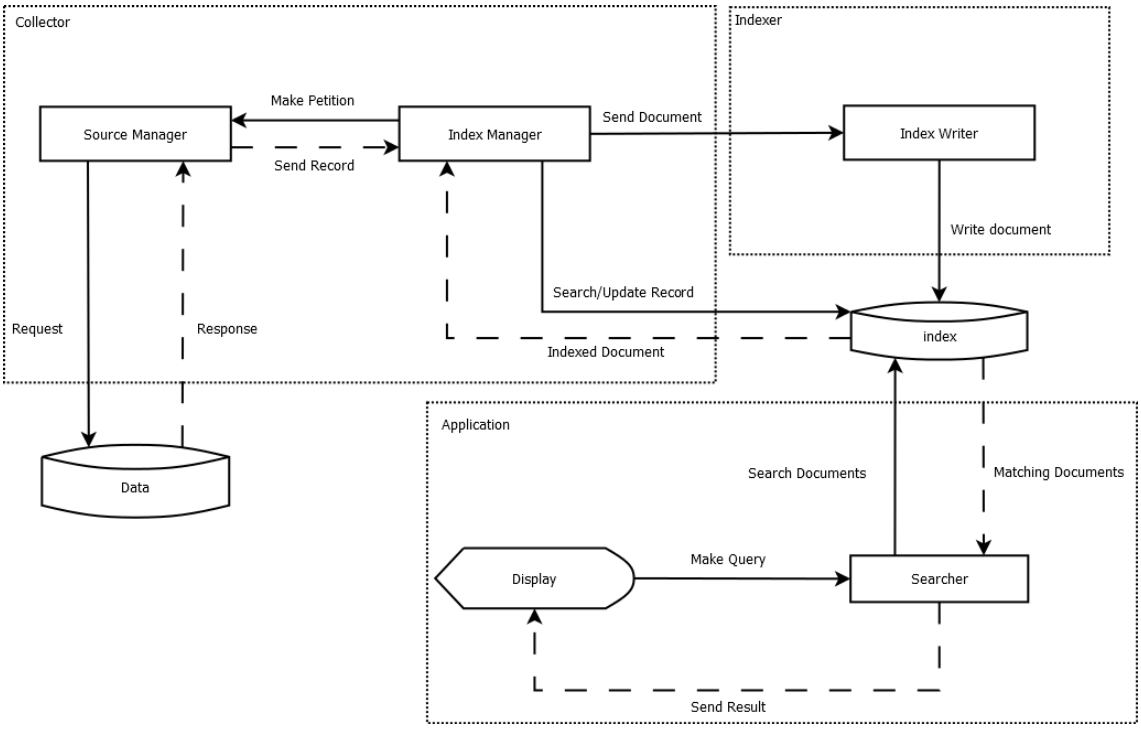


fig. 4 Diagrama del sistema

Com es pot observar en la *fig 4.* existeixen dos blocs principals en l'estructura del programari: *backend* i *frontend*.

Backend

El processos de col·lecció i indexació formen part del *backend* de l'aplicació i estan en directe relació amb el flux d'indexació:

- **Col·lector**: el col·lector té la responsabilitat de extreure els documents de la font i preparar-los per ser indexats. Per aconseguir això es compon de dos mòduls: *Source Manager* – que s'encarrega de comunicar-se amb la font dels documents i extreure'n el contingut (text, data de la última modificació, format i *metadata*) – i *Index Manager* – que s'encarrega de mantenir la coherència dels documents i d'afegir aquella

[1] El diagrama està deliberadament en anglès; els noms dels diferents mòduls es trobaran escrits de la mateixa manera en el codi de l'aplicació.

informació que depèn directament de la configuració de l'índex. Al final d'aquest procés es crea un objecte denominat *Document* que ja esta preparat per ser indexat.
- **Indexador:** El indexador es compon d'un únic mòdul: *Index Writer* – encarregat de guardar el document a l'índex.

Aquests processos no s'executen en temps real, això implica que el contingut generat en l'índex no es correspon amb el contingut de la font dels documents en el moment de realitzar la petició sinó que és una imatge d'un passat recent – l'antiguitat d'aquesta imatge depèn de la programació del *backend*, si el procés esta programat per esser executat cada 24 hores aquesta imatge tindrà una antiguitat de entre 0 i 24 hores segons el moment en que s'ha realitzat la petició.

Ambdós processos estan separats virtualment en dos o més fils d'execució i es comuniquen entre elles a través d'una cua. Aquesta separació és necessària per optimitzar el temps de processament degut a la presència d'operacions d'entrada/sortida; en el cas del col·lector a l'hora de obtenir nous documents i en el cas del indexador a l'hora de escriure els documents a l'índex. Depenent de les limitacions del sistema pot ser beneficiari introduir múltiples col·lectors o minvar-ne el número, prendre d'aquesta decisió és responsabilitat de l'administrador del sistema.

Frontend

Els processos de cerca i la interfície de l'usuari formen part del *frontend*.

La interfície de l'usuari (denotada com a *display* en la figura), té dos responsabilitats:

- Proporcionar un formulari on l'usuari sigui capaç d'introduir peticions.
- Mostrar la informació dels documents extrets per a una petició i proporcionar una manera de navegar entre ells.

Els processos de cerca estan en directe relació amb el flux del procés de peticions.

Ambdós processos es comuniquen a través de protocols RPC de manera que el procés de peticions resideix en la mateixa màquina on l'índex es troba físicament però la interfície pot residir en una màquina diferent o un procés físicament separat.

3.3 Diagrames de flux

3.3.1. Flux del procés de indexació

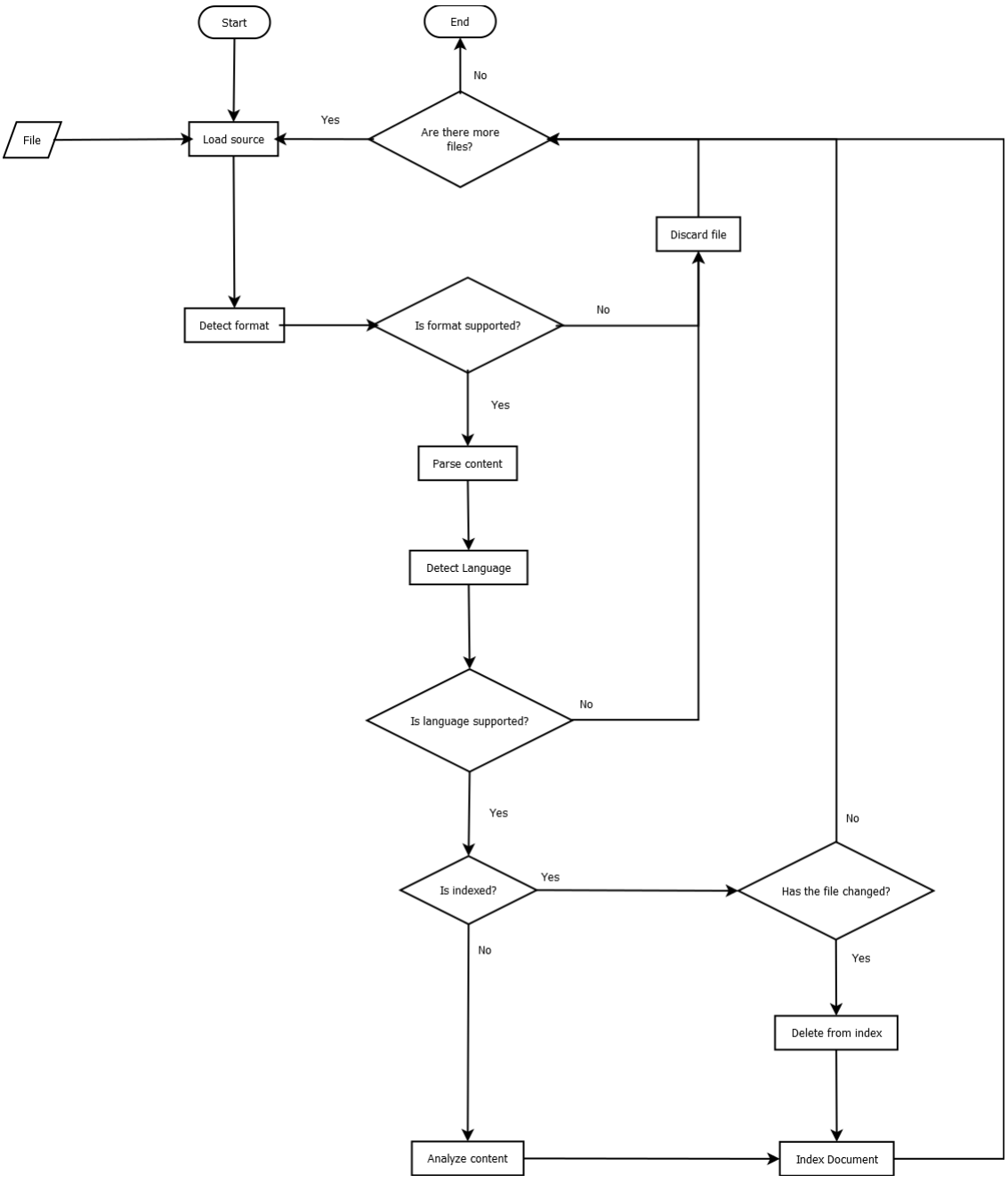


fig. 5 Flux del procés de indexació

El procés d'indexació comença en l'obtenció de documents; per defecte els documents seran col·leccionats únicament del sistema de fitxers – és a dir en local – però el disseny ha de permetre la implementació de noves estratègies de descobriment (e.g.: robots i *crawlers*).

El mòdul de descobriment proporcionarà diferents dades a la lògica d'indexació: format del document, contingut del document, última data de modificació i un plataforma per recuperar el document de la seva font original. El format del document és utilitzat per la lògica

d'indexació per determinar si és o no un document vàlid en respecte a la configuració de l'índex que s'està creant o actualitzant.

A continuació s'extreu el contingut el document, d'aquest se'n detecta el llenguatge automàticament – permet filtrar el document, de la mateixa manera que es fa amb el format. Del contingut se'n extreu a una clau *hash* a través d'algoritmes criptogràfics – que permet identificar documents duplicats.

Si el document existeix i la seva última data de modificació ha canviat s'esborra l'entrada original del índex i es torna a indexar, en cas contrari no es fa res.

3.3.2. Flux del procés de peticions

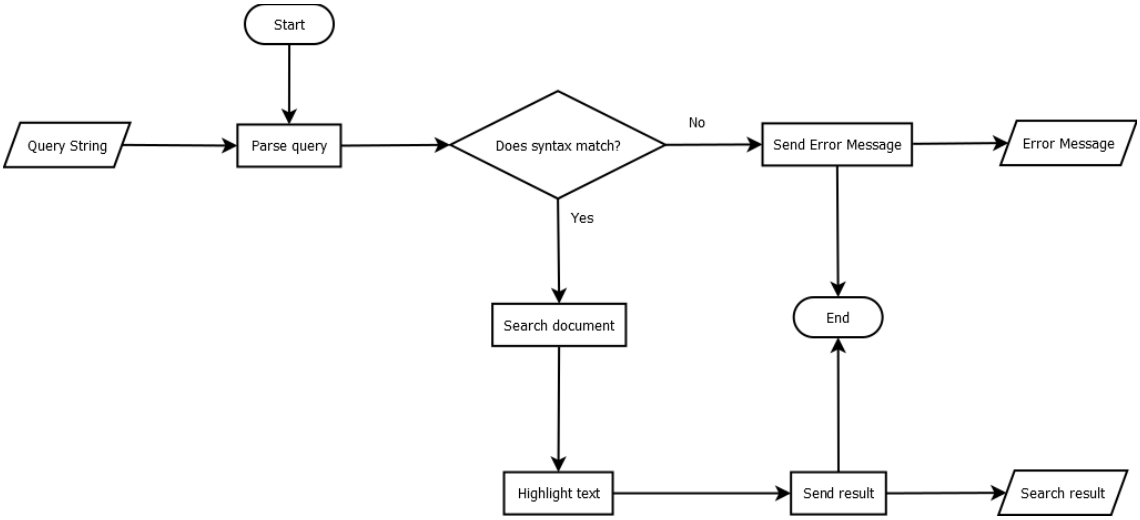


fig. 6 Flux del procés de peticions

El procés de peticions comença quan l'usuari introdueix una cerca en l'aplicació – en el cas del present treball, la demostració web. Aquesta cerca és enviada al processador de peticions, que es composa de:

- *Parser* (o analitzador): encarregat de verificar que la sintaxis de la petició sigui correcta i de construir l'objecte de petició que serà enviat al buscador.
- Buscador: encarregat de extreure els documents de l'índex segons els paràmetres de la l'objecte generat a partir de la petició.
- *Highlighter* (o ressaltador): encarregat de ressaltar les paraules cercades dins del contingut del document i extreure'n aquelles parts més rellevants per l'usuari.

El flux de la petició segueix el mateix ordre que la llista anterior; la petició és analitzada pel *parser*, en cas que la sintaxis no sigui correcte es genera un missatge d'error explicant amb una breu explicació de què ha fallat, si és correcte l'objecte de petició generat és enviat al buscador, qui n'extreu els documents de l'índex. Aquests documents (si n'hi ha) són enviats al *highlighter* aportant al resultat aquelles parts més rellevants del document que contenen les paraules claus introduïdes per l'usuari; el resultat és enviat a l'aplicació de nou.

És important entendre que la interfície d'usuari té la responsabilitat de decidir com aquesta informació és mostrada i no del mòdul de peticions.

4. Tecnologies

En aquest apartat es descriuen les tecnologies utilitzades durant el projecte (tant en l'entorn de treball com durant la codificació). Per entendre la relació entre aquestes tecnologies i la fase desenvolupament cal consultar l'apartat *5. Implementació*.

4.1. Tecnologies en l'entorn de treball

Entenem per entorn de treball totes aquelles tecnologies que ajuden al programador a desenvolupar les seves tasques. El entorn de treball que hem ideat per aquest projecte es compon de quatre parts:

- Entorn de desenvolupament integrat.
- Entorn de control de codi (o *Sistema de control de versions*).
- Entorn de compilació i desplegament (o *Sistema de compilació i gestió del projecte*).
- Entorn de gestió de qualitat del codi.

4.1.1. Entorn de desenvolupament integrat

L'entorn de desenvolupament integrat (o *IDE – Integrated development environment*) és l'aplicació on el programador desenvolupa les tasques habituals de codificació en la seva màquina local. En el context d'aquest projecte s'ha intentat que les dependència entre totes les parts relacionades amb l'entorn de treball (desplegament de l'aplicació, control de versions, compilació i anàlisis de la qualitat del codi) estiguin totalment desacoblades del IDE utilitzat. De manera que el desenvolupador es pugui sentir còmode utilitzant una u altre eina com a ell més li convingui per continuar mantenint el codi de l'aplicació i desenvolupar noves característiques.

Tot i així, el IDE preferit durant la fase de desenvolupament ha estat *Eclipse*[2]. Aquesta elecció es deu a la utilització de *JavaCC* per *Eclipse* denota una millor integració que les demés alternatives de codi obert (en aquest cas: *Netbeans*).

4.1.2. Sistema de control de versions

El sistema de control versions empleat ha estat majoritàriament manual. Tot i així s'ha desplegat un servidor Subversion[3] (*SVN* d'ara endavant) en un servidor local per la comoditat de desenvolupar el codi des de diferents terminals.

[2] http://www.eclipse.org/
[3] http://subversion.apache.org/

4.1.3. Sistema de compilació i gestió del projecte

Per tal de gestionar les dependencies i la compilació del projecte s'ha optat per un sistema de compilació automàtic. Com s'ha enunciat en els punts anteriors el sistema escollit ha estat *Maven*[4].

Les avantatges d'un sistema d'aquest tipus sobre sistemes tradicionals són:

- Gestió automàtica de dependències.
- Organització automàtica dels projectes en jerarquia.
- Estandardització dels processos de compilació i desplegament.
- Agilització del procés de desenvolupament.
- Convencions enlloc de configuració.
- Independent del IDE, no cal compartir fitxers de configuració del projecte.

Tot això permet integrar molt fàcilment terceres aplicacions que aporten un valor afegit al projecte: *Jenkins*[5], *SonarSource*[6], *JRebel*[7], *Artifactory*[8], etc. En el context d'aquest projecte ens fixarem en *SonarSource*.

4.1.4. Sistema per a la gestió de la qualitat de software

Durant la fase de desenvolupament s'ha introduït el concepte *qualitat del software*. L'objectiu és bàsicament desenvolupar l'aplicació sota convencions i patrons de disseny que permetin: desacoblar la lògica, complir amb un estil de codi estandarditzat, seguiment de la cobertura de codi de les proves unitàries, seguiment de la cobertura de la documentació en el codi, detecció a priori d'errors potencials, detecció de codi duplicat i mantenir una arquitectura coherent.

Tots aquests seguiments són realitzats per Sonar. Sonar és un sistema per a la gestió de qualitat del software que s'integra molt fàcilment amb el procés de desenvolupament descrit anteriorment.

L'objectiu ha estat que, cada vegada que es desenvolupin nous mòduls del projecte, el codi sigui desplegat en un servidor Sonar que analitza tots els punts anteriors a través de diferents mètrics: regles d'estilització del codi, percentatge de cobertura de codi i documentació, interdependències entre paquets del projecte, falta cohesió de les classes – LCOM4[9] –, etc. Tot això permet fer un seguiment estricte del codi implementat i solucionar problemes potencials abans de que sorgeixin .

[4] http://maven.apache.org/
[5] http://jenkins-ci.org/
[6] http://www.sonarsource.com/
[7] http://zeroturnaround.com/software/jrebel/
[8] http://www.jfrog.com/products.php
[9] http://www.aivosto.com/project/help/pm-oo-cohesion.html

4.2. Tecnologies en la implementació de l'aplicació

Les tecnologies en la implementació de l'aplicació són totes aquelles tecnologies utilitzades directament en el codi de l'aplicació. En el cas que ens ocupa això inclou tant el mateix llenguatge, utilitzat per codificar l'aplicació, com les llibreries i eines que aporten funcionalitat al codi.

Pel que respecte a les llibreries em dedicat un apartat únicament a aquelles que, pel valor que aporten a l'aplicació, requereixen d'un coneixement més profund. La resta de llibreries es troben enumerades en el punt *4.2.8. Miscel·lània* amb una breu descripció de la seva funcionalitat.

4.2.1. Java

En els requeriments del projecte, i a nivell de l'elecció de llenguatge de programació, es presenten dos grans dificultats (íntimament relacionades amb les dependències del projecte). Aquestes les hem vistes en l'apartat *2.4* del projecte de viabilitat (*Alternatives i selecció de la* solució). Per un costat la llibreria de IR i per l'altre el motor PLN, les raons per les que s'ha escollit una i altre solució estan exposades allà i no és necessari repetir-les, tot i així la elecció per la que s'ha optat presenta una lleugera incompatibilitat: que un (Lucene) esta escrit Java i l'altre (FreeLing) esta escrit en C. Tenint en compte això quedaven dos vies per escollir el llenguatge: C o Java. (altres llenguatges són també possibles arribats a aquest punt, però en aquests cassos el cost d'adaptació hagués estat molt més gran tot i que no infranquejable).

Tant FreeLing com Lucene proporcionen API en diferents llenguatges: FreeLing ofereix un API Java a través de JNI i Lucene diferents ports a llenguatges. A més a més, FreeLing pot utilitzar com a servei. Tenint en compte que tant un com l'altre s'adapten amb certa facilitat en ambdós llenguatges l'elecció es converteix en una qüestió de prioritats: Quina plataforma ofereix les eines que tenen un cost d'adaptació més baix pel tipus de projecte que volem desenvolupar? La resposta és Java. Les eines a les que ens referim són:

- JNI[10] (*Java native interface*): A través de JNI una aplicació que s'executa en la màquina virtual de Java és capaç de interactuar amb programes que no estan compilats per la màquina virtual. Aquesta eina és particularment útil a l'hora integrar FreeLing en l'aplicació.
- JUnit[11]: Framework de proves unitàries, permet avaluar si el codi escrit es comporta de la manera desitjada o no. Es pot integrar molt fàcilment amb algunes de les eines que ja hem vist, com per exemple Mavan o SVN, per tal d'evitar (en la mesura del possible) publicar codi o desplegar una aplicació que no passa les proves unitàries.
- Tomcat[12]: Tomcat és un servidor web i contenidor de servlets per aplicacions escrites en Java.

[10] http://java.sun.com/docs/books/jni/
[11] http://www.junit.org/
[12] http://tomcat.apache.org/

Existeixen 5 eines més escrites en Java que estan íntimament relacionades amb el projecte: Apache Wicket, Apache Tika, Mockito, Googl-Guice i JavaCC. Per a entendre que aporten cal referir-se als seus apartats corresponents.

4.2.1. JavaCC[13]

JavaCC és un generador d'analitzadors sintàctics en notació EBNF (notació que permet expressar gramàtiques lliures de context). Aquesta tecnologia ens permet generar un interpretador d'un llenguatge – en el nostre cas, per a un llenguatge de peticions.

JavaCC genera codi Java a partir d'un codi escrit en una sintaxis pròpia. S'ha de tenir en compte que el aquest codi no ha de ser analitzat pel nostra sistema de gestió per la qualitat del software, això es pot aconseguir fàcilment a través del model Maven del nostra projecte. La generació del codi es pot duu a terme en temps de compilació també a través d'aquest mateix model – facilitant-ne la integració.

4.2.2. Google Guice[14]

En els 5 principis definits per Robert C. Martin de la programació orientada objecte i el disseny (*SOLID*) un dels més importants és la inversió de dependències, la noció de que tots el mòduls d'una aplicació han de dependre d'abstraccions. Una de les maneres d'acomplir aquest propòsit és mitjançant un framework d'injecció de dependències;. El més estès i utilitzat per aquest propòsit és *Spring*, tot i així en aquest projecte, i a mode d'experiment, hem volgut utilitzar un framework diferent, que fos més lleuger, adaptable i en el que tota la descripció de dependències i creació d'objectes quedés patent en el codi i no en un fitxer de configuració: Google-Guice (d'ara endavant Guice).

Guice no ofereix cap característica que no estigui inclosa en Spring ni tampoc l'eficiència que ofereix és significativament superior com perquè sigui un factor de decisió important. Escollir un o l'altre és una qüestió de preferències.

Guice esta totalment integrat en el codi i no permet cap mena de configuració externa pel que respecte al enllaç de dependències. Guice no introdueix grans canvis en el flux de la lògica, en comptes proporciona una gran quantitat d'anotacions i interfícies amb l'objectiu que el codi Java original estigui totalment desacoblat de la lògica d'injecció de dependències fins al punt de ni tant sols forçar cap tipus de norma de nomenclatura sobre el codi.

4.2.3. Mockito[15]

Mockito és una infraestructura per a la simulació d'objectes (*mock*) durant les proves unitàries. Simular objectes és crucial a l'hora de testejar la funcionalitat d'una classe sense que les implementacions concretes de les seves dependències interfereixin en el resultat del test.

[13] http://javacc.java.net/
[14] http://code.google.com/p/google-guice/
[15] http://code.google.com/p/mockito/

Existeixen altres infraestructures que utilitzen aquest paradigma, tot i així de totes elles Mockito és la única que obliga a pensar el codi des del precís moment en que es comença a implementar per tal de que no sigui difícil de desenvolupar-ne les proves (reforça el bon disseny de l'aplicació).

Mockito ofereix principalment simplicitat, tant a nivell de sintaxis com d'implementació:

- No conté cap mena de llenguatge de simulació.
- No hi ha implementació de classes anònimes.
- No implementa un context de test com altres infraestructures.
- API simple: Si és difícil desenvolupar un test és que la classe està mal dissenyada.

4.2.4. Lucene[16]

Lucence és una llibreria de recuperació d'informació de codi obert. Lucene permet indexar qualsevol tipus de document sempre i quan sigui possible extreure'n text (per exemple, no és capaç d'indexar imatges). Tot i així, Lucene en sí mateix no ofereix cap manera d'extreure aquest contingut des de fitxer en diferents formats (veure *4.2.5 Apache Tika* per veure la solució per la que s'ha optat en aquest problema).

Tot el que ofereix Lucene és indexació de text i extracció de documents a través de peticions, no ofereix replicació ni distribució de índexs (tot i que si ofereix un munt d'eines per facilitar-ne la tasca). Degut a la naturalesa de l'aplicació la replicació i distribució d'índexs no són característiques requerides, ja que l'únic índex generat coexisteix en la mateixa màquina on es troben els documents.

Pel que fa la recuperació de informació, Lucene ofereix un sistema de peticions complet i fins hi tot en proporciona un llenguatge, tot i que aquest no exprimeix tot el potencial del sistema. En aquest projecte es modifica aquest llenguatge per tal de fer un ús complet del sistema que Lucene ofereix i per introduir-hi funcionalitat específica de l'aplicació.

Pel que fa a d'indexació de documents, Lucene també ofereix eines simples d'anàlisis de text: eliminació de stop words, tokenització, etc. Aquestes eines estan pensades per estendre'n la funcionalitat i per combinar-les amb eines específiques de l'aplicació que es vol desenvolupar. En l'aplicació que ens ocupa aquestes es fan servir en conjunt amb FreeLing.

4.2.5. Apache Tika[17]

Tika és una llibreria que ofereix detecció i extracció automàtica del contingut de documents en diferents formats. En un projecte d'aquestes característiques tenir una extensa compatibilitat en diferents formats de documents és de vital importància per tal de oferir una experiència completa a l'usuari.

[16] http://lucene.apache.org/core/
[17] http://tika.apache.org/

Tika reuneix les principals llibreries opensource per llegir el contingut de documents en una gran varietat de formats. A més a més d'oferir un interfície comuna d'accés a la funcionalitat de les diferents llibreries, ofereix extracció de la metada dels documents. Depenent del fitxer en concret aquests camps seran uns o altres – autors, companyia, data de creació, títol, subjecte, etc. Aquests camps són d'especial importància per tal d'afegir nous paràmetres per oferir filtres més específics en els resultats d'una cerca i diferenciació entre diferents verticals de cerca (en cas que es volgués implementar).

4.2.6. FreeLing[18]

FreeLing es una llibreria de processament del llenguatge natural escrita en C++ per la Universitat Politècnica de Catalunya. Aquesta llibreria suporta 9 llenguatges: Anglès, Català, Castellà, Portuguès, Italià, Rus, Gallec, Asturià i Gal·lès. A l'hora d'utilitzar aquests llenguatges existeixen certes limitacions. Com hem vist en apartats anteriors l'aplicació ha de ser capaç de detectar el idioma d'un document automàticament abans de que sigui processat, si algun idioma manca serà incapaç de processar el document amb FreeLing (en comptes ho farà amb un analitzador propi de Lucene).

FreeLing proporciona una interfície JNI força rudimentària (codi generat) i un servidor per *sockets* per tal de transmetre i analitzar dades. Algunes de les característiques que inclou són:

- Tokenització de text.
- Separació de sentències.
- Anàlisis morfològic.
- Tractament de sufixos.
- Retokenització de pronoms clítics.
- Reconeixement flexible de multi-paraules.
- Detecció probabilística de categories en paraules desconegudes.
- Anàlisis sintàctic.
- Reconeixement i classificació de noms de entitats.
- Detecció automàtica de dates, números, proporcions, monedes i magnituds físiques.
- Anàlisis gramatical.
- Anotació de significat i desambigüació basat en WordNet.
- Anàlisis de dependències en las paraules d'una frase.
- Resolució de coreferència nominal.

Les més interessants per el nostra projecte són la tokenització del text (necessari per indexar la posició de les paraules dins del contingut d'un document), l'anàlisi morfològic (per extreure l'arrel morfològica d'una paraula sense recórrer al stemming) i Anàlisis gramatical (per tal d'indexar la categoria gramatical de les paraules dins d'una frase). Tot i així altres característiques han de ser activades per tal de assolir el nivell de detall desitjat durant el anàlisis del text: Separació de sentències, tractament de sufixos, retokenització de pronoms clítics i detecció automàtica de dates, números, proporcions, monedes i magnituds físiques.

[18] http://nlp.lsi.upc.edu/freeling/

FreeLing esta present durant la indexació i les peticions al índex (ja que les peticions han de ser processades amb el mateix algoritme que tracta el contingut dels documents per tal de trobar resultats en l'índex).

4.2.7. Apache Wicket[19]

Per tal de realitzar la demostració gràfica de l'aplicació s'ha optat per realitzar una petita aplicació web. L'aplicació web en qüestió esta suportada y generada per un software (anomenat web framework) que s'encarrega alleujar la carga del programador a l'hora d'escriure aplicacions web dinàmiques: gestionar sessions, omplir plantilles HTML amb el contingut dinàmica, cache i generació de URLs entre altres avantatges.

Apache Wicket és un web framework simple, reutilitzable, no intrusiu, seguir i escalable. Hem escollit aquest web framework perquè és molt fàcil d'utilitzar i esta preparat per ser integrat amb Google-Guice i perquè els sistema de plantilles HTML esta completament desacoblat de la lògica de generació de contingut.

4.2.8. Bootstrap[20]

Bootstrap és un framework HTML, CSS i javascript que facilita el i disseny de pàgines web proporcionant gran quantitat d'elements reutilitzables que es poden introduir molt fàcilment amb l'objectiu de millorar l'usabilitat d'una aplicació web. Per tal d'utilitzar aquest framework és necessari disposar de jquery. Twitter proporciona aquest framework escrit en HTML5 amb una llicència de codi obert.

4.2.8. Miscel·lània

El projecte inclou altres dependències de menor importància que faciliten l'etapa de programació. Aquestes llibreries normalment estenen la funcionalitat pròpia de Java implementant solucions ben estudiades a problemes concrets i comuns en aplicacions Java, maximitzant la productivitat del desenvolupador. Tot i que no ser tant importants, no està de més enumerar-les a continuació:

- *commons-io*[21]: recursos per operacions entrada/sortida, estén la funcionalitat de *java.io*.
- *commons-cli*[22]*:* recursos per el anàlisis d'arguments escrit des d'un terminal.
- *commons-lang*[23]: recursos bàsics del llenguatge, que estenen la funcionalitat de *java.lang*.
- *commons-codec*[24]: recursos per la codificació i descodificació de dades.

[19] http://wicket.apache.org/
[20] http://twitter.github.com/bootstrap/
[21] http://commons.apache.org/io/
[22] http://commons.apache.org/cli/
[23] http://commons.apache.org/lang/
[24] http://commons.apache.org/codec/

4.3. Tecnologies adaptades

Si recapitulem al punt 2.3 d'aquest document: *Requisits del sistema*, el requisit funcional número 7 enunciava: <<*Detecció automàtica d'idiomes en els fitxers de entrada*>>. Existeixen diverses tecnologies capaces d'acomplir aquest requisit i que es podien haver integrat en l'aplicació, la més coneguda i utilitzada és l'API de Google per a la traducció de textos, aquesta esta limitada a un cert número de peticions diàries que un cop exhaurides requereixen de contractar el servei complet de Google. En altres llenguatges, com per exemple Python, existeixen eines de codi obert que proporcionen una funcionalitat similar. Tot i així en el context del nostre projecte es requeria d'una solució que fos: 1) Oberta, 2) Que si pogués accedir sense connexió a Internet i 3) Que estigués escrita en Java.

Després de una cerca intensiva no es va trobar cap tecnologia que solucionés aquest problema i complís els tres requisits aquí llistats. Finalment es va optar per adaptar la tecnologia existent i crear una eina que solucionés el problema tot complint els requisits anteriors. A aquesta llibreria se l'ha anomenat *JGram*

Considerat això, procedim a explicar-ne el seu funcionament:

4.3.1. JGram

El nom de la llibreria fa honor a la seva funcionalitat: JGram permet extreure i manipular n-grames d'una col·lecció de documents – o corpus. Aquesta tecnologia ens permet comparar textos contra el corpus mitjançant els n-grames extrets i determinar de forma molt precisa si les combinacions de caràcters que es troben en aquest text es corresponen a aquelles presents en el corpus original.

Què és un n-grama?

Un n-grama és una seqüència d'elements contigus de mida *n* per a una mostra de text. Per exemple: imaginem que volem extreu tots els 3-grames de la següent sentència: ***Brown fox*** a nivell de caràcter el resultat seria:

["Bro", "row", "own", "wn ", "w f", " fo", "fox"]

El resultat final pot variar depenent dels filtres aplicats (eliminació de caràcters irrellevants, conversió de les lletres a minúsula, etc.). Tots aquests filtres ens ajuden a reduir el número de possibles sortides sense perdre entropia i, per tant, a optimitzar la mida en memòria de l'estructura resultant.

Per tal d'optimitzar la velocitat de cerca dels n-grames aquests estan continguts en una estructura anomenada Trie. Aquesta estructura és també força adient perquè permet optimitzar l'espai utilitzat amb gran eficiència, tot i així la mateixa llibreria inclou eines per comprimir l'estructura tot eliminant aquelles branques de l'arbre que, per la poca quantitat d'entropia que contenen, no són rellevants. L'estructura en si mateixa també ha estat

desenvolupada tenint en compte limitacions d'espai (tal i com hem pogut veure fa un moment).

Què és un Trie?

Un Trie és una estructura en forma d'arbre que conté una sola lletre per node. Cada nivell de profunditat del Trie conté la següent lletra d'una seqüència de caràcters contigus. Cada fulla conté un pes que correspon el número de coincidències en el text original del n-grama que referències la branca del arbre.

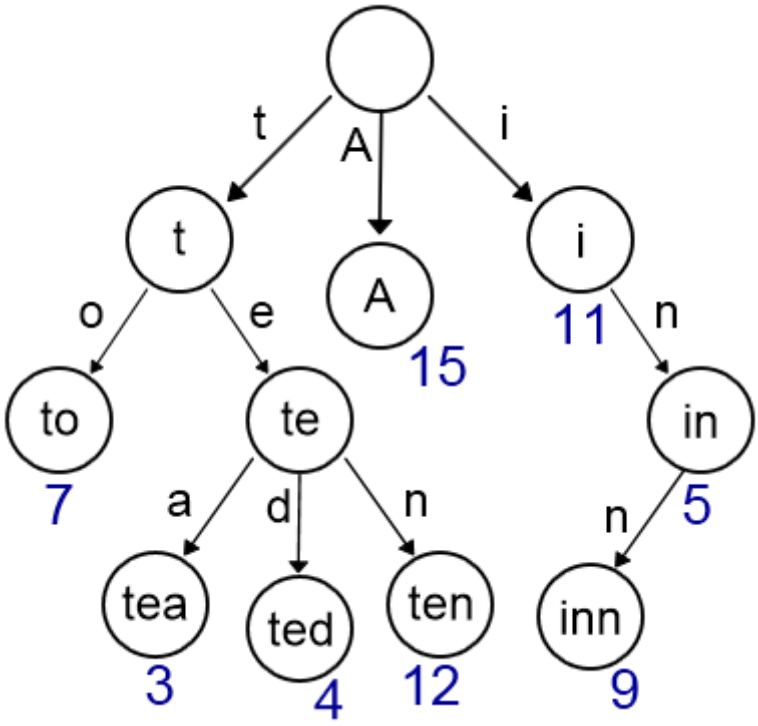


fig. 7 Diagrama d'un Trie[25]

Cada arbre, a més a més, conté la suma de tots els pesos de totes les fulles, de manera que es pugui calcular ràpidament la freqüència de qualsevol coincidència ràpidament.

Per tal detectar a quina classe correspon un text qualsevol es realitzen dos passos:

1. Es compara el text contra cada arbre per tal d'obtenir una puntuació (*score*), és a dir, quan semblant és el text a la classe que representa l'arbre. Per a cada n-grama (k) del text (*x*) es multiplica la freqüència (*freq*) del n-grama en el text contra la freqüència en el corpus original (*C*), la suma d'aquests valors és la puntuació que li atribueix a aquesta classe:

$$score(x, C) = \sum_{i=0}^{n} freq(k_i, x) * freq(k_i, C)$$

fig. 8 Fórmula de puntuació

[25] http://en.wikipedia.org/wiki/File:Trie_example.svg

2. Es calcula la probabilitat de que el text sigui un membre de la classe d'acord amb les observacions recollides en el punt 1):

$$P(x|C) = \frac{score(x, C)}{\sum_{i=0}^{n} score(x, C_i)}$$

fig. 9 Fórmula de probabilitat

Cóm pot ajudar aquesta tecnologia a detectar llenguatges? Un anàlisis de n-grames d'una col·lecció de corpus suficientment gran per a cada llenguatge, ens aporta suficient entropia com per decidir amb precisió a quin corpus es més afí un text concret. El text en si mateix també ha de contenir suficient entropia per tal minimitzar el número de falsos positius durant la detecció, així doncs una paraula o una frase no contenen prou informació com perquè en podem detectar el idioma, tot i així un paràgraf si. Comparar un text contra tots els arbres de n-grames una operació relativament poc costosa comparat amb el número d'operacions d'entrada / sortida que tenen lloc en el marc de l'aplicació (l'autèntic coll de botella), de fet la llibreria es prou eficient com per donar una resposta en una magnitud de ms.

Idiomes suportats

Per tal de generar l'arbre d'un llenguatge es necessari tenir un corpus suficientment gran per tal de que l'eficàcia de la detecció convergeixi fins a un punt acceptable. Els llenguatges suportats de moment són: Búlgar, Català, Txec, Danès, Alemany, Danès, Grec, Anglès, Español, Estonià, Finlandès, Francès, Hongarès, Italià, Lituà, Letó, Holandès, Polonès, Português, Romanès, Eslovac, Eslovè y Suec. Tot proporcionats per el *Europarl Parallel Corpus*[26]. El corups en català esta proporcionat per la Universitat Politècnica de Catalunya en una extracció de la Wikipedia de 2006[27].

Ampliar aquesta llista és només qüestió de trobar un corpus adequat pel llenguatge que es vol afegir.

[26] http://www.statmt.org/europarl/
[27] http://www.lsi.upc.edu/~nlp/wikicorpus/

5. Implementació

En aquest s'exposa l'execució final del projecte en relació al que s'ha pogut veure en els apartats *3. Anàlisi* i *4.Tecnologies*. Com es pot veure al llarg d'aquest apartat algunes de les característiques llistades en l'apartat *2. Estudi de Viabilitat*, han variat lleugerament, per entendre aquests canvis durant la fase d'implementació cal consultar les conclusions (apartat *6. Conclusions*).

5.1. Desenvolupament

Entenem per desenvolupament totes les metodologies associades en la resolució de tasques, és a dir, el procés que existeix entre la creació d'una tasca i la seva resolució. Aquest procés l'hem dividit en 5 fases que estan estretament relacionades amb les tecnologies enumerades a l'apartat *4.1. Tecnologies a l'entorn de treball*:

1. Codificació en l'entorn de desenvolupament integrat.
2. Publicació del codi en l'entorn de control de codi.
3. Compilació i desplegament de l'aplicació en un entorn de proves.
4. Un sistema de gestió de qualitat del codi verifica que els canvis són correctes.
5. Quan es considera que els canvis realitzats funcionen correctament i compleixen els requisits, l'aplicació es desplega en un entorn de producció.

Evidentment no totes les tasques dins del marc d'un projecte segueixen aquest esquema, moltes tasques de manteniment poden no estar directament relacionades amb el codi directament, en aquest cas el procés esta definit d'aquesta manera:

1. Modificació de l'entorn de proves per complir amb els nous requeriments.
2. Desplegament de l'última versió de l'aplicació en un entorn de proves.
3. Quan es considera que els canviïs realitzats funcionen correctament, desplegament de l'aplicació en un entorn de producció.

Com podem veure els punts 1, 2 i 4 no són necessaris ja que el codi (és a dir, la versió efectiva de l'aplicació) no ha canviat.

5.1.1. Codificació

En aquest apartat veurem en quins mòduls consisteix l'aplicació i com es comparen respecte al que hem pogut veure en l'apartat *3.2. Analisis del sistema*. El codi de l'aplicació esta dividit en 6 projectes, cada un d'ells té una responsabilitat concreta, i existeixen interdependències entre ells de la mateixa manera que amb les tecnologies vistes en l'apartat *4.2. Tecnologies en la implementació de l'aplicació*. Aquests projectes són:

5.1.1.1. JGram

Com ja hem pogut veure en l'apartat *4.3. Tecnologies adaptades*, aquest projecte és utilitzat com a motor de comparació de textos a través de n-grames per determinar estadísticament el idioma d'un document. Cal denotar que aquest projecte no és per si sol capaç de detectar idiomes ni esta entrenat en cap altre tasca, cal doncs entrenar la llibreria i generar els fitxer corresponents per tal d'utilitzar-lo.

Utilització

Per tal de generar arbres de coneixement és necessari compilar el projecte en un *.jar* executable amb totes les dependències (a través de la comanda *mvn assembly:assembly*). A continuació s'especifica el directori on es troba el corpus i el fitxer de sortida. Els paràmetres opcionals són:

- *Encoding* del fitxer (l'especificat pel sistema operatiu per defecte).
- Filtre de caràcters (espais, salts de línia i signes de puntuació són ignorats per defecte).
- Conversió de lletres en minúscules (desactivat per defecte).
- Factor de *prunning* de l'arbre (els n-grames amb menys de 100 repeticions són descartats per defecte).
- Mida dels n-grames (3 per defecte).

Implementació

Veure l'apartat *4.3. Tecnologies adaptades*.

Dependències:

Externes:

- **GroupId:** org.apache.commons **Artifcat:** commons-lang3 **Version:** 3.0.
- **GroupId:** org.apache.commons **Artifact:** commons-io **Version:** 2.0.1.
- **GroupId:** org.apache.commons **Artifact:** commons-codec **Version:** 1.5.
- **GroupId:** org.apache.commons **Artifact:** commons-cli **Version:** 1.2.
- **GroupId:** log4j **Artifact:** log4j **Version:** 1.2.16.

Test:

- **GroupId:** org.mockito **Artifact:** mockito-all **Version:** 1.8.5.
- **GroupId:** junit **Artifact:** junit **Version:** 4.8.2.

Desplegament

Es pot publicar com a dependència o es pot compilar en un *.jar* amb totes les dependències per tal d'entrenar-lo amb corpus específics i generar-ne els arbres de coneixement.

5.1.1.2. Morfo

En un principi aquest projecte decorava la implementació JNI de FreeLing per tal de realitzar operacions bàsiques, tot i així s'ha considerat que el baix rendiment d'aquesta opció genera un coll de botella important en el flux de l'aplicació. A més a més, la compilació de llibreries JNI trenca el principi d'independència que s'intenta assolir quan es programa una aplicació en Java, fent-ne més complicats els cicles de compilació i distribució (especialment si s'utilitza Maven). Finalment s'ha optat per implementar aquest projecte un servei client-servidor on s'accedeix a FreeLing com a un servei remot a través de un Socket sota el protocol TCP. D'aquesta manera s'exclou el manteniment de la llibreria JNI de FreeLing i les dependències que aquesta genera i s'obté un major rendiment durant l'anàlisi morfosintàctic i l'extracció de lemes.

Implementació

Conté un únic mòdul anomenat *MorfoAnalyzer*. No és necessari que el servidor de FreeLing estigui instal·lat en la mateixa màquina on es vol utilitzar aquest mòdul, tot i que es pot configurar perquè així sigui (aquesta es la opció que s'ha escollit en la demostració web desenvolupada en aquest projecte). L'únic mètode del mòdul (*analyze*) retorna una llista ordenada dels elements extrets del text que es proveeix per paràmetre, cada element conté la paraula original analitzada, el seu lema i una cadena que representa l'anàlisi POS dins de la sentència (la categoria gramatical de la paraula en detall).

L'analitzador s'encarrega de reconstruir l'ordre original de las paraules i de calcular la seva posició en la sentència original – aquesta part és particularment delicada, ja que més endavant aquestes posicions s'utilitzen per destacar paraules en els fragments d'un document, així doncs existeix una lògica dedicada a detectar i corregir errades per minimitzar l'impacta de falsos positius.

Utilització

El projecte permet realitzar anàlisis de sentències a través de la consola de comandes. Només cal especificar la sentència i l'idioma. És necessari que el servei de FreeLing de destí estigui en execució.

Dependències:

Test:

- **GroupId:** junit **Artifact:** junit **Version:** 4.8.2.

Desplegament

Es pot publicar com a dependència o es pot compilar en un *.jar* amb totes les dependències per tal de analitzar sentències específiques.

5.1.1.3. Nuuie-Aggregatoer

Aquest projecte es tracta d'una agregació de 4 projectes diferents (*nuuie-common*, *nuuie-query*, *nuuie* i *nuuie-web*). Conté un únic fitxer de configuració Maven on es declaren tots els mòduls i les versions de les seves dependències. Aquesta metodologia permet, no només introduir el concepte d'herència en el cicle de compilació i desplegament del software (que ajuda a evitar conflictes de versions entre mòduls amb gran interdependència), sinó que a més a més construeix un entorn de treball en el que és necessari que tots els mòduls estiguin lligats a un únic cicle, facilitant la gestió de versions i els cicles de vida de cada mòdul sense que s'interfereixi en el desenvolupament de la resta.

Els dos projectes citats fins ara (aquests són: *JGram* i *Morfo*) no col·laboren en el flux de dades de l'aplicació que ens ocupa, i per tant, el seu cicle de desenvolupament és independent del dels projecte que col·laboren en aquest agregador (que són, essencialment, els projectes que defineix el flux de dades tal i com s'ha vist en l'apartat *3.3. Diagrames de flux* i l'arquitectura tal i com s'ha vist en l'apartat *3.2. Diagrama del sistema*).

5.1.1.4. Nuuie-Common

Aquest projecte conté totes les utilitats que són comunes en l'agregador**,** s'ha creat així per tal d'evitar dependències cícliques entre els mòduls de l'agregador.

Utilització

Aquest projecte, a diferència dels seus germans, no és executable.

Implementació

Aquest projecte defineix totes les variables d'entorn del agregador:

- Llenguatges suportats.
- Localització de l'índex.
- Analitzador suportats.
- Mòduls de configuració
- Camps continguts en l'índex.
- Classificació de formats de fitxers.

A més a més conté totes aquelles utilitats que depenen de la configuració d'entorn del projecte, això inclou:

- *FreeLingAnalyzer*: Que defineix una política d'extracció de tokens basada en FreeLing (veure *5.1.1.2 Morfo*).
- *FreeLingTokenizer*: Enumera la seqüència de tokens que s'indexaran utilitzant un *MorfoAnalyze* (definit també en el modul *Morfo*).
- *JGramLanguageDetector*: Detector de llenguatges basat en *JGram* (veure l'apartat *5.1.1.1. JGram*).
- *NuuieModule*: Mòdul de configuració Guice, inicialitza les propietats contingudes en el fitxer de configuració juntament amb les dependències comunes de l'agregador.

Cal tenir en compte que no tots aquests idiomes enumerats en aquest mòdul són suportats per FreeLing i per tant el mètode d'anàlisis canvia segons el idioma (d'aquesta distinció se'n encarrega tant el procés d'indexació com el procés de peticions). La lematització utilitzant FreeLing només s'aplica si la configuració del servidor es troba en el fitxer de configuració de l'aplicació, sinó, s'analitza el text utilitzant un analitzador per defecte, o bé un proporcionat per la dependència *lucene-analyzers* o, si no n'existeix cap en concret per l'idioma, el *StandardAnalyzer* proporcionat per Lucene.

Dependències

Internes:

- **GroupId:** com.uab.morfo **Artifact:** morfo **Version:** 1.0.
- **GroupId:** com.uab.jgram **Artifact:** jgram **Version:** 1.0.

Externes:

- **GroupId:** com.google.inject **Artifact:** guice **Version:** 3.0.
- **GroupId:** com.google.inject.extensions **Artifact:** guice-multibindings **Version:** 3.0.
- **GroupId:** org.apache.lucene **Artifact:** lucene-core **Version:** 3.6.0.
- **GroupId:** org.apache.lucene **Artifact:** lucene-analyzers **Version:** 3.6.0.
- **GroupId:** log4j **Artifact:** log4j **Version:** 1.2.16.

Test:

- **GroupId:** org.mockito **Artifact:** mockito-all **Version:** 1.8.5.
- **GroupId:** junit **Artifact:** junit **Version:** 4.8.2.

Desplegament

Només es pot publicar com a dependència.

5.1.1.5. Nuuie

Aquest és essencialment el cor del backend de l'aplicació. La seva responsabilitat és la de col·leccionar i indexar correctament els documents. Veure el diagrama a *3.3.1. Flux del procés de indexació*, per entendre en detall com s'executen les tasques d'aquest projecte en diferents situacions.

Utilització

Pot ser executat a través de la consola de comandes (útil per provar el mòdul en un entorn d'implementació), com a part d'un mòdul més gran o com a servei dins d'un servidor. En el punt de desenvolupament actual, executar el mòdul en un servidor no és recomanable ja que no és possible enviar comandament a través d'un servei web per tal de afectar el procés, és per això que a mode de demostració de l'aplicació s'ha optat per integrar-lo directament en la demostració web (com a dependència).

Implementació

El projecte es compon de dos mòduls principals que controlen el flux de l'indexació, per una banda el *Collector* es dedica a extreure el contingut dels documents d'una font (*Source*) i preparar-los per a que puguin ser indexats. Per l'altra, el *Indexer* es dedica a rebre aquests documents i indexar-los. Ambdós mòduls s'executen en fils d'execució diferents i es comuniquen a través d'una cua on el *Collector* bolca els documents que el *Indexer* ha de processar. Quan el *Collector* acaba, afegeix un element contaminat (estrategia coneguda com a *Poison Pill*) per assenyalar al *Indexer* que ha d'acabar l'execució.

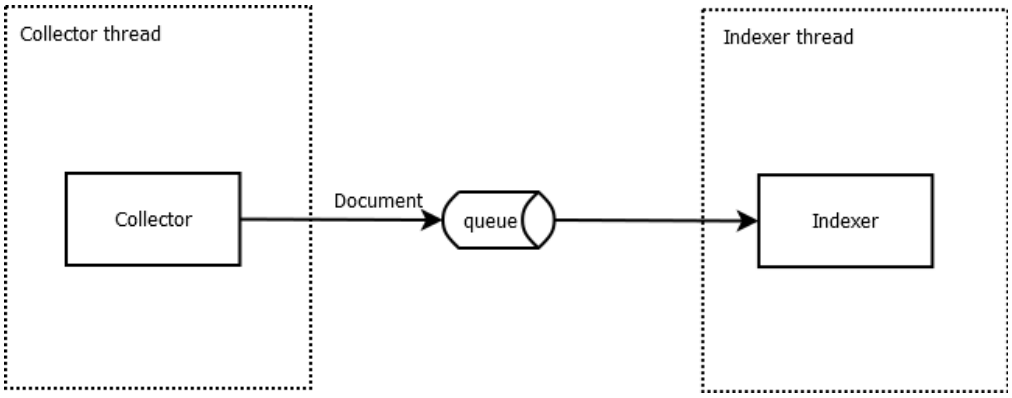


fig. 10 Diagrama de transaccions entre el *Collector* i el *Indexer*.

Un dels altres mòduls més importants és el *IndexManager* que s'encarrega de mantenir la coherència en l'índex per a totes aquelles entrades que troba el *Collector*. Aquest mòdul existeix com a dependència del *Collector*. L'eliminació de documents corre a part del *Renewer*, que inspecciona tots els documents de l'índex, prèviament a iniciar el procés d'indexació, i determina si continuen existint en la font original o no.

La detecció de duplicats s'aconsegueix mitjançant la creació d'un hash a través d'un algoritme de codificació SHA-512 utilitzant tot el contingut del fitxer original. La detecció de idiomes es realitza a través de JGram (veure apartat *5.1.1.1*).

La inspecció de documents només es pot duu a terme en un entorn local mitjançant un explorador recursiu del sistema de fitxers en el que se l'hi poden definir punts d'entrada. Tot i així implementar nous model de descobriment de fitxers o combinar-los és força senzill degut al disseny de l'aplicació, ja que es defineixen a través d'un *Strategy Pattern*. Un *Collector* pot acceptar un número il·limitat de fonts amb diferents estratègies d'inspecció.

Configuració

El fitxer de configuració permet modificar els següents paràmetres:

- Número màxim d'elements a la cua (*BlockingQueue.size*).
- Temps d'espera en ms per a la inserció d'elements a la cua (*Collector.timeout*).
- Temps d'espera en ms per a la recepció d'elements a la cua (*Indexer.timeout*).
- Número de caràcters utilitzats com a mostra per a la detecció de l'idioma *IndexManagerImpl.sampleChars*).

La localització de l'índex ve donada per la variable d'entorn *NUUIE_PATH*.

Dependències

Internes:

- **GroupId:** com.uab.morfo **Artifact:** morfo **Version:** 1.0.

Externes:

- **GroupId:** com.google.inject **Artifact:** guice **Version:** 3.0.
- **GroupId:** org.apache.commons **Artifcat:** commons-lang3 **Version:** 3.0.
- **GroupId:** org.apache.lucene **Artifact: l**ucene-core **Version:** 3.6.0.
- **GroupId:** org.apache.tika **Artifact:** tika-core **Version:** 1.1.
- **GroupId:** org.apache.tika **Artifact:** tika-parsers **Version:** 1.1.
- **GroupId:** org.apache.commons **Artifact:** commons-io **Version:** 2.0.1.
- **GroupId:** org.apache.commons **Artifact:** commons-codec **Version:** 1.5.
- **GroupId:** log4j **Artifact:** log4j **Version:** 1.2.16.

Test:

- **GroupId:** org.mockito **Artifact:** mockito-all **Version:** 1.8.5.
- **GroupId:** junit **Artifact:** junit **Version:** 4.8.2.

Desplegament

Es pot publicar com a dependència, compilar com un jar amb totes les dependències (per utilitzar-lo a través d'un terminal).

5.1.1.6. Nuuie-Query

Aquest projecte encapsula totes les classes necessàries per tal de realitzar peticions de cerca sobre l'índex. Veure el diagrama de *3.3.2. Flux del procés de peticions*, per entendre en detall com s'executen les tasques d'aquest projecte en diferents situacions.

Utilització

Pot ser executat a través de la consola de comandes o com a part d'un mòdul més gran. A mode de demostració s'ha integrat directament (és a dir, com a dependència) a la pàgina web.

Implementació

El projecte conté un mòdul principal que s'encarrega de transformar sentències de cerca en resultats. Integra un analitzador gramatical generat amb JavaCC que accepta sentències que compleixin la següent gramàtica (denotada en JavaCC):

```
QUERY := ( TERMS() | ("+" | "-")? SENTENCE() )+ < EOF >
SENTENCE := "\"" < WORD > + "\""
TERMS := (  ( "+" | "-" )? < WORD > ( "*" )? )+
```

El tipus < WORD > accepta qualsevol tipus de caràcter que no sigui un símbol de la gramàtica. Tot i que aquesta gramàtica és força simple el procés de descomposició de les paraules en lemes i termes és força complex i es duu a terme una sola vegada (quan totes les paraules han estat recollides). En les sentències no s'utilitzen els lemes sinó les paraules originals, ja que l'objectiu d'una sentència és cercar coincidències exactes en l'ordre que l'usuari ha escrit les paraules.

Aquestes sentències són transformades a un objecte *Query* (natiu de Lucene) que retorna els documents. Es poden aplicar filtres de diversos tipus en el rang de documents de la cerca, aquests filtres són proveïts per diferents classes que implementen el patró de disseny conegut com a *Factory*, en la present versió només es poden filtrar els documents per un rang de dates que corresponen a la data d'última modificació dels documents (es recomana la utilització de filtres per tal de no modificar l'ordre natural de rellevància dels documents en una cerca i perquè Lucene gestiona una cache interna per cert tipus de filtre, si s'apliquen correctament, que redueix l'impacta enormement en temps de processament de les peticions).

El número de documents cercats es pot limitar per tal de paginar el resultat. Només d'aquells documents dins del límit de la paginació se'n extreure'n tots els camps amb el format correcte (data de modificació, mostra del contingut rellevant, direcció relativa en el sistema de fitxers en format URL, etc.) que són susceptibles de ser utilitzats en l'aplicació final.

Sobre les peticions rebudes no s'executa cap detecció automàtica del llenguatge (s'assumeix que el text és massa curt per tal de detectar el llenguatge) així que per obtenir resultats més rellevants, s'ha d'especificar el llenguatge d'entrada manualment.

Configuració

En el fitxer de configuració les següents variables corresponen a aquest mòdul:

- *FragmentExtractor.fragmentSize*: Nombre de caràcters de cada mostra del contingut.
- *FragmentExtractor.maxFragments:* Nombre de mostres extretes per a cada document.

Dependències

Internes:

- **GroupId:** com.uab.nuuie **Artifact:** nuuie-common **Version:** 1.0.

Externes:

- **GroupId:** com.google.inject **Artifact:** guice **Version:** 3.0.
- **GroupId:** com.google.inject.extensions **Artifact:** guice-multibindings **Version:** 3.0.
- **GroupId:** org.apache.commons **Artifcat:** commons-lang3 **Version:** 3.0.
- **GroupId:** org.apache.lucene **Artifact:** lucene-core **Version:** 3.6.0.
- **GroupId:** org.apache.lucene **Artifact:** lucene-higlighter **Version:** 3.6.0.
- **GroupId:** org.apache.commons **Artifact:** commons-io **Version:** 2.0.1.
- **GroupId:** org.apache.commons **Artifact:** commons-codec **Version:** 1.5.
- **GroupId:** log4j **Artifact:** log4j **Version:** 1.2.16.

Test:

- **GroupId:** org.mockito **Artifact:** mockito-all **Version:** 1.8.5.
- **GroupId:** junit **Artifact:** junit **Version:** 4.8.2.

Desplegament

Es pot publicar com a dependència, compilar com un jar amb totes les dependències (per utilitzar-lo a través d'un terminal).

5.1.1.7. Nuuie-Web

Aquest projecte compren tots els mòduls relacionats amb la demostració web del projecte fent ús principalment de Apache Wicket com a framework per a la creació de l'aplicació web i de Bootstrap per generar la interfície d'usuari.

Utilització

El WAR generat en aquest projecte es desplega en un servidor servlet (en el nostra cas tomcat) des del que s'accedeix a l'aplicació web. També és possible debugar l'aplicació a través d'un servidor embedded (*Jetty*) directament des del IDE o través de la consola de comandes.

Pel que fa a l'a utilització de l'aplicació web en sí veure l'apartat *5.1.3. Demostració web*.

Implementació

El projecte esta dividit en 3 seccions principals: *Page* (pàgines)*, Panels* (panells) i *Markups* (components). Cada pàgina és una vista de l'aplicació, ja sigui la pàgina d'inici o la pàgina de resultats, aquestes es componen tant de panells com components. Cada Panell és una secció reutilitzable en el marc d'una pàgina, per exemple, la barra de cerca és un panell ja que s'utilitza tant en la pàgina principal com en la pàgina de resultats, un panell només conté components. Un component és la unitat mínima en una pàgina, inclou qualsevol tipus d'element: un enllaç, un formulari, un camp de text dins del formulari, etc. Poden estar compostos de més components o no. Els components es comporten com a controladors, responen a les entrades de l'usuari.

Existeix un únic mòdul que inicialitzar l'aplicació web denotat com a *WebApplication,* que s'encarrega de configurar tots els paràmetres necessaris, mapejar les urls i injectar totes les dependències a través d'un framework d'inversió de control extern (en el nostre cas, Google-Guice). A més a més s'encarrega de planificar el procés d'indexació (veure apartat *5.1.1.5 Nuuie*) a través de *quartz*.

Configuració:

- *ResultPage.numberOfResults*: Nombre de resultats per pàgina.
- *ResultPage.displayPages*: Nombre de pàgines mostrades en el panell de paginació.
- *Nuuie.sources*: Llista de directoris que es volen inspeccionar en el procés d'indexació.
- *JobActivator.cronExpression*: Expressió cron[28] que denota la planificació del procés d'indexació.

[28] http://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm

Dependències:

Internes:

- **GroupId:** com.uab.nuuie **Artifact:** nuuie-query **Version:** 1.0.
- **GroupId:** com.uab.nuuie **Artifact:** nuuie **Version:** 1.0.
- **GroupId:** com.uab.nuuie **Artifact:** nuuie-common **Version:** 1.0.

Externes:

- **GroupId:** com.apache.wicket **Artifact:** wicket-core **Version:** 1.5-RC7.
- **GroupId:** com.apache.wicket **Artifact:** wicket-guice **Version:** 1.5-RC7.
- **GroupId:** com.apache.wicket **Artifact:** wicket-extensions **Version:** 1.5-RC7.
- **GroupId:** com.google.inject **Artifact:** guice **Version:** 3.0.
- **GroupId:** log4j **Artifact:** log4j **Version:** 1.2.16.
- **GroupId:** org.slf4j **Artifact:** slf4j-log4j12 **Version:** 1.6.1.
- **GroupId:** org.quartz-scheduler **Artifact:** quartz **Version:** 2.1.5.

Test:

- **GroupId:** org.mockito **Artifact:** mockito-all **Version:** 1.8.5.
- **GroupId:** junit **Artifact:** junit **Version:** 4.8.2.

Desplegament

Com a war, es pot publicar en el repositori de Maven local o desplegar en un servidor tomcat automàticament utilitzant *maven-tomcat-plugin*.

5.1.2. Publicació del codi en l'entorn de control de codi

Quan es disposa d'un entorn de control de codi és important definir dues coses:

1. Quina és l'esquema de fitxers utilitzat.
2. Quina és l'estratègia de ramificació de versions.

En el nostra cas aquests dos punts estan estretament relacions amb el sistema de compilació, ja que Maven ofereix eines que, donat un esquema de fitxers, podem automatitzar la publicació de versions i la creació de branques i etiquetes en el l'entorn de control de codi.

5.1.2.1. Numeració de versions

Cada versió esta denotada per tres números, per exemple: 1.0.2. El primer número denota versions incompatibles entre elles, el segon número són versions que introdueixen canvis substancials al codi o noves característiques, però segueixen essent compatibles entre elles, l'últim número són publicacions de manteniment. Finalment s'afegeix -SNAPSHOT al final d'una versió si esta en desenvolupament, sinó significa que la versió és definitiva.

5.1.2.2. Esquema de directoris en el dipòsit de codi

El sistema de fitxer utilitzat segueix la següent estructura, d'acord amb la disposició recomenada del SVN book[29]: Cada projecte conté 3 directoris diferents (*trunk*, *branches* i *tags*). La següent imatge il·lustra l'estat del projecte d'agregació en un moment concret, on es poden veure els diferents directoris i versions en desenvolupament.

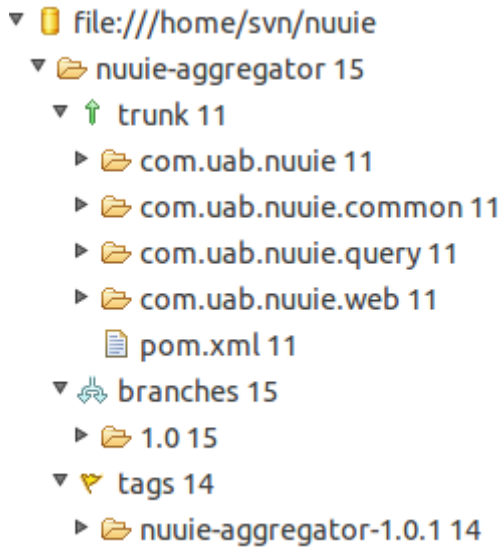


fig. 11 Estructura en el dipòsit de codi.

Trunk

El trunk conté el codi de la propera versió en desenvolupament. És obligat replicar els canviïs d'una branca en el trunk quan la primera és promoguda a una versió estable (aquesta acció rep el nom de *merge*, ja que el codi resultant ha de ser consistent tant amb l'esta de la branca com amb l'estat del trunk, sense que aquests canviïs afectin als requeriments de la pròxima versió en desenvolupament).

Si l'última versió que ha estat ramificada ha sigut la 1.1, aleshores el trunk conté la versió 1.2-SNAPSHOT. Tot i així el trunk pot contenir salts més grans si es creu necessari (és a dir canviïs incompatibles amb l'anterior versió, seria el cas de la versió 2.0-SNAPSHOT).

Branches

Cada branch conté el codi en desenvolupament de la següent versió del branch, és a dir, del manteniment d'aquesta versió. Per exemple: Si l'última publicació del branch 1.0 ha estat 1.0.1, la versió ara continguda en el branch 1.0 és 1.0.2-SNAPSHOT. Cada ramificació és creada automàticament per Maven.

[29] http://svnbook.red-bean.com/en/1.6/svn-book.html#svn.tour.importing.layout

Tags

Els tags contenen versions de codi fixes i no modificables, és a dir, una imatge del codi en el moment en el que va ser promogut. Aquests són generats per *Maven* en el moment de publicar (release) una versió.

5.1.2.3. Compilació i desplegament de l'aplicació

La compilació i el desplegament de l'aplicació estan també gestionats per *Maven*. Cada projecte compleix els requisits d'un arquetip diferent, de manera que *Maven* és capaç de decidir automàticament (a través dels passos definits en el fitxer de configuració) com compilar el codi i desplegar-ne els fitxers generats en l'entorn de desenvolupament. Cada paquet generat després d'un cicle de compilació és instal·lat en un dipòsit local des de on la resta de projectes són capaços de recuperar-ne el contingut, si és que en depenen. A més a més els mòduls són analitzats a través de *Sonar* per avaluar-ne la qualitat:

Name	Version	Lines of code	Rules compliance
Aggregator POM for Nuuie	1.0-SNAPSHOT	2,735	99.5%
jgram	1.0-SNAPSHOT	578	99.5%
morfo	1.0-SNAPSHOT	593	92.7%

fig. 12 Resum de la última compilació dels mòduls.

En la figura anterior es pot veure un resum del anàlisis dut a terme per Sonar dels tres mòduls del projecte (l'agregador es considera un sol mòdul a efectes de avaluar-ne la qualitat). Per a mètriques més detallades cal referir-se a la pàgina de detall de cada projecte.

Nuuie-Web es pot instal·lar, com a *war*. en un servidor *tomcat*, conté totes les dependències necessàries per tal de començar a indexar documents i realitzar cerques en l'aplicació web. Cal definir la variable d'entorn *NUUIE_PATH*, aquest directori ha de contenir el fitxer de configuració de l'aplicació (anomenat *nuuie.conf*) juntament amb un subdirectori (anomenat *index*), que contindrà l'índex de documents generat per *Lucene*. Finalment si es vol utilitzar *FreeLing* s'ha de procedir a la seva instal·lació i l'execució del servidor[30] (s'ha d'executar un servidor per llenguatge, ja que *FreeLing* no suporta múltiples llenguatges sota un mateix servidor). Per a informació més detallada veure el *README.txt* adjuntat en el codi del projecte.

[30] http://nlp.lsi.upc.edu/freeling/doc/userman/html/node10.html

5.1.3. Disseny de la demostració web

Aquest apartat és una breu introducció a les funcionalitats i disseny de l'aplicació web destinada a demostrar les capacitats del motor de cerca desenvolupat en el projecte.

L'aplicació web esta composada de dues vistes principals: una pàgina de cerca i una pàgina d'ajuda. La segona és una pàgina estàtica on s'explica breument però amb claredat com exprimir les funcionalitats del cercador.

Pàgina de cerca: Aquesta és la pàgina d'inici de l'aplicació web. Consisteix en una barra de cerca on l'usuari introdueix la petició de cerca (en el format vist en l'apartat *5.1.1.6. Nuuie-Query*) i un desplegable on es pot seleccionar el idioma en el que es vol realitzar la cerca.

fig. 13 Pàgina d'inici de la demostració web.

fig. 14 Detall de la barra de cerca.

Pàgina de resultats: Una vegada realitzada la petició l'usuari veu aquesta pàgina on se li mostren els resultats. En cada vista es mostren un màxim de 10 resultats.

fig. 15 Pàgina de resultats de la demostració web.

Cada resultat mostra: el nom del document (que pot ser o bé el títol o el nom del fitxer) amb un enllaç que permet descarregar el fitxer original, la última data de modificació, un fragment rellevant del contingut y una llista de etiquetes assignades al document (de moment només el llenguatge). Si es volen veure més resultats és necessari utilitzar el panell de paginació situat al final de la pàgina.

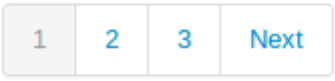


fig. 16 Detall de la paginació.

A més a més la pàgina inclou la barra de cerca de la pàgina d'inici. Al menú de l'esquerra hi ha un desplegable en acordió que mostra opcions avançades, en la versió actual de l'aplicació la demostració web només pot aplicar filtres per dates de modificació, però gràcies al disseny de l'aplicació és possible afegir noves estratègies per filtrar els resultats.

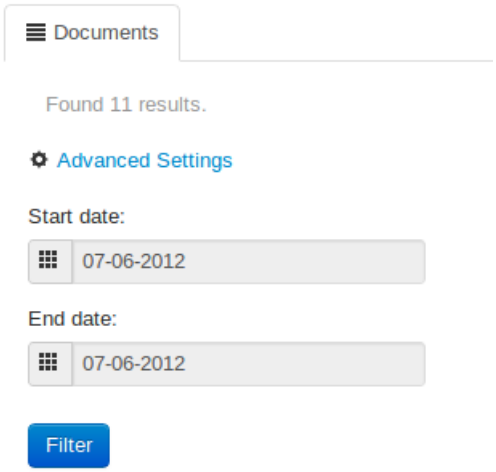


fig. 17 Detall del formulari de filtre per data de modificació.

La selecció de dades es realitza a través d'un calendari[31], on es pot navegar tant a nivell de dies, com de mesos o anys fàcilment:

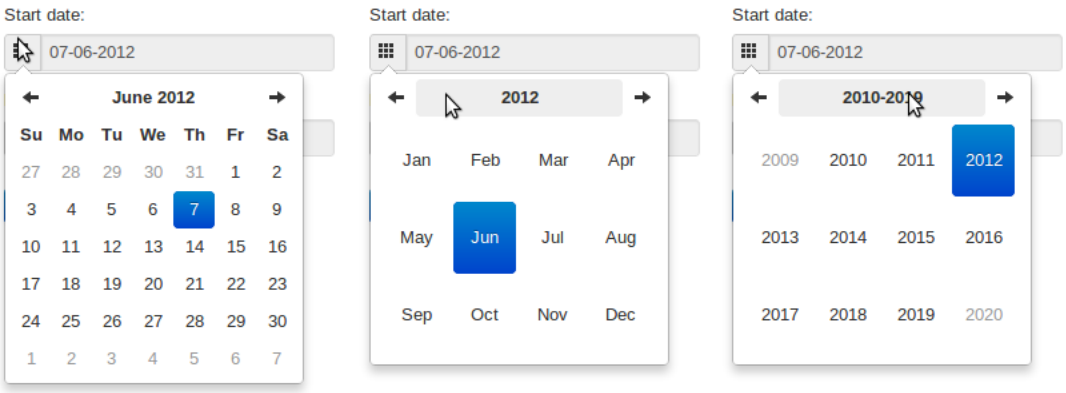


fig. 18 Detall de la interacció amb el calendari.

[31] http://www.eyecon.ro/bootstrap-datepicker/

A més a més l'aplicació web esta disponible per a dispositius mòbils (ja que s'adapta automàticament a qualsevol resolució), tal i com mostren les següents captures:

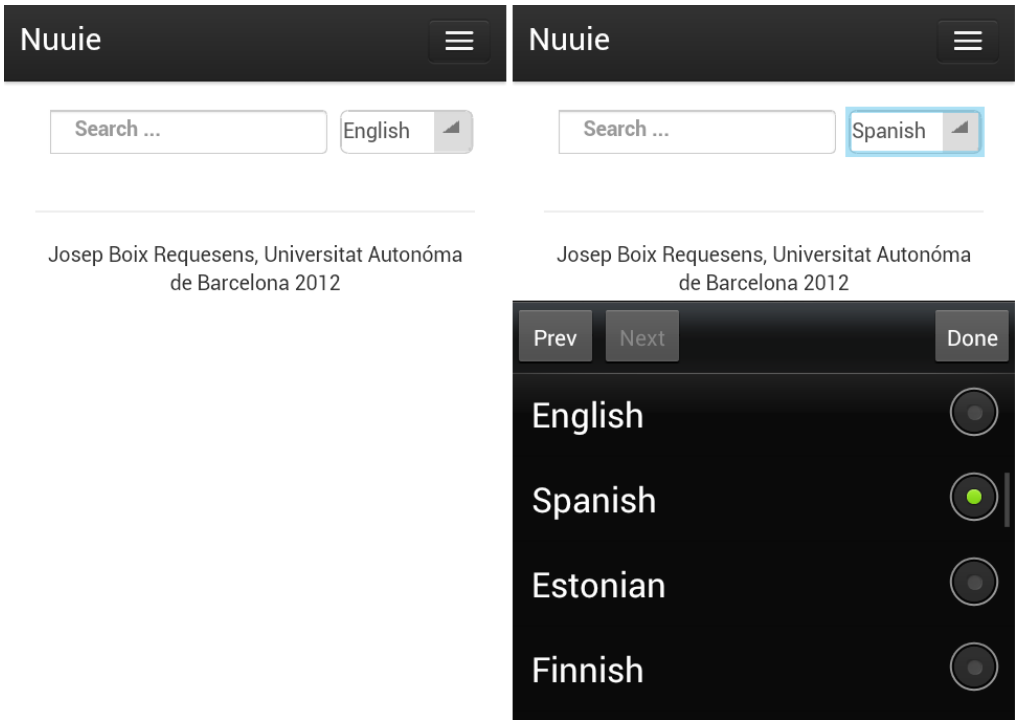


fig. 19 Pàgina principal en un dispositiu mòbil.

fig. 20 Pàgina de resultats i detall de la paginació en un dispositiu mòbil.

5.2. Proves

Les proves realitzades es divideixen en dos grups: proves unitàries i proves d'integració.

5.2.1. Proves unitàries

El conjunt de proves unitàries es centra en comprovar el correcte funcionament de parts aïllades del codi en un entorn simulat. Aquestes proves esta en el codi, la única que destacarem en aquest document és *SearchTest*.

La prova anomenada *SearchTest* és una prova que comprova el correcta funcionament de la navegació a través de la pàgina web en la totalitat de l'aplicació sota un entorn simulat. Per tal d'aconseguir aquest propòsit segueix els següents passos:

1. Crear un índex temporal en memòria RAM amb un total de 11 documents sota una configuració per defecte (és a dir, sense que FreeLing estigui implicat en el procés).
2. Comprovar el correcte funcionament de la barra de cerca a través d'una petició que retorni tots els documents.
3. Navegar a través de les pàgines i comprovar que el contingut retornat és l'esperat.
4. Utilitzar el filtre per data de modificació amb una data que no retorna cap document i una altra que els retorna tots.

D'aquesta manera ens assegurem que qualsevol de detectar qualsevol canvi en la total de l'aplicació que afecti negativament la interacció entre els diferents mòduls i la renderització de l'aplicació web.

Les següents captura de *Sonar* mostren, per a cada mòdul del projecte, el percentatge de codi provat en el joc de proves unitàries:

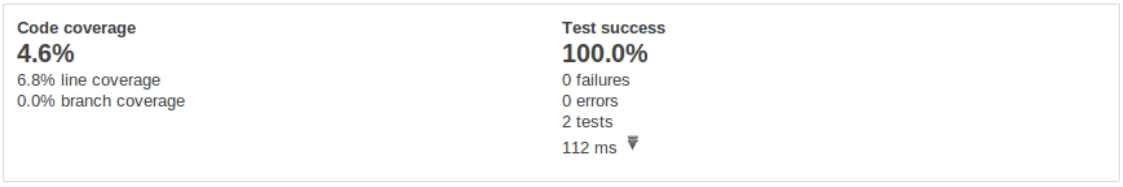


fig. 21 Percentatge de codi cobert per el projecte Morfo.

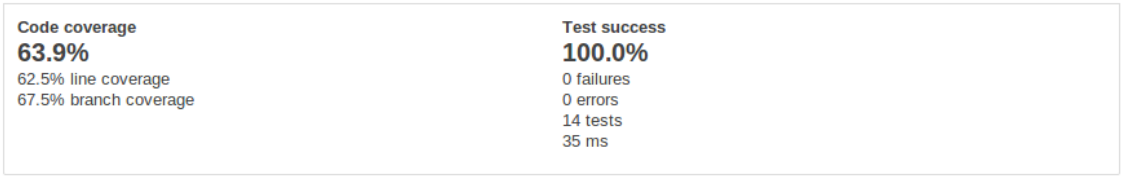


fig. 22 Percentatge de codi cobert per el projecte JGram.

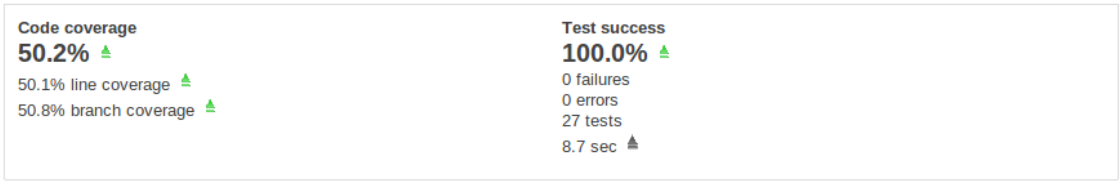


fig. 23 Percentatge de codi cobert per el projecte Nuuie Aggregator.

Name	Rules compliance	Coverage
Backend project for Nuuie	99.9%	58.7%
Query project for Nuuie	99.4%	34.1%
Common project for Nuuie	99.5%	47.2%
Web demo project for Nuuie	99.2%	56.6%

fig. 24 Percentatge de codi cobert per cada mòdul del agregador.

El cicle de vida dels projectes esta estretament relacionat amb aquest joc de proves, ja que si les proves no són vàlides **no es procedeix a la compilació** (amb l'objectiu d'evitar una distribució corrupte).

5.2.2. Proves d'integració

Les proves d'integració s'encarreguen de comprovar el correcte funcionament d'un conjunt de mòduls en un entorn real, a més a més permeten depurar l'aplicació en un entorn d'integració. És important entendre que aquest joc de proves no forma part del cicle de compilació.

El joc de proves d'integració es compon de:

- *AnalyzerDemo*: Encarregada de depurar el funcionament de la política d'extracció de tokens, funciona per a qualsevol a analitzador però és particularment interessant utilitzar-la en un entorn real amb un servidor FreeLing. Com entrada accepta un fitxer o una cadena de caràcters, a continuació executa l'analitzador i retorna una llista de tokens extrets on es mostra tota la informació de manera coherent i fàcil d'analitzar. Per a cada paraula del text original mostra els tokens extrets juntament amb la seva posició relativa a nivell de caràcter tal i com Lucene els rebrà:

```
Analyzing "Frase de prueba."

  FreeLingAnalyzer:

1: [frase::0->5:word] [frase::0->5:word]
3: [prueba::9->15:word] [prueba::9->15:word]

Elapsed time: 20

[frase] [frase] [prueba] [prueba]
```

fig. 25 Resultat d'exemple del *AnalyzerDemo*.

- *Morfo*: Encarregada de depurar el funcionament del mòdul *morfo* (veure *5.1.1.2. Morfo*) en un entorn real. De la mateixa manera que l'anterior accepta un fitxer d'entrada o una cadena i mostra el resultat, a diferència de l'anterior prova, aquí no es mostra cap tipus de filtratge dels tokens (no hi ha normalització de majúscules i minúscules ni eliminació de *stop words*).

```
Sentence: 0
0: [TOKEN="bebiendo", LEMMA="beber", POSTAG="VMG0000", PROBABILITY=1.0", START=0", END=10"], Bebiéndose
1: [TOKEN="se", LEMMA="se", POSTAG="PP3CN000", PROBABILITY=1.0", START=8", END=10"], se
2: [TOKEN="su", LEMMA="su", POSTAG="DP3CS0", PROBABILITY=1.0", START=11", END=13"], su
3: [TOKEN="última", LEMMA="último", POSTAG="AO0FS0", PROBABILITY=1.0", START=14", END=20"], última
4: [TOKEN="botella", LEMMA="botella", POSTAG="NCFS000", PROBABILITY=1.0", START=21", END=28"], botella
5: [TOKEN="de", LEMMA="de", POSTAG="SPS00", PROBABILITY=0.999919", START=29", END=31"], de
6: [TOKEN="vino", LEMMA="vino", POSTAG="NCMS000", PROBABILITY=0.5625", START=32", END=36"], vino
7: [TOKEN=".", LEMMA=".", POSTAG="Fp", PROBABILITY=1.0", START=36", END=37"], .
Elapsed time: 9
Sentence: 1
0: [TOKEN="tiene", LEMMA="tener", POSTAG="VMIP3S0", PROBABILITY=1.0", START=38", END=43"], Tiene
1: [TOKEN="la", LEMMA="el", POSTAG="DA0FS0", PROBABILITY=0.972146", START=44", END=46"], la
2: [TOKEN="nariza", LEMMA="nariza", POSTAG="VMIP3S0", PROBABILITY=1.0", START=47", END=53"], nariza
3: [TOKEN="roja", LEMMA="rojo", POSTAG="AQ0FS0", PROBABILITY=1.0", START=54", END=58"], roja
4: [TOKEN=".", LEMMA=".", POSTAG="Fp", PROBABILITY=1.0", START=58", END=59"], .
Elapsed time: 3
------
Total Elapsed Time: 13
Total Sentences: 2
```

fig. 26 Resultat d'exemple de *Morfo*.

- *Nuuie*: Permet comprovar i depurar el correcte funcionament del mòdul *nuuie* (veure apartat *5.1.1.4 Nuuie*) en un entorn real. Accepta un entrada en forma de directori i recórrer el directori tot analitzant i indexant els documents extrets del sistema de fitxers.

```
2012-06-09 17:21:59,768 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Analyzing document: Grooverum.docx
2012-06-09 17:21:59,769 [Thread-1] INFO  com.uab.nuuie.concurrent.Collector - Found document: file:/home/josep/test
2012-06-09 17:21:59,836 [Thread-1] INFO  com.uab.nuuie.lang.JGramLanguageDetector - Detected language: SPANISH
2012-06-09 17:21:59,839 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Elapsed Time: 71ms
2012-06-09 17:21:59,839 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Analyzing document: Trasfondos.docx
2012-06-09 17:21:59,840 [Thread-1] INFO  com.uab.nuuie.concurrent.Collector - Found document: file:/home/josep/test
2012-06-09 17:21:59,908 [Thread-1] INFO  com.uab.nuuie.lang.JGramLanguageDetector - Detected language: SPANISH
2012-06-09 17:21:59,911 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Elapsed Time: 72ms
2012-06-09 17:21:59,912 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Analyzing document: Tablas de encuentro
2012-06-09 17:21:59,913 [Thread-1] INFO  com.uab.nuuie.concurrent.Collector - Found document: file:/home/josep/test
2012-06-09 17:22:00,019 [Thread-1] INFO  com.uab.nuuie.lang.JGramLanguageDetector - Detected language: SPANISH
2012-06-09 17:22:00,027 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Elapsed Time: 115ms
2012-06-09 17:22:00,027 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Analyzing document: preferida de Lolth.
2012-06-09 17:22:00,028 [Thread-1] INFO  com.uab.nuuie.concurrent.Collector - Found document: file:/home/josep/test
2012-06-09 17:22:00,090 [Thread-1] INFO  com.uab.nuuie.lang.JGramLanguageDetector - Detected language: SPANISH
2012-06-09 17:22:00,091 [Thread-1] INFO  com.uab.nuuie.concurrent.Collector - Found document: file:/home/josep/test
2012-06-09 17:22:00,109 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Elapsed Time: 82ms
2012-06-09 17:22:00,109 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Analyzing document: historia.txt
2012-06-09 17:22:00,140 [Thread-1] INFO  com.uab.nuuie.lang.JGramLanguageDetector - Detected language: SPANISH
2012-06-09 17:22:00,140 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Elapsed Time: 31ms
2012-06-09 17:22:00,140 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Analyzing document: datos de astor.txt
2012-06-09 17:22:00,142 [Thread-1] INFO  com.uab.nuuie.concurrent.Collector - Found document: file:/home/josep/test
2012-06-09 17:22:00,228 [Thread-1] INFO  com.uab.nuuie.lang.JGramLanguageDetector - Detected language: ENGLISH
2012-06-09 17:22:00,232 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Elapsed Time: 92ms
2012-06-09 17:22:00,232 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Analyzing document: gorgias.txt
2012-06-09 17:22:00,253 [Thread-2] INFO  com.uab.nuuie.concurrent.Indexer - Elapsed Time: 21ms
2012-06-09 17:22:01,899 [main] INFO  com.uab.nuuie.Nuuie - Total Elapsed Time: 11568
```

fig. 27 Resultat d'exemple de *Nuuie*.

- *NuuieQuery*: Permet comprovar i depurar el correcte funcionament del mòdul *nuuie-query* (veure *5.1.1.6. Nuuie-Query*) en un entorn real. Accepta una entrada en forma de petició i retorna els 10 primers resultats obtinguts d'un índex prèviament generat.

```
{URL: file:/home/josep/test/D&D/Veros/Regiones/Bof-Gak/Bof-Gak.docx,
LANGUAGE: SPANISH,
DATE: Sat Apr 11 05:20:26 CEST 2009,
CONTENT: <b>....</b> puede <b>encontrar</b> comida con una tirada a supervivencia con un+5 a la dificulta
{URL: file:/home/josep/test/D&D/Regiones/Gislaved/Quests/Traición/Traición.docx,
LANGUAGE: SPANISH,
DATE: Sat Sep 27 02:04:14 CEST 2008,
CONTENT: <b>....</b> interrumpirán el <b>encuentro</b> dando la posibilidad a Mogocoria de huir. Viendo q
{URL: file:/home/josep/test/D&D/Regiones/Gislaved/Quests/Espiar%20al%20Príncipe/Espiar%20al%20príncipe.do
LANGUAGE: SPANISH,
DATE: Sat Sep 27 01:44:27 CEST 2008,
CONTENT: <b>....</b> probabilidad de <b>encuentro</b> la primera hora y aumentará en un 10% a cada hora s
{URL: file:/home/josep/test/D&D/Regiones/Gislaved/Quests/Kabyle%20Aislada/Kabyle%20Aislada.docx,
LANGUAGE: SPANISH,
DATE: Wed Sep 03 13:08:02 CEST 2008,
CONTENT: <b>....</b> deben <b>encontrar</b> al alcalde y explicarle la situación, la única ayuda que podr
{URL: file:/home/josep/test/D&D/Regiones/Gislaved/Quests/Investigar%20Damazín/Investigar%20Damazin.docx,
LANGUAGE: SPANISH,
DATE: Wed Sep 24 01:12:06 CEST 2008,
CONTENT: <b>....</b> quieren. Un <b>encuentro</b> con un Centauro cazador en el bosque también serviría p
{URL: file:/home/josep/test/D&D/segment%20narratiu.docx,
LANGUAGE: SPANISH,
DATE: Sat Jun 13 17:01:23 CEST 2009,
CONTENT: <b>....</b> posición (<b>encuentro</b> con 4 alips). Durante el combate la gente empieza arreple
2012-06-09 17:26:23,554 [main] INFO  com.uab.nuuie.query.NuuieQuery - Found: 55 results.
2012-06-09 17:26:23,554 [main] INFO  com.uab.nuuie.query.NuuieQuery - Total Elapsed Time: 80 ms.
```

fig. 28 Resultat d'exemple de *NuuieQuery*.

- *NuuieWebStart*: Aquesta prova desplega un servidor a través de *jetty* que permet utilitzar la totalitat de l'aplicació web a través d'un navegador. Les proves realitzades sobre aquest joc han estat exclusivament manuals tot i que en una etapa més avançada del projecte es podrien automatitzar a través de *Selenium* o algun altre framework d'integració web.

Els documents de prova utilitzats varien segons el tipus de prova, existeix diferents jocs de document: Un joc de documents que correspon a un partida de rol de *Dungeons and Dragons* en castellà, un recull de llibreries de filosofia en anglès i un joc de proves destinar a estressar l'algoritme d'ordenació i la lematització (cada document consisteix en una única sentència on les paraules varien lleugerament).

6. Conclusions

Finalment l'objectiu del projecte (el de crear un cercador capaç de aprofitar les avantatges de lematització en diferents idiomes) ha estat assolit, de la mateixa manera que els objectius descrits en l'estudi de viabilitat. Tot i així, un dels requisits descrits ha estat descartats durant el transcurs del projecte degut als problemes que han sorgit durant el desenvolupament: l'exportació dels resultats en format JSON i XML. La raó per la que s'ha desistit d'implementar aquesta part és perquè el projecte pot viure també com a mòdul complet i no només com a serveis separats. Tot i així altres característiques imprevistes han entrat en joc com són:

- Extracció de fragments rellevants en relació a una petició de cerca.
- Filtració per dates de modificació.
- Extracció de Metadata en els documents (que permet, per exemple, extreure el títol original d'un document o l'autor, proporcionant noves formes de cerc a prou interessants de cara l'usuari).
- L'aplicació web és també compatible amb dispositius mòbils.

Aquests canvis han provocat un retràs en la planificació original del projecte (amb entrega prevista per Febrer de 2012), tot i així han ajudat a millorar l'experiència de l'usuari.

La dificultat principal del projecte ha consistit en conciliar Lucene amb un motor de lematització com és FreeLing, el problema principal de FreeLing és que no ofereix una forma clara d'integració en Java. En un principi es va optar per utilitzar la interficie JNI que proporciona per defecte, però (com s'explica en l'apartat *5.1.1.2 Morfo*) aquesta opció va quedar descartada per suposar un coll de botella. El desenvolupament de un analitzador capaç de relacionar una sentència analitzada per el servidor de FreeLing amb la sentència original no és trivial, especialment quan es tracta amb paraules compostes o noms propis (per exemple, FreeLing interpreta la paraual *pel* com dos tokens: *pel* i *el*, però la sentència original no conté aquests tokens! revertir aquest procés sense afectar el rendiment és una tasca prou complexa).

Una altra dificultat important ha estat la classificació del llenguatge, però especialment resoldre la qüestió: Com proveir al desenvolupador d'eines per tal de que sigui capaç de determinar un interval de confiança en relació a la detecció efectuada? I fer-ho sense tenir en compte la precedència de n-grames per tal de que la classificació no suposi un coll de botella. Finalment es va optar per una simplificació inspirada en la implementació d'un classificador bayesià. El resultat és prou satisfactori i seria possible continuar aquesta línea de desenvolupament proporcionant eines per calcular intervals de confiança segons un joc de proves entrenat (que, en cap cas, s'ha implementat).

Un punt fort a destacar del projecte, a nivell de metodologies de desenvolupament, és la integració de cicles de versions, compilació i desplegament a través de *Maven* a més a més de la integració amb *Sonar* (que permet analitzar detectar i solucionar problemes molt ràpidament). S'ha procurat en tot moment també desenvolupar totes les classes de manera que sigui fàcil desenvolupar proves respectant la filosofia de proves unitàries darrera *Mockito*. Tot això, juntament amb un desenvolupament basat en patrons de disseny (el codi està ple d'exemples pràctics d'estratègies, composició, factories, inversió de control, etc.) faciliten enormement el futurs treball sobre aquesta aplicació.

6.2. Treball futur

La següent llista és un recull de les possibles ampliacions que es podrien realitzar sobre aquest projecte:

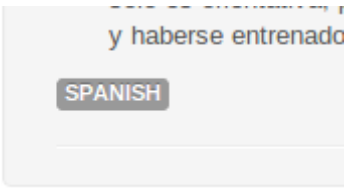
1. Desenvolupar una eina de càlcul d'intervals de confiança per *JGram*.

2. Classificació dels mime-types proporcionats per *Tika* per a una millor distinció del tipus de fitxer indexat, en aquest moment només es distingeix entre *application* (documents per a nosaltres), *image*, *video* i *audio*. Tot i així la detecció de documents es pot millorar, ja que molts dels tipus de fitxers que cauen sota la categoria *application* no són en realitat documents (tot i així aquesta és una tasca manual i força tediosa).

3. Investigar millores en la detecció del llenguatge per a textos petits utilitzant diccionaris (FreeLing proporciona una base extensa de diccionaris i es podrien explotar també en aquest sentit).

4. Integració més extensa de la metadata dels fitxers per tal de fer una classificació més exhaustiva dels camps rellevants, per exemple: La mida de les imatges o l'artista d'una cançó o la data de creació d'un document (de moment només s'indexa la de última modificació).

5. La possibilitat de filtrar directament des de les etiquetes d'un resultat en l'aplicació web. Aquesta és una característica que no queda explicada en la memòria, ja que no vaig tenir temps d'implementar-la. A sota de cada resultat es veu una etiqueta, que de moment només mostra el llenguatge del document i no es pot interactuar amb ella:



   La idea és que directament des d'aquest element sigui possible filtrar la cerca, per exemple: Pel nom de l'artista d'una cançó o l'autor d'un document.

6. Categorització dels documents basada en el contingut[32]: crec que aquest és un punt d'investigació força interessant i esta relacionat també amb el treball realitzat per a la classificació de llenguatges. La mateixa llibreria es podria utilitzar per classificar els textos segons el contingut a través de categories prèviament establertes. Per tal d'aconseguir això seria necessari tenir un corpus força extens de documents sobre diferents temes.

7. Integració amb lucene-spellchecker: Aquesta llibreria permet suggerir correccions a la petició de l'usuari de la mateixa manera que és capaç de generar autocompletació de peticions. Només necessita que se li proveeixi el diccionari (com hem vist en altres ampliacions, FreeLing proporciona diccionaris per a tots els idiomes que integra).

8. Rastrejador de webs. Com s'ha pogut veure durant el transcurs de la memòria la indexació és independent de la font de documents, desenvolupar un *crawler* sota aquestes condicions no és difícil. Es podria utilitzar per indexar una pàgina web i proporcionar un motor de cerca sobre el contingut d'aquesta pàgina.

9. *Renewer* com a procés independent: De moment el *Renewer* verifica l'existència de tots els documents de l'índex abans de procedir a la indexació. Sota aquest disseny escalar l'aplicació pot ser complicat, ja que suposa un coll de botella. Cal modificar aquest disseny perquè es pugui executar com a procés resident, és a dir, independent del procés d'indexació, addicionalment hauria de processar els documents per *batches*.

7. Bibliografia

[1] Otis Gospodnetic, Erik Hatcher, *Lucene in Action*, Manning, 2005.
[2] Martijn Dashorst, Eelco Hillenius, *Wicket in Action*, Manning, 2009.
[3] Aivosto Oy, http://www.aivosto.com/project/help/pm-oo-cohesion.html, 10-06-2012
[4] Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato, http://svnbook.red-bean.com/en/1.6/svn-book.html#svn.tour.importing.layout, 10-06-2012.
[5] Іван Куцкір, http://blog.ivank.net/trie-in-as3.html, 03-02-2012.
[6] https://github.com/twitter/bootstrap/issues/445, 15-05-2012.
[7] http://maven.apache.org/what-is-maven.html, 01-06-2012.
[8] http://www.sonarsource.org/support/documentation/ 01-12-2011.
[9] Lluís Padró, http://nlp.lsi.upc.edu/freeling/doc/userman/html/ , 01-12-2011.
[10] William B. Cavnar, John M. Trenkle, http://www.let.rug.nl/vannoord/TextCat/textcat.pdf, 10-06-2012.
[11] Oracle,http://docs.oracle.com/javase/1.5.0/docs/guide/jni/spec/jniTOC.html, 10-01-2012.
[12] Robert C. Martin, http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf, 16-05-2012.

[32] http://www.let.rug.nl/vannoord/TextCat/textcat.pdf