Text Plagi, detecció de text no citat

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Gestió

realitzat per

Albert Martínez Vilanova

i dirigit per

Jordi Duran Cals

Escola d'Enginyeria

Sabadell, *Setembre* de *2013*

El sotasignat, ***Jordi Duran Cals***,

professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció per

Albert Martínez Vilanova

I per a que consti firma la present.
Sabadell, ***Setembre*** de ***2013***

Signat: ***Jordi Duran Cals***

Resum del projecte

El present document relata, especifica i justifica l'elaboració del Projecte Final de Carrera de la diplomatura d'Enginyeria Tècnica d'Informàtica de Gestió defensat per l'alumne Albert Martínez Vilanova a l'escola Universitària d'Informàtica de Sabadell – EUIS al Setembre de 2013.

El treball presenta una aplicació web que permet la detecció de text plagiat en un arxiu prèviament seleccionat per l'usuari gràcies a l'API Summon Service[1] a través d'un entorn web.

La idea sorgeix del director de projecte Jordi Duran Cals quan em va proposar una col·laboració amb la Universitat Oberta de Catalunya per desenvolupar aquesta nova eina, que donaria solució a la necessitat dels professors de poder detectar possibles plagis.

Amb la nostra aplicació hem complert l'objectiu principal de crear satisfactòriament i amb excel·lents resultats una aplicació web que busca i detecta entre els documents continguts a la biblioteca virtual de la Universitat Oberta de Catalunya mitjançant l'ús de Summon Service.

[1] Summon Service és una biblioteca virtual on hi podem trobar tots els artícles, comunicacions, actes de congresos, legislacions i bases de dades a les que s'està subscrit.

Taula de continguts

1 Introducció

1.1 Motivacions

Cada cop més, fruit del gran accés a la informació a través de la xarxa, les persones utilitzen de manera lliure, textos i recursos sense cap limitació o restricció. És per això que és molt habitual trobar documents amb text copiat literalment de fonts terceres.

Per aquest motiu, la detecció de plagi ha esdevingut en els darrers anys una eina crucial dins l'entorn educatiu, però la majoria de cops, són eines destinades als avaluadors, per tal que detectin text copiat i no mecanismes per detectar el text no citat.

1.2 Estructura de la memòria del projecte

Hem organitzat la memòria en els apartats més importants en quant el desenvolupament d'un projecte d'aquest tipus requereix.

El primer capítol ens serveix a mode d'introducció i resum del que parlarem en més detall en els següents capítols.

En el segon capítol presentem l'estudi de viabilitat on s'ha analitzat tots els aspectes que intervenen en la construcció de la nostra aplicació per poder decidir si és factible o no dur-la a terme. Exposarem els objectius i requisits així com la situació actual, és a dir, quin problema volem resoldre i què s'està fent actualment per solucionar-ho, què proposem nosaltres i quines alternatives trobem en el mercat, anàlisi de riscos, costos, planificació, etc.

La següent secció es parla de com s'ha plantejat i dut a terme la solució. Hi trobem el model de desenvolupament, l'entorn de desenvolupament i el disseny.

El quart capítol està dedicat a les proves a que hem sotmès l'aplicació.

La secció cinc és un resum sobre les conclusions obtingudes partint de la idea inicial i quines futures línies de treball aplicar-hi, així com una valoració a nivell personal.

Finalment, l'últim capítol està dedicat a bibliografia i agraïments.

2 Estudi de viabilitat

2.1 Introducció

La intenció d'aquest apartat és la de fer un estudi de mercat en relació als productes actuals relacionats amb el que presentem. Abans de tirar endavant el projecte, s'han d'avaluar riscos i calcular els beneficis esperats per decidir-ne la seva viabilitat.

El punt clau és que amb la nostra aplicació volem proporcionar al mercat una eina capaç de respondre a una necessitat real.

2.2 Descripció

Partint de la definició del concepte plagiar que dóna el Diccionari de la Llengua Catalana de l'Institut d'Estudis Catalans,

> *Copiar (idees, paraules, obres, etc., d'una altra persona) usant-les com a pròpies.*

És fàcil agafar prestades idees o paraules d'altres. El problema és no fer-ho explícit quan es tracta d'una obra personal per presentar-se en avaluacions o congressos.

Amb l'ajuda d'internet i les noves tecnologies, el "copiar i pegar" s'està convertint en un greu problema per als avaluadors del sistema educatiu. És on entra en joc la nostra aplicació que no pretén ser un *anticòpia* amb el seu sentit més estricte de la paraula (penar al plagiador) sinó detectar aquelles parts copiades sense referenciar a l'autor.

2.3 Objectius del projecte

1. Pujar els documents a tractar al servidor.
2. Dividir els documents segons estratègies, sense importar la seva extensió.
3. Cercar coincidències a l'API de Summon Service.
4. Visualitzar la informació obtinguda en un gestor de continguts web.
5. Els procediments de pujar els documents al servidor, dividir els documents i buscar coincidències a Summon Service es realitzin de manera asíncrona.
6. Pujar els documents a tractar al servidor utilitzant la tecnologia "drag and drop".
7. Opció d'enviar els resultats mitjançant correu electrònic i crear un arxiu pdf amb tota la informació obtinguda per aquell document.

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

Taula 1: Taula de catalogació d'objectius.

2.4 Parts interessades

2.4.1 Stakeholders

Nom	Descripció	Responsabilitat
Albert Martínez Vilanova	Analista, programador.	Participa en la definició de requisits, representa el usuari tipus. Participa en la validació del projecte.
Jordi Duran Cals	Director del projecte.	Supervisa la feina de l'alumne, avalua el projecte.

Taula 2: Stakeholders que intervenen

2.4.2 Perfils d'usuaris

Tipus	Perfil	Responsabilitat
Equip del projecte	Administrador del sistema.	Gestionar incidències, millores al sistema, suport.
Eng. Tècnic Informàtic	Administrador de l'aplicació.	Gestió, control del sistema. Restaurar les possibles caigudes del servidor, i procurar un servei eficient.
Professorat o altres usuaris	Usuari no expert.	Consultar la informació usant l'aplicació.

Taula 3: Relació dels diferents perfils d'usuari.

2.4.3 Equip de projecte

Nom	Descripció	Responsabilitat
Albert Martínez Vilanova	Cap del projecte	Defineix, gestiona, planifica i controla el projecte.
	Analista	Col·labora en l'estudi de viabilitat i la planificació. Realitza un anàlisi exhaustiu de l'aplicació. Participa en el disseny i la validació.
	Programador	Participa en el disseny i realitza les proves internes i externes. Participa en el procés de control de qualitat
Jordi Duran Cals	Director del projecte / Tutor	Supervisa el projecte.

Taula 4: Relació de l'equip de projecte.

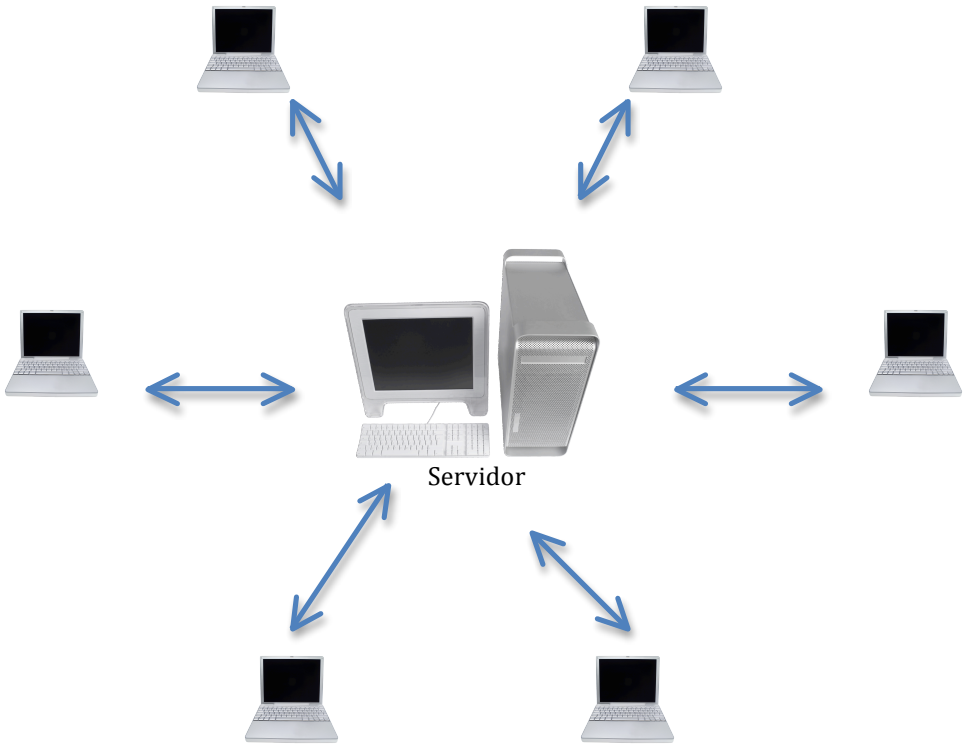
2.5 Estudi de la situació actual

2.5.1 Context

Actualment existeixen diferents eines que volen solucionar aquest problema. Parlem d'aplicacions web que permeten cercar continguts plagiats.

2.5.2 Descripció física

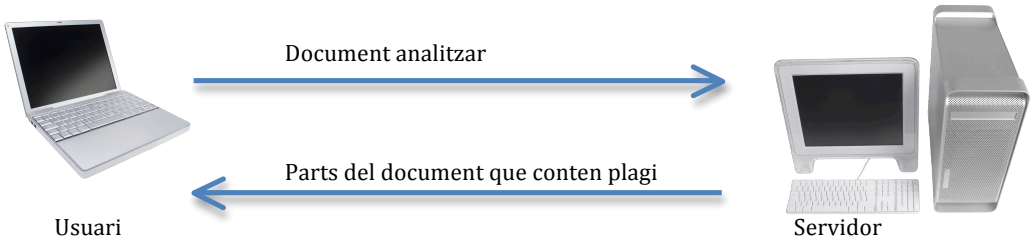
La imatge mostra com els diferents usuaris es connecten a l'aplicació de manera bidireccional i reben informació.



Il·lustració 1: Graf esquemàtic il·lustrant la connectivitat.

2.6 Lògica del sistema

La imatge següent mostra el diagrama de flux del sitema actual.



Il·lustració 2: Graf esquemàtic il·lustrant el flux de treball.

2.7 Requisits del projecte

2.7.1 Requisits funcionals

1. El software té com a referència el càlcul ràpid i eficient en totes les operacions.
2. Manteniment, gestió i administració dels documents pujats pels usuaris al servidor.
3. Manteniment i gestió de les consultes a través del sistema.
4. Extracció dels resultats en fitxers compatibles amb eines d'ofimàtica dedicades a la lectura de documents pdf.

2.7.2 Requisits no funcionals

1. Usabilitat: la interfície ha de complir la ISO 9241: norma enfocada a la qualitat sobre l'usabilitat i ergonomia tant de hardware com de software.
2. L'usuari ha de ser capaç d'utilitzar totes les funcions del sistema sense cap problema.
3. Quant hi hagi fins a 100 usuaris accedint al sistema simultàniament, el seu temps de resposta no serà en cap cas superior a 2 segons.
4. Davant un problema en el software del sistema, no es trigarà més de 30 minuts en restaurar i tornar a posar en marxa el sistema.
5. El sistema ha de poder suportar vàries sessions en paral·lel.
6. El codi font del software ha de ser modulat per tal de poder suportar futures ampliacions.
7. Compliment amb la normativa sobre els estàndards utilitzats en el procés de desenvolupament i adaptable a qualsevol entorn tan com Sistema Operatiu, hardware del servidor, connectivitat, etc.

2.7.3 Restriccions del sistema

En aquest sector actualment tant el maquinari com els sistemes operatius usats són molt variables. Per tant el nostre producte ha de ser totalment compatible i adaptable en qualsevol entorn, tant de hardaware com de software.

2.7.4 Catalogació i priorització dels requisits

Funcionals

	RF1	RF2	RF3	RF4
Essencial		X	X	
Condicional	X			
Opcional				X

Taula 5: Taula dels requisits funcionals.

No Funcionals

	RNF1	RNF2	RNF3	RNF4	RNF5	RNF6	RNF7
Essencial	X	X	X		X		X
Condicional				X		X	
Opcional							

Taula 6: Taula dels requisits no funcionals.

Relació entre els requisits i objectius

	RF1	RF2	RF3	RF4	RNF1	RNF2	RNF3	RNF4	RNF5	RNF6	RNF7
O1	X	X					X			X	X
02	X								X	X	X
03	X		X					X	X	X	X
04	X				X	X	X	X			X
05									X	X	X
06						X	X	X	X	X	X
07				X					X	X	X

Taula 7: Relació d'objectius del projecte amb requisits funcionals i no funcionals.

2.8 Alternatives i selecció de la solució

En aquest apartat exposarem les diferents opcions en relació a les diverses eines que ja existeixen al mercat que ens poden servir per detectar els plagis i els motius que han motivat la nostra elecció final.

2.8.1 Alternativa 1: Google Scholar

Google Scholar és un buscador de Google especialitzat en articles de revistes científiques, enfocat al món acadèmic.

Web: http://scholar.google.es/

Avantatges	Inconvenients
Gratuït.	Actualment, encara no esta disponible l'API.
Permet buscar copies físiques o digitals d'articles, ja sigui online o en biblioteques.	
Utilitza un algoritme similar al que utilitza el propi Google per a les cerques generals, per tant, és senyal de qualitat.	

Taula 8: Relació d'avantatges i inconvenients entre l'alternativa 1.

Els costos derivats d'aplicar aquesta alternativa no els podem determinar perquè no disposem d'aquestes dades, amb l'afegit que Google encara no a publicat l'API cosa que descarta aquesta opció.

2.8.2 Alternativa 2: Plagium

Plagium és un servei de Septet Systems Inc. especialitzada en solucions avançades de cerques.

Web: http://www.plagium.com/

Avantatges	Inconvenients
No esta limitat al número de termes que pot ingressar en cada consulta.	No pot dir-nos si s'ha plagiat el text.
Gratuït.	API en fase beta.
Permet guardar un històric de cerques.	

Taula 9: Alternativa 2.

Tot i que a diferencia d'altres motors de cerca que estan limitats al número de consultes i de termes inclosos en aquestes, Plagium accepta blocs de text molt més grans.

D'altra banda, Plagium no ens mostra si s'ha plagiat el text, només mostra per pantalla enllaços a documents que contenen blocs de text que coincideixen amb aquells introduïts i tampoc pot determinar la legitimitat o autoritat dels documents que es recuperen. Aquest és un requisit absolutament necessari i determinant per al nostre projecte. És per aquest motiu que es va descartar com a alternativa.

2.8.3 Alternativa 3 i solució proposada

L'eina Summon Service de Serials Solutions és la línia de projecte escollida. Aquesta ens permet trobar tots els articles, comunicacions, actes de congressos, legislacions, etc. que es troben en les revistes i bases de dades en les que la Universitat Oberta de Catalunya està subscrita. L'ús d'aquesta eina no ens a suposat cap cos per al projecte tot i que és de pagament.

Summon ens dóna un accés ràpid i fàcil al text complet, resultats ordenats per rellevància, opcions per refinar els resultats, limitar les cerques a articles acadèmics entre d'altres.

A l'igual que la primera alternativa proposada, Summon i Google Scholar són dos motors de cerca que ens permeten buscar ràpidament en tot un índex massiu d'informació acadèmica, cobrint una part comú de continguts. Aquest motor, a diferència de Google Scholar, disposa d'una API per als desenvolupadors que ens proporciona una gran varietat d'informació que ens seran de gran utilitat en el desenvolupament del nostre projecte.

2.9 Planificació del projecte

Per a la planificació del projecte hem utilitzat l'eina GNU anomenada OpenProj[2].

La data de començament del projecte va quedar fixada al 29 d'Octubre de 2012 i la data aproximada de la finalització, finals de setembre de 2013 un cop feta la presentació davant del tribunal, en la què podem considerar finalitzat el cicle del projecte.

Tot i que segons la primera planificació en el desenvolupament del projecte, aquest conclouria el dia 25 de Març de 2013 però amb una estimació d'aplaçament degut a imprevistos i/o aparició de riscos i una possible planificació optimista. Basant-nos en l'estimació actual de la durada del projecte seria de 19 setmanes.

Albert Martínez Vilanova (AMV)

	Laborables	Festius
Dies	Dilluns-Divendres	Dissabtes i Diumenge
Hores	16h-18h	-

Taula 10: Calendari de treball de AMV.

Això representen 19 setmanes * 10 hores de dedicació setmanal = 190hores.

[2] Més informació sobre el programari a http://sourceforge.net/projects/openproj/

Jordi Duran Cals (JDC)

	Laborables	Festius
Dies	Dimecres	Dilluns-Dissabte
Hores	17h-19h	-

Taula 11: Calendari de participació al projecte de JDC.

En aquest cas, 19 setmanes * 2 hora = 38 hores.

2.10 Recursos del projecte

2.10.1 Recursos humans

Nom	Correu electrònic	Càrrec	Responsabilitat	Cost
Albert Martínez Vilanova (AMV)	mailto:albert.martinez.vilanova@gmail.com	Analista programador	Desenvolupament del projecte	20€/h
Jordi Duran Cals (JDC)	mailto:jduran@deic.uab.cat	Director del projecte	Supervisar el bon desenvolupament així com la seva viabilitat	40€/h

Taula 12: Informació dels actors dins del desenvolupament del projecte.

2.10.2 Recursos materials

Maquinari disponible

Maquinari	Tipus	Característiques	Propietari	Disponibilitat	Cost	Càrrec pel projecte
MacBook Pro	Portàtil	Processador: Intel Core 2 Duo 2,4 Ghz Memòria: 4GB 1067 DDR3 Gràfics: NVIDIA GeForce 320M 256MB Sistema Operatiu: OS X 10.8.4	AMV	Total	1.100€ (Cost al 2010)	0€
Acer Aspire One	Portàtil	Processador: Intel Atom N270 1.6Ghz Memòria 1.5GB Sistema Operatiu Ubuntu Linux 11.10	AMV	Total	300€ (Cost al 2008)	0€

Taula 13: Realació i detall del maquinari disponible.

Programari disponible

Programari	Llicència	Cost pel projecte
OS X 10.8.4	Pagament	0*
Microsoft Office 2011 MAC	Pagament	0*
Ubuntu Linux 11.10	GNU General Public License (GPL)	0
Eclipse JAVA EE	Eclipse Public License	0
Summon Service	Cedit	0
JBoss 6.0	GNU Lesser General Public License (LGPL)	0
Ruby on Rails	MIT	0
JRuby	GNU Lesser General Public License (LGPL)	0
JSON	GNU Freeware	0
Apache Tika	Apache License	0
OpenProj	Common Public Attribution License (CPAL)	0

Taula 14: Programari disponible.

*En el nostre cas, ja es comptava amb el software privatiu previ al projecte.

2.11 Tasques del projecte

Tasca	Subtasca	Predecessors	Recursos	Durada dins del projecte*
1. Fixar els objectius del projecte	1.1 Analitzar la necessitat del mercat actual.		AMV, JDC	2d
	1.2 Estudiar el sector.	1.1	AMV	5d
	1.3 Buscar solucions dins del sector actual	1.1	AMV, JDC	5d
	1.4 Pluja d'idees.		AMV, JDC	1d
	1.5 Plantejar proposta.	1.4	AMV, JDC	3d
	1.6 Fixar objectius.		AMV, JDC	1d
	1.7 Estudi de viabilitat.	1.1 1.2 1.4 1.5	AMV, JDC	12d
2. Fixar recursos del projecte	2.1 Marcar els suports que necessitem.	1.7	AMV, JDC	12d
	2.2 Anàlisi sistema operatiu (cost, compatibilitat, manteniment, etc.).	2.1	AMV	1d
	2.3 Programari requerit, el seu cost i adquisició.	2.2	AMV	1d

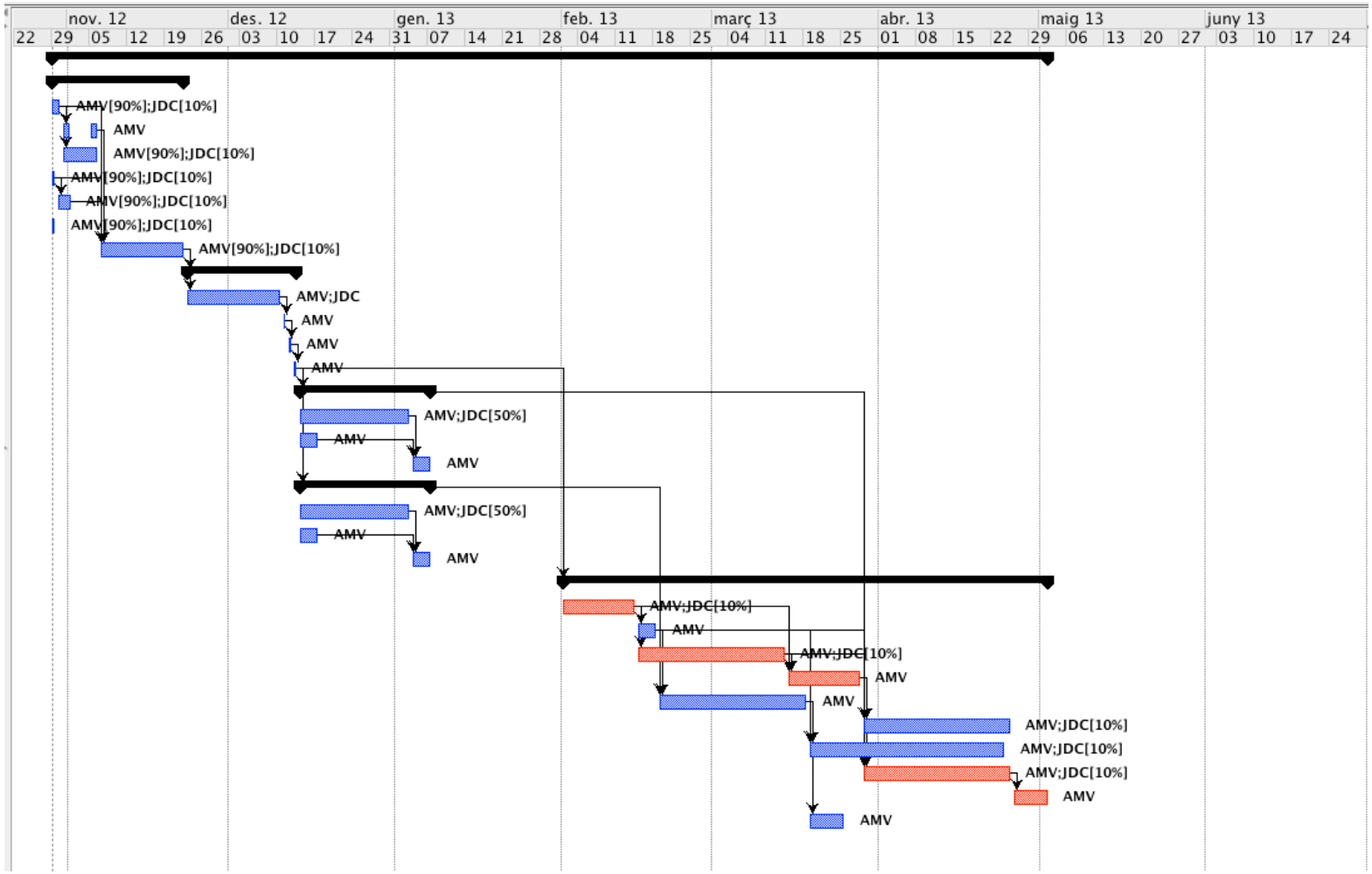
	2.4 Escollir i configurar l'entorn de desenvolupament.	2.3	AMV	1d
3. Estudi de l'API Summon Service	3.1 Documentació.		AMV, JDC	15d
	3.2 Proves.		AMV	2d
	3.3 Instal·lació en el projecte.	3.1 3.2	AMV	2d
4. Estudi de l'API J2EE	4.1 Documentació.		AMV, JDC	15d
	4.2 Proves.		AMV	2d
	4.3 Instal·lació en el projecte.	4.1 4.2	AMV	2d
5. Desenvolupament de l'aplicació	5.1 Dissenyar el model relacional, diagrama de classes, etc.		AMV, JDC	10d
	5.2 Plasmar el disseny en el nostre entorn de desenvolupament.	5.1	AMV	2d
	5.3 Integració del Framework Tika al nostre projecte.	5.1	AMV, JDC	20d
	5.4 Dividir els documents en varies estratègies.	5.1 5.3	AMV	10d
	5.5 Creació de l'entorn web.	4 5.2	AMV	20d
	5.6 Pujar documents	5.2	AMV, JDC	20d

	al servidor.	5.4		
	5.7 Analitzar els documents mitjançant Summon Service.	5.2 5.3 5.5	AMV, JDC	26d16h
	5.8 Obtenir resposta i adaptar-la a l'entorn web creat.	3 5.2 5.3 5.4 5.5	AMV, JDC	20d
	5.9 Generar pdf's i enviar les respostes per correu electrònic.	5.8	AMV	5d
	5.10 Privacitat de les dades en el servidor.	5.5	AMV	5d

Taula 15: Taula de les tasques del projecte planificades a l'octubre de 2012.

*En aquesta taula hem aplicat els temps de tasca segons la planificació. Els dies i les hores es basen en els calendaris de treball de cada un dels actors, no en dies natural

2.12 Planificació temporal



Il·lustració 3: Diagrama de Gantt de la planificació del projecte feta a l'octubre de 2012.

2.13 Avaluació de riscos

2.13.1 Llista de riscos

R1. Planificació temporal optimista: estudi de viabilitat no s'acaba en la data prevista.

R2. Manca d'alguna tasca necessària: estudi de viabilitat. No és compleix els objectius del projecte.

R3. Pressupost poc ajustat: estudi de viabilitat. Menys qualitat i pèrdues econòmiques.

R4. Canvi de requisits: estudi de viabilitat, anàlisi. Endarreriment en el desenvolupament i en el resultat.

R5. Equip del projecte massa reduït: estudi de viabilitat. Endarreriment en la finalització del projecte, no es compleixen els objectius del projecte.

R6. Eines de desenvolupament inadequades: implementació. Endarreriment en la finalització del projecte.

R7. Dificultat per accedir als stakeholders: estudi de viabilitat, anàlisi, proves, formació. Manquen requisits o són inadequats, endarreriments, insatisfacció usuaris.

R8. No es fa correctament la fase de test: desenvolupament, implementació. Manca de qualitat, deficiència en l'operativa, insatisfacció usuaris, pèrdues econòmiques.

R9. Incompliment de la normativa vigent: en qualsevol fase. No es compleixen els objectius del projecte, repercussions legals.

R10. Manca d'implementació de mesures de seguretat: estudi de viabilitat, anàlisi, desenvolupament. Pèrdua d'informació, incompliment legal, pèrdues econòmiques.

R11. Abandonament del projecte abans de la finalització: en qualsevol fase. Frustració.

2.13.2 Catalogació de riscos

	Probabilitat	Impacte
R1	Alta	Crític
R2	Alta	Crític
R3	Alta	Crític
R4	Alta	Crític
R5	Alta	Crític
R6	Alta	Crític
R7	Baixa	Crític
R8	Alta	Crític
R9	Mitjana	Crític
R10	Alta	Crític
R11	Baixa	Catastròfic

Taula 16: Catalogació dels riscos.

2.13.3 Pla de contingència

	Solució que cal adoptar
R1	Ajornar alguna funcionalitat, afrontar possibles pèrdues, fer una assegurança.
R2	Revisar estudi de viabilitat, modificar la planificació existent.
R3	Renegociar amb el client i afrontar possibles pèrdues.
R4	Renegociar amb el client, ajornar funcionalitats, modificar planificació i pressupost.
R5	Demanar un ajornament, renegociar amb el client, afrontar possibles pèrdues.
R6	Millorar la qualitat de l'equip i les eines a utilitzar.
R7	Fer trobades periòdiques per fer el seguiment del projecte.
R8	Redissenyar els test, que siguin automàtics. Negociar el contracte de manteniment, afrontar possibles pèrdues.
R9	Revisar la normativa vigent, afrontar possibles repercussions legals.
R10	Revisar la seguretat de cada fase, aplicar diferents polítiques de seguretat.
R11	No té solució.

Taula 17: Resultat d'aplicar les solucions als riscos comentats anteriorment.

2.14 Pressupost

2.14.1 Estimació cost personal

	Hores	Preu/Hora	Cost total
Desenvolupador	190h	20€	3800€
Cap de projecte	38h	40€	1520€

Taula 18: Justificació costos de personal.

La suma total dels recursos és de 5.320€.

2.14.2 Estimació cost dels recursos

Cost que suposaria si se n'hagés de fer càrrec el projecte.

	Cost amortització	Cost unitari	Període utilització
MacBook Pro	234€	1100€	4,7m
Asus Aspire One	64€	300€	4,7m
OS X Mountain Lion	4€	17,99€	4,7m
Microsoft Office 2011 MAC	81€	379€	4,7m

Taula 19: Desglaçament costos de maquinari.

Això representa un total de 383€ de cost de recursos de materials.

2.14.3 Resum i anàlisi

El cost del projecte tenint en compte tots els conceptes suma un total de :

- Desenvolupament del projecte: 5320€.
- Costos d'amortització de material: 383€.

Total: 5.703€

Beneficis

- Detectar plagis.
- No només és útil per detectar plagis sinó que a més a més pot ser una eina molt útil per a les cerques bibliogràfiques.
- Empleats més autònoms i versàtils.
- Un cop funcionant és un servei que creix de forma exponencial.
- Servei aplicable a d'altres universitats així com empreses

Inconvenients

- Servei amb un temps d'implementació i anàlisi de l'aplicació.
- Recel per part dels alumnes envers l'eina.
- Necessitat mínima d'instal·lació del sistema.

PROJECTE VIABLE

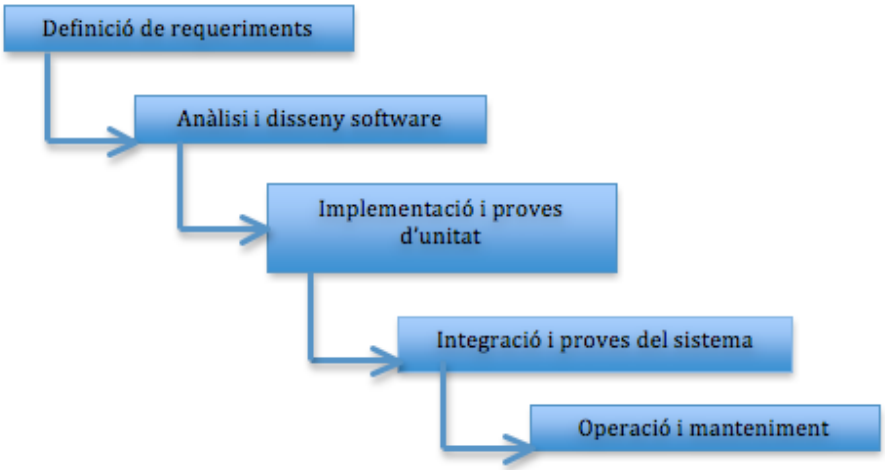
3 Disseny tècnic i implementació del sistema

3.1 Model de desenvolupament

Hem utilitzat un model de desenvolupament evolutiu. En aquest model, el software evoluciona amb el temps i els requisits de l'usuari i producte solen canviar conforme es desenvolupa el mateix. Aquest enfocament, entrellaça les activitats d'especificació, desenvolupament i validació, que ens porta a un sistema que es desenvolupa ràpidament a partir de les especificacions abstractes.

El projecte és ambiciós, compta de moltes parts i toca bastants llenguatges de programació, per tant l'èxit és el d'aconseguir un prototip capaç de resoldre en gran part els objectius proposats, amb el desavantatge de no poder obtenir un producte final degut al desenvolupament complex i innovador.

Sovint, cal desenvolupar primer un prototip que intenti ser capaç de portar a terme una idea innovadora (versió funcional limitada) abans d'arriscar-se a presentar una versió final el qual seria molt més costosa en cas de que resultés un projecte no viable.



Il·lustració 4: Model evolutiu

3.2 Entorn de desenvolupament

El llenguatge de programació base escollit per a l'entorn de desenvolupament del projecte és JAVA, i més concretament JAVA EE. S'ha triat aquest llenguatge de programació perquè és apte per a qualsevol plataforma ja que és interpretat, és avalat per múltiples empreses (com SUN, IBM, ORACLE), té una gran competitivitat i ens dóna moltes solucions lliures.

JAVA EE és una plataforma de programació d'Oracle Corporation per a desenvolupar i executar software escrit amb el llenguatge JAVA, amb una arquitectura distribuïda en nivells i basada en components de programari, tot plegat executant-se en un servidor d'aplicacions.

Un dels beneficis de JAVA EE com a plataforma, és la possibilitat de començar amb poc o relativament cap cost. La implementació és pot descarregar gratuïtament, i hi podem trobar moltes eines Open Source disponibles per a fer més extensa la plataforma o simplement per simplificar les nostres tasques en el desenvolupament.

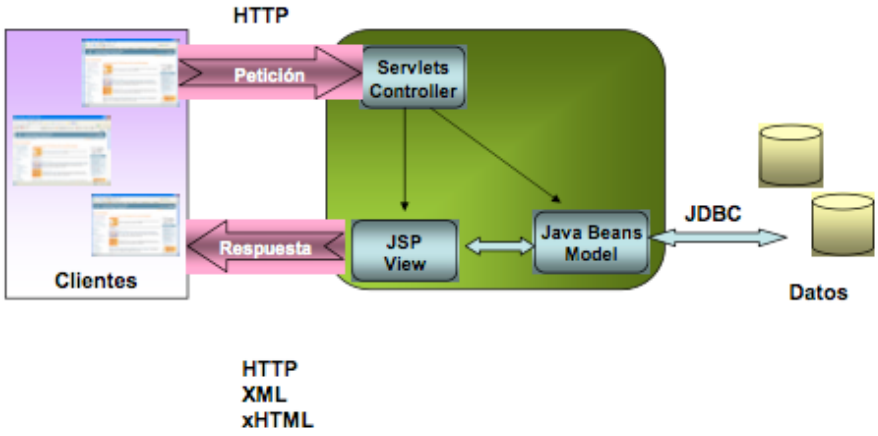
Com ja s'ha esmentat, JAVA EE és una plataforma que s'executa en un servidor d'aplicacions, en el nostre projecte ens hem decantat pel servidor JBOSS, un servidor Opens Source implementat en JAVA. A l'estar basat en JAVA, aquest pot ser utilitzat en qualsevol sistema operatiu.

L'aplicació ha estat desenvolupada seguint el paradigma Model Vista Controlador (MVC). Aquest és un patró d'arquitectura software que ens permet separar les dades i la lògica d'una aplicació de la interfície d'usuari i el mòdul encarregat de gestionar els esdeveniments i comunicacions.

El model és la representació de la informació amb la qual el sistema opera, per tant gestiona tots els accessos. Aquest envia a la vista aquella part de la informació que en cada moment es sol·licita per a que sigui mostrada. Si fem l'analogia amb el paradigma orientat a objecte seria l'equivalent a una classe. Aquests models es defineixen usant codi de la llibreria JAVA.

Les vistes, ens presenten el model en un format adequat per a interactuar. Aleshores podem tenir vàries vistes per un mateix model segons els perfils d'usuaris que tinguem al sistema i de les accions que es puguin aplicar sobre cada model. Són bàsicament codi xHTML que poden contenir accions, funcions i variables.

El controlador són bàsicament Servlets i és el responsable d'agafar les dades que ens interessen



Il·lustració 5: Model de programació web MVC

Per tant, s'adapta perfectament al model de desenvolupament ja que ens permet obtenir un pseudo-producte final ràpidament i poder modificar el que sigui necessari més tard sense perjudicar la resta ni haver d'aplicar grans canvis. Podem afegir com a argument extra per triar aquest entorn de desenvolupament el fet que en l'actualitat és el més usat dins del sector de desenvolupament web empresarial.

3.3 Disseny i implementació del sistema

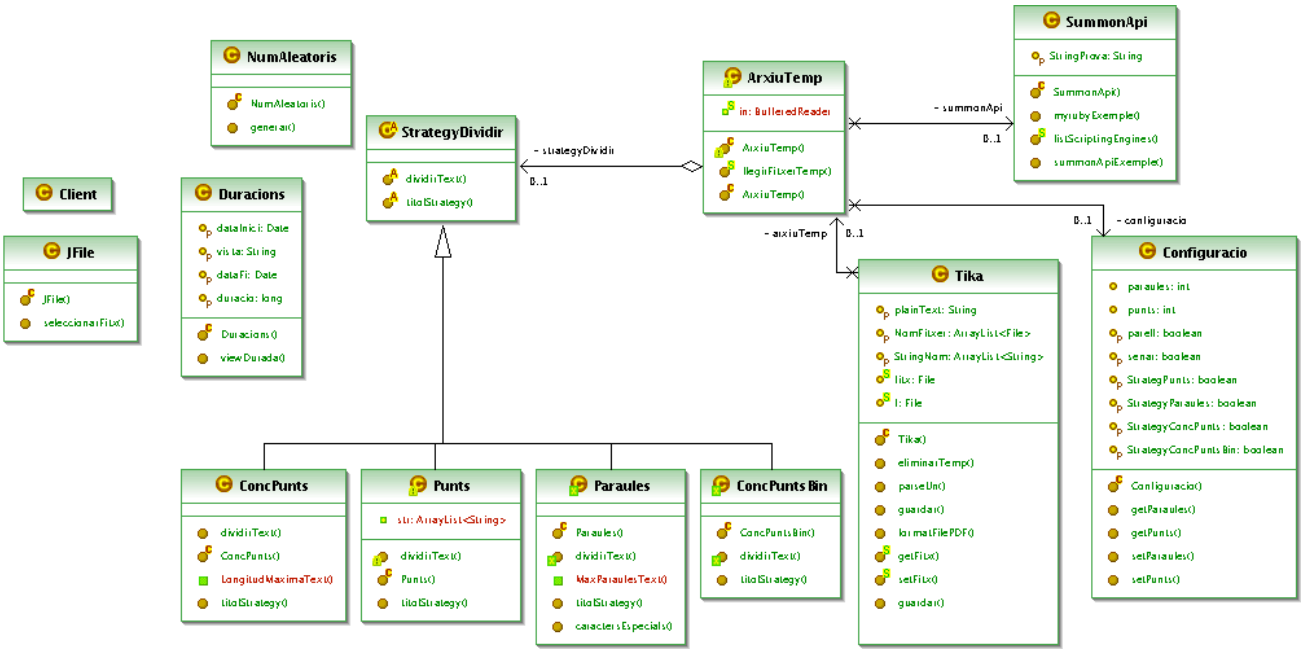
El projecte està dividit en dues parts ben diferenciades, el back-end i front-end. El back-end és l'esquelet de la nostra aplicació, conté les classes més primitives en JAVA així com també la connexió amb l'API Summon Service. L'altra part del projecte, ens acostarà més a la vista que tindrà el client / usuari final. Aquí entra en joc el servidor d'aplicacions JBOSS que ens permetrà interactuar entre les classes més primitives així com també la resposta de l'API en format JSON i mostrar-ho tot en un entorn HTML.

Un cop unides aquestes dues parts l'aplicació serà capaç de detectar plagis d'un fitxer, amb tota la corresponent informació, ja sigui l'autor, el nom del llibre i moltes altres opcions; suportat pel sistema prèviament seleccionat per l'usuari.

3.4 Back-end

Aquí s'exposarà el funcionament de l'aplicació a més baix nivell, podem dir que és el motor. Veurem com a partir d'un document qualsevol obtenim una resposta del servidor de Summon Service depenent de les estratègies proposades en el projecte, utilitzant el patró de disseny Strategy, que ens permetrà definir una sèrie d'algoritmes, encapsular-los i fer-los intercanviables.

En els següents punts analitzarem pas a pas cadascuna de les classes del nostre projecte.



Il·lustració 6: Diagrama de classes del back-end del projecte.

3.5 Framework Apache Tika

Un dels objectius proposats per al projecte i que a la vegada el feien un punt atractiu és el poder analitzar qualsevol document sense importar la seva extensió. Un dels punts forts, com ja hem esmentat en anterioritat, del llenguatge que utilitzem es que ens dona moltes solucions lliures; i Tika[3] ens ho proporciona de una forma molt senzilla. Aquest framework d'Apache ens permet detectar i extreure tant els metadata com el contingut d'un text estructurat utilitzant llibreries d'anàlisi existents.

La versió més recent de Tika, soporta:

- HyperText Markup Language (HTML).
- XML i derivats.
- Format de documents Microsoft Office.
- Format OpenDocument (.odf).
- Portable Document Format (.pdf).
- Electronic Publication Format (.epub).
- Rich Text Format (.rtf).
- Compression and packing formats (.zip).
- Format de text (.txt).
- Format d'audio (.mp3).
- Format d'imatge (.jpg).
- Format de vídeo (.flv).
- Classes de JAVA.
- Format mbox.

En el desenvolupament del nostre projecte ens hem centrat exclusivament en els formats de text pla (.txt), portable document format (.pdf) i els formats de Microsoft Office Word, tant per a les versions 97-2004 com posteriors.

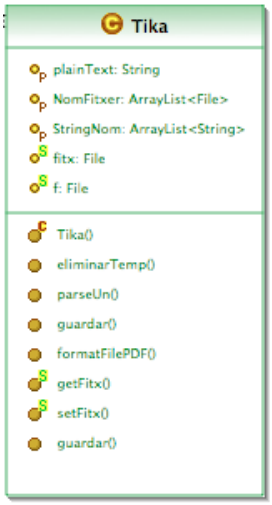
Per a utilitzar aquest framework en el nostre projecte serà necessari descarregar de manera gratuïta un fitxer, o més ben dit, una llibreria .jar de la pagina de Apache Tika.

[3] Més informació de Tika: http://tika.apache.org/

Un cop inclosa en la nostra classe JAVA, per poder utilitzar les classes que ens facilita Apache és tant senzill com afegir aquest imports:

```
import java.io.*;
import org.apache.tika.exception.TikaException;
import org.apache.tika.metadata.Metadata;
import org.apache.tika.parser.AutoDetectParser;
import org.apache.tika.parser.Parser;
import org.apache.tika.sax.BodyContentHandler;
import org.xml.sax.ContentHandler;
import org.xml.sax.SAXException;
```

Il·lustració 7: Imports del framework Tika



Podem comentar la funció de la nostra classe Tika . La principal funcionalitat és convertir els fitxers suportats pel projecte i guardar-ne un fitxer temporal en format de text pla per facilitar les operacions alhora d'aplicar l'algoritme de dividir el text.

L'encarregat de fer aquesta conversió, com es pot veure a l'imatge de l'esquerra, és el mètode parseUn, encarregat de extreure el metadata i el cos del text, i finalment cridant un altre mètode de la mateixa classe, guardant-ho en un fitxer.

Il·lustració 8: Classe Tika

```
ContentHandler contenthandler = (ContentHandler) new BodyContentHandler(-1);

Metadata metadata = new Metadata();

metadata.set(Metadata.RESOURCE_NAME_KEY, fil.getName());

Parser parser = new AutoDetectParser();
//                    OOXMLParser parser = new OOXMLParser();
//                    PDFParser parser3 = new PDFParser();
//                    TXTParser parser2 = new TXTParser();

parser.parse(is, contenthandler, metadata, new ParseContext());
//                    LOGGER.info("Mime: " +
metadata.get(Metadata.CONTENT_TYPE));
//                    LOGGER.info("Title: " + metadata.get(Metadata.TITLE));
//                    LOGGER.info("Author: " + metadata.get(Metadata.AUTHOR));
//                    LOGGER.info("content: " + contenthandler.toString());
this.guardar();
```

Aquest fragment de codi, ens permet obtenir el content type, el títol, l'autor i el contingut del document que esta analitzant el mètode parseUn, que és de tipus File.

El propi framework de Tika permet auto detectar quin tipus d'arxiu estem analitzant, gràcies al mètode AutoDetectParser, en cas de no funcionar, també et faciliten mètodes exclusius per a cada tipus d'arxiu suportat, com per exemple, PDFParser si l'arxiu és un Portable Document Format o TXTParser si el que estem analitzant és un text pla. Finalment es guarda l'arxiu ja transformat en text pla.

```
if(this.formatFilePDF(fil).equals("pdf")){
      LOGGER.info("Document pdf");
      StringBuffer buffer = new StringBuffer();
     BufferedReader in = ArxiuTemp.llegirFitxerTemp(new
File(this.getStringNom().listIterator().next()));
      String linea;
      while((linea = in.readLine()) != null){
            StringTokenizer st = new StringTokenizer(linea);
            while (st.hasMoreTokens()){
                  buffer.append(st.nextToken() + " ");
            }
      }
      this.guardar(buffer);
      }
```

Un cop guardat, es mira si el document que s'ha analitzat és un fitxer de tipus PDF , si es dona el cas, se li dona un format ja que els documents en format PDF un cop

transformats amb el Tika tenen un format que perjudicaria al bon funcionament de l'aplicació alhora de dividir el text. Això passa perquè a diferencia d'altres formats de text, el PDF és un format de tipus compost, està format per imatges vectorials, mapa de bits i text.

Aquesta modificació consisteix en analitzar línia a línia i col·locar tots els elements trobats un rere l'altre separats per un espai en blanc, per així després poder aplicar l'algoritme de separació de manera molt més eficient. Finalment es guarda i es reescriu el document creat anteriorment.

```
String format = "format correcte";
String f = fil.getName();
int j=0;
String[] list;
list = f.split("\\.");
while(j < list.length){
        if(list[j].equals("pdf")){
                format = list[j];
        }
        j++;
}
return format;
```

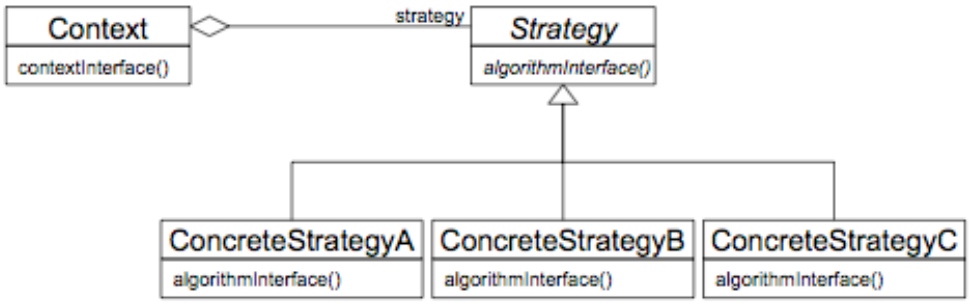
Aquest és el mètode per detectar si un arxiu és PDF, si és així retorna que és el format correcte.

3.6 Patró de disseny Strategy

El patró de disseny Strategy, com ja hem comentat anteriorment, ens permet definir una sèrie d'algoritmes, encapsular-los i fer-los intercanviables. Utilitzar adequadament patrons de disseny ens soluciona molt problemes de disseny. Aquest patró de disseny es aplicable a:

- Varies classes relacionades sols difereixen en el seu comportament. Strategy, permet configurar una classe amb un de entre varis comportaments.
- Es necessiten variants del mateix algoritme, que s'implementen com una jerarquia de classes.
- Un algoritme utilitza dades que els clients no tenen perquè conèixer.
- Una classe defineix molts comportaments que apareixen en sentències condicionals. Solució: moure els relacionats a un Strategy.

Per a que una solució sigui considerada un patró ha de posseir certes característiques. Una d'elles és que ha de ser efectiu resolent problemes similars. També ha de ser reutilitzable, el que significa que és aplicable a diferents problemes de disseny.

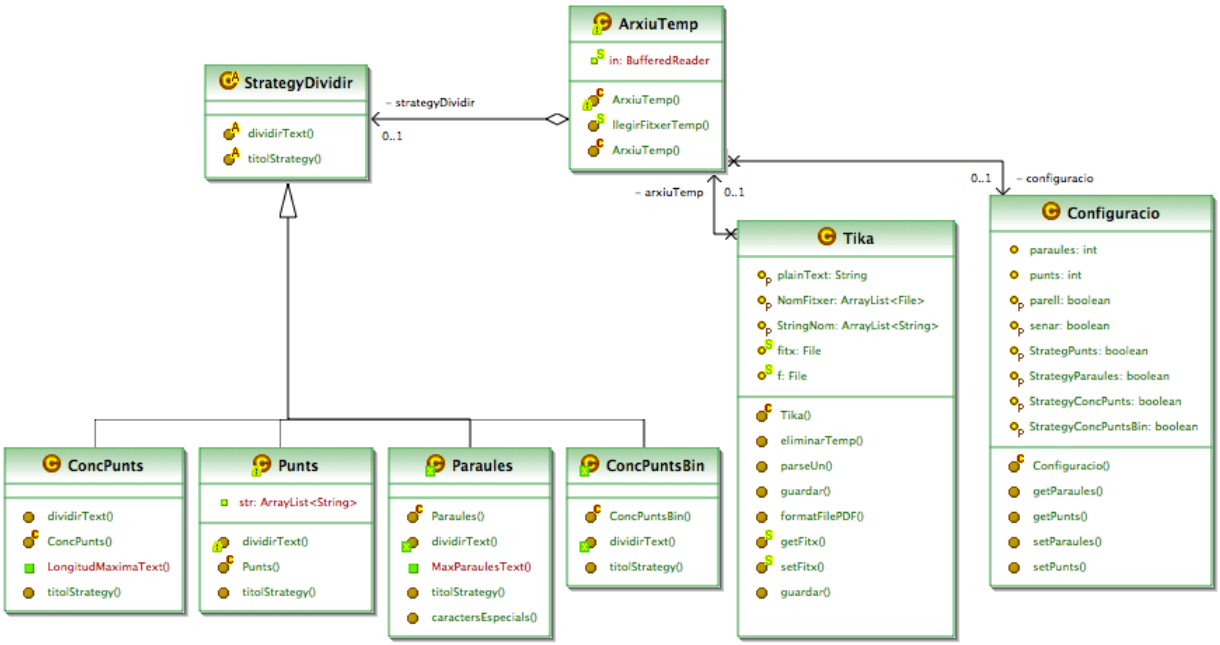


Il·lustració 9: Esquema del model de disseny Strategy.

Aquesta és l'estructura del patró Strategy. Com es pot veure, es diferencien tres parts:

- **Strategy**. Defineix una interfície comú per als algoritmes que suporta. Aquesta funció en el nostre projecte la realitza la classe StrategyDividir.
- **ConcreteStrategy**. Implementa un algoritme utilitzant l'interficie Strategy. Aquí trobarem les diferents estratègies que s'implementaran: Paraules, Punts, ConcPunts i ConcPuntsBin. Aquestes s'expliquen de forma detallada en l'apartat 3.6.2.
- **Context**. Està configurat com un objecte ConcreteStrategy, manté una referència a l'objecte Strategy i pot definir una interfície que permet a Strategy accedir a les dades. La classe ArxiuTemp ens permetrà realitzar aquestes tasques.

Així és com queda el nostre diagrama de classes després de dissenyar-lo utilitzant el patró Strategy.



Il·lustració 10: Patró de disseny Strategy aplicat al nostre projecte.

3.6.1 Strategy

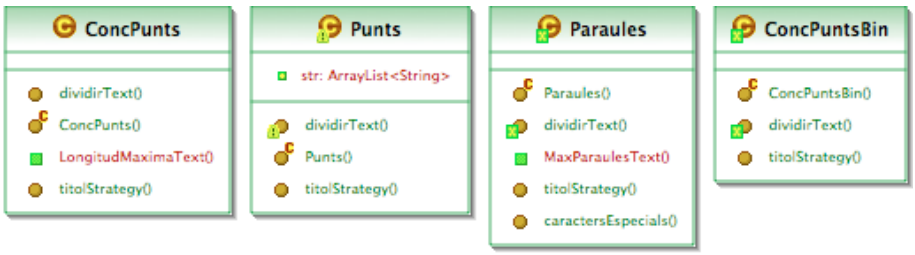
Il·lustració 11: Diagrama de classes de StrategyDividir.

En aquesta classe abstracta definim els mètodes i atributs que seran iguals per a totes les classes heretades. StrategyDividir sols crea els mètodes abstractes sense la seva implementació, serà a més baix nivell on es definirà aquest mètodes i on s'implementaran.

```
public abstract ArrayList<?> dividirText(File fil, Configuracio conf);
```

Com es pot veure, el mètode dividirText conté per paràmetre un fitxer, el que volem analitzar; i un paràmetre Configuració on com el seu nom indica contindrà diferents elements per a configurar l'algoritme. Aquest paràmetre s'exposa detalladament més endavant.

3.6.2 ConcretStrategy



Il·lustració 12: Diagrama de classes ConcretStrategy.

Cadascuna d'aquestes classes, implementa el mètode abstracte (dividirText) creat en el component Strategy del patró, en el nostre cas, la classe StrategyDividir. Ara passem analitzar aquestes classes per separat.

Punts

Aquesta estratègia divideix el text cada cop que es troba un punt (tant sigui punt i seguit com punt i a part) i en guarda el resultat en un ArrayList String. Està pensat

com l'estratègia estrella, ja que finalment aquests fragments de text dividits seran els que es buscaran si té alguna coincidència a Summon Service. Per tant, cercar cada punt ens garanteix que el que estem buscant té sentit sintàctic ja que és una frase o conjunt de frases que tenen cohesió.

Aquest és l'algoritme de dividirText de la classe Punts, per tallar les cadenes de caràcters del text s'utilitza la classe del framework de JAVA StringTokenizer.

```
try{
        BufferedReader in = ArxiuTemp.llegirFitxerTemp(fil);
        String linea;
        while((linea = in.readLine()) != null ){
                String delim = ".";
                StringTokenizer stoken = new StringTokenizer(linea,"\n");
                while(stoken.hasMoreTokens()){
                        str.add(stoken.nextToken(delim));
                }
        }
}catch (Exception e){
        e.printStackTrace();
}
return str;
```

```
private ArrayList<String> str = new ArrayList<String>();
```

La classe ArrayList permet l'emmagatzemament de dades en memòria de forma similar als Arrays convencionals, però amb el gran avantatge de que el número d'elements que pot guardar és dinàmic. Per tant, podem afegir elements i accedir de forma quasi immediata a qualsevol índex de l'ArrayList.

Paraules

El mètode dividirText de la classe Paraules permet dividir el text segons les paraules que determini l'usuari. Aquí és on entra en joc la classe Configuració, mitjançant el constructor de la pròpia classe podem seleccionar el número pel qual volem tallar un text. Esta pensat per fer una cerca més tard a Summon Service amb un temps de resposta molt menor que si s'utilitzés l'estratègia de Punts. Si el temps és un avantatge, la cohesió és un inconvenient. Al deixar a l'usuari seleccionar el número de paraules es possible que el conjunt dividit no tingui cap sentit i alhora de buscar a l'API el resultat no sigui del tot esperat.

```
public Configuracio(int Paraules, int Punts, boolean Senar, boolean Parell){
                this.setParaules(Paraules);
                this.setPunts(Punts);
                this.setSenar(Senar);
                this.setParell(Parell);
        }
```

El constructor conté totes les possibles configuracions, en el cas que estem analitzant, de l'estratègia paraules serà un número enter. Aquest objecte de tipus Configuració és el que s'especifica al mètode dividirText. Pràcticament és el mateix algoritme que s'utilitza per a la classe Punts amb algunes petites modificacions.

```
conf.getParaules() > this.MaxParaulesText(fil)
```

Aquesta expressió ens permet comprovar si el número pel qual l'usuari vol dividir el text, no és superior al número màxim de paraules que té el propi text. El mètode MaxParaulesText és un simple comptador que cada cop que llegim una paraula va augmentant i finalment ens retorna el valor d'aquest.

```
par == this.caractersEspecials(par)
```

Un altra diferència és especificar el que és una paraula del que és un caràcter especial. La classe StringTokenizer del framework de JAVA que utilitzem per dividir el text no ho detecta, creu que aquest signes son paraules i fa que el funcionament esperat de l'aplicació no sigui el correcte. Per aquest motiu hem creat el mètode caractersEspecials per poder solucionar aquest problema.

ConcPunts

A l'igual que la classe Paraules, ConcPunts depèn de la classe Configuració. L'usuari determina el número de punts pels quals ha de tallar l'algoritme, és a dir, si l'usuari selecciona tres punts, l'algoritme ignorarà el primer i segon punts i és quan trobi el tercer punt, dividirà el text. En cas de que l'última divisió sigui menor que el nombre establert per l'usuari, igualment es mostrarà. Mentre l'algoritme no hagi arribat al número establert per l'usuari, el contingut dels anteriors punts s'emmagatzemen en un Buffer, un espai temporal fins a llegir l'esperat i es guarden en un ArrayList String. Té un resultat esperat semblant a l'estrategia de Punts però amb un temps molt menor ja que reduïm el número de posicions.

```
if(contador < conf.getPunts()){
        par = stoken.nextToken(delim);
        res.append(par + ".");
        contador++;
} else if (contador == conf.getPunts()){
        str.add(res);
        res = new StringBuffer();
        contador = 0;
}
```

També es comprova que el número seleccionat per l'usuari no superi el màxim de número de punts que conte el text. LongitudMaximaText és un comptador que analitza tot el document en busca de punts i en retorna el número.

```
conf.getPunts() > this.LongitudMaximaText(fil)
```

ConcPuntsBin

Aquesta estratègia té el mateix funcionament que la de Punts en l'excepció que podem determinar els elements de l'ArrayList segons si són parells o senars. Aquesta estratègia es força útil si el que es vol és reduir el temps de resposta en el servidor Summon Service ja que haurà d'avaluar menys posicions de l'ArrayList.

```
if(conf.isParell()){
        for(int x=0;x<str_parell.size();x++){
                str_solucio.add(str_parell.get(x).toString());
        }
}else if(conf.isSenar()){
        for(int x=0;x<str_senar.size();x++){
                str_solucio.add(str_senar.get(x).toString());
        }
}
```

Segons el que seleccioni l'usuari, si vol dividir i guardar les posicions parells o les senars segons el número de punts en el que es vol dividir el text.

3.6.3 Context

ArxiuTemp manté una referència a l'objecte StrategyDividir i així poder accedir a les dades.

```
public ArrayList aplicarEstrategia(StrategyDividir strategy, File fil, Configuracio conf){
        this.setStrategyDividir(strategy);
        ArrayList<String> sol = (ArrayList<String>) this.getStrategyDividir().dividirText(fil,conf);
        return sol;
}
```

La classe ArxiuTemp és l'encarregada de muntar tota l'estructura del patró Strategy. AplicarEstrategia és un ArrayList que conté el resultant d'aplicar l'estratègia a seguir i la seva configuració corresponent.

3.7 Summon Service

Com ja hem comentat en anterioritat, l'eina encarregada de detectar si un fragment del text que volem analitzar conté elements plagiats és Summon Service. La connexió entre el nostre projecte i l'API de Summon Service es realitza mitjançant comunicació HTTP com s'explica en el seu web (http://api.summon.serialssolutions.com/).

Per realitza aquesta comunicació és necessari obtenir cinc components, els quals després seran la capçalera de la connexió. Necessitem per tant:

- **Accept header value**. Aquest component és el content-type, que determina com serà la resposta de la petició, application/xml o application/json. En el nostre projecte la resposta es modelarà amb JSON.
- **X-Summon-date header value.** Conté la data en que s'envia la petició. Aquesta ha de ser del tipus "EEE, d MMM yyyy HH:mm:ss Z" (un exemple seria: Tue, 30 Jun 2013 12:10:24 GMT).

- **Host header value.** El host sempre serà el mateix, "api.summon.serialssolutions.com".
- **Path portion of the resource URI.** El path com el host no canvia, "/2.0.0/search".
- **Sorted query string unencoded.** Aquest component inclou els elements que volem buscar a Summon Service.

Un cop obtinguts tots el components necessaris s'aplica el mètode que ens proporciona Summon Service computeIdString, aquest mètode genera el format acceptat per l'API. Generat aquest, necessitem una clau secreta que juntament amb el computeIdString s'envia a Summon Service per rebre la resposta. Per obtenir la clau secreta, també se'ns proporciona un mètode, aquest esta format per una clau. La clau ens ha sigut proporcionada per la Universitat Oberta de Catalunya (UOC). L'altre component de la clau secreta és la cadena de caràcters adquirida de l'anterior mètode, computeIdString. El resultat serà d'aquest tipus:

```
rYiYzRaaZ9/QYpiQFZADqpkgJfM=
```

L'últim pas abans de rebre una resposta, és enviar una autorització al servidor estructura de la qual esta fixada. El format serà una cadena de caràcters que conté la paraula Summon seguit d'un espai en blanc, l'accesId també proporcionat per la UOC, punt i coma; i per finalitzar el resultat d'aplicar el computeIdString. Aquí es pot veure un exemple:

```
Summon test;rYiYzRaaZ9/QYpiQFZADqpkgJfM=
```

Aquest mètode per obtenir la resposta del servidor no s'ha pogut assolir en el projecte, ja sigui per la falta de documentació de la pròpia API com altres elements. Com alternativa es va proposar realitzar la petició a través de Ruby on Rails. El propi Summon Service proporciona les llibreries per desenvolupar tant en PHP com en Ruby (http://api.summon.serialssolutions.com/help/api/code).

El projecte no podia canviar el llenguatge de programació i ja estava en una etapa avançada, un canvi d'aquest tipus suposaria un impacte crític. Per tant, ens vam decantar per utilitzar les llibreries que proporciona l'API amb el llenguatge Ruby, però continuant treballant amb el mateix llenguatge i entorn de programació fins al moment. Això ha estat possible gràcies al framework JRuby[4].

JRuby és una implementació en JAVA del llenguatge de programació Ruby. Com qualsevol altre framework hem d'importar les llibreries necessàries per poder utilitzar-lo en la nostra aplicació.

Un cop instal·lat tot el requerit, per utilitzar aquest motor és necessari seleccionar-ho mitjançant ScriptEngine de JAVA.

```
ScriptEngine jruby = new ScriptEngineManager().getEngineByName("jruby");
```

Aquest fragment declara un objecte que serà de tipus JRuby. Un cop declarat tant sols queda definir l'arxiu Ruby que es vol avaluar i per tant utilitzar en el nostre entorn JAVA.

```
Reader reader = new FileReader(new File("summon.rb"));

jruby.eval(reader);
```

L'arxiu summon.rb conté el codi per fer la comunicació amb l'API. Per utilitzar aquest arxiu és necessari instal·lar les llibreries de Ruby i importar el projecte Summon que se'ns proporciona (https://github.com/summon/summon.rb).

Es va poder comprovar que JRuby funciona perfectament si el que estem avaluant és un arxiu simple Ruby, és a dir si tenim un sol fitxer amb mètodes sense cap dependència externa. Però utilitzar aquest motor juntament amb les llibreries de Summon Services en Ruby, aquest no funciona degut a les múltiples dependències externes.

[4] Més informació: http://jruby.org/

Aquest també va ser un punt crític en el desenvolupament del projecte. Com a solució es va proposar realitzar la connexió inicialment pensada en JAVA, però enlloc de realitzar-la directament sobre el propi Summon Service, s'utilitza un pont.

Degut als constants problemes per accedir a l'API es va presentar la solució gràcies a l'enginyer Xavi Duran, treballador de la UOC. Ens va cedir el seu projecte realitzat íntegrament amb Ruby i que realitzava satisfactòriament la connexió amb SummonService, per tant, parlem d'un pont. Es realitza la connexió inicialment exposada per HTTP però aquest cop es realitza al servidor Apila, el projecte cedit, i és aquest últim el que efectua la petició final via Ruby.

Per poder utilitzar Apila és necessari instal·lar el servidor.

```
git clone git://github.com/xdurana/apila.git
cd apila
bundle install
```

Un cop instal·lat tenim que configurar l'usuari i contrasenya, que en anterioritat s'ha comentat, proporcionat per la UOC.

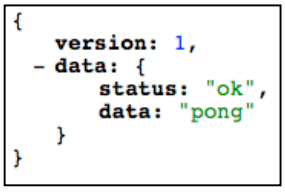
```
cp config/summon.example.yml config/summon.yml
subl config/summon.yml
```

Finalment resta executar el servidor.

```
shotgun
```

Perquè funcioni la nostra aplicació, aquest servidor ha d'estar en funcionament, sinó no es podrà realitzar la connexió a tres bandes. Per saber si esta en funcionament podem realitzar la següent prova:

```
http://localhost:9393/api/v1/status/ping
```



Il·lustració 13: Resposta de l'API si esta en funcionament.

Aquesta és la resposta esperada si el servidor està executat.

Si inicialment la connexió HTTP es realitza a

"api.summon.serialssolutions.com", aquest cop s'executa a "http://localhost:9393/api/v1/summon/", seguit del que volem cercar, en el nostre cas el contingut de l'ArrayList un cop aplicada l'estratègia corresponent.

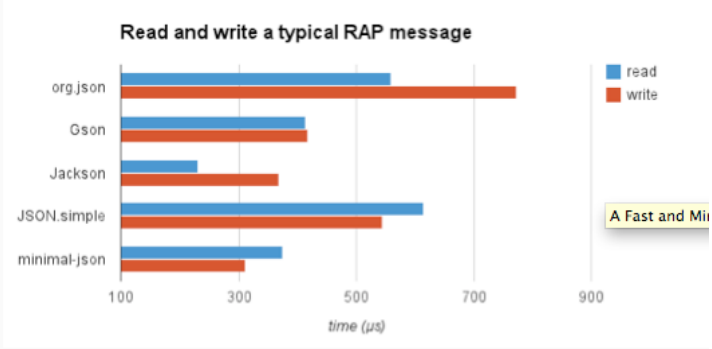
```
URL url = new URL(URL_API + METHOD_SEARCH +URLEncoder.encode(str.get(z).toString(),"UTF-8"));
```

Si s'ha detectat un possible plagi, el resultat serà un objecte JSON semblant al següent.


```
{
   version: 1,
 - data: [
    - {
       - authors: [
          - {
                fullname: "Haigh, John",
                surname: "Haigh",
                givenname: "John",
                middlename: null
            }
        ],
        title: "<h>Probability</h> <h>Models</h>",
       - subject: [
            "Operation Research/Decision Theory",
            "Simulation and Modeling",
            "Mathematical Applications in Computer Science",
            "Mathematical Applications in the Physical Sciences",
            "Probability and Statistics in Computer Science",
            "Mathematics",
            "Probability Theory and Stochastic Processes"
        ]
    },
```


Il·lustració 14: Resposta de l'API si a trobat coincidència.

Per extreure aquesta informació és necessari crear un objecte de tipus JSON en el nostre entorn de desenvolupament. S'ha seleccionat l'estructura de org.json[5] ja que segons benchmarks realitzats entre les estructures més important, és la que dóna un rendiment més elevat, com es pot apreciar en el gràfic següent (http://eclipsesource.com/blogs/2013/04/18/minimal-json-parser-for-java/),.



Il·lustració 15: Benchmark de les principals estructures JSON.

[5] Més informació: http://www.json.org/java/index.html

```
JSONObject json = new JSONObject(sb.toString());
```

Creat aquest objecte, podem recórrer i extreure els elements d'aquesta estructura. Inicialment el nostre projecte un cop detecti plagi ens mostrarà l'autor/s, el títol/s i un conjunt de paraules clau que defineixen el contingut del llibre. L'API ens proporciona molta més informació però per al nostre projecte aquest tres components són suficients, encara que en futures línies de treball no es descarta afegir més informació.

Aquest tres component com a resposta són extrets i emmagatzemats en diferents estructures dinàmiques pensat en un futur alhora de realitzar la visualització en l'entorn web, molt més ràpid i senzill. El projecte ha estat configurat per a que detecti que un fragment de text conté plagi si, el nombre de llibre detectat a la resposta és igual o inferior a 5. Més d'aquest número pensem que no és una informació rellevant per determinar que allò està plagiat.

3.8 Front-end

Acabat el nostre esquelet o back-end, només resta plasmar-ho en un entorn web facilitant a l'usuari final la seva utilització. Per fer possible aquesta comunicació entre les nostres classes més primàries desenvolupades en JAVA i finalment mostrar el resultat en HTML, utilitzarem el servidor d'aplicacions JBoss. En el desenvolupament de la nostra aplicació no hem disposat d'un servidor, per tant, hem integrat aquest servidor d'aplicacions en la nostra eina de desenvolupament (Eclipse JAVA EE) i transformar-ho també amb un servidor.

Instal·lat el servidor JBoss en el nostre entorn de desenvolupament, cal tenir en compte l'estructura de directoris que ens proporciona el propi servidor.

- **Java Resource.** Conté totes les classes JAVA exposades en el back-end i les noves creades en el procés de desenvolupament web.

- **WebContent.** Conté tot allò relacionat en el desenvolupament web, CSS, imatges, vídeos, arxius HTML, JSP, JAVASCRIPT, etc. Dins d'aquesta trobem també WEB-INF que conté dos elements, una carpeta lib. Aquí hem de col·locar tots els JAR's que utilitza la nostra aplicació. I un arxiu web.xml on es definiran diferents aspectes del servidor com els servlets creats, temps màxim de resposta, listeners, etc.

Per a la creació de l'entorn web s'ha utilitzat com a motor principal HTML5[6] i els frameworks de JAVASCRIPT, JQuery[7] i Prototype[8]; i XMLHTTPRequest[9] (XHR).

[6] HTML5 és la cinquena revisió del llenguatge per a l'elaboració de pàgines web. Més informació: http://www.w3.org/html/wg/drafts/html/master/

[7] JQuery és una biblioteca de JavasScript que permet simplificar la manera d'interactuar amb els documents HTML. Més infomació: http://jquery.com/

[8] Prototype és un framework escrit en JavaScript orientat al desenvolupament senzill i dinàmic d'aplicacions web. Més informació: http://prototypejs.org/

[9] XHR és una interfície usada per a realitzar peticions HTTP i HTTPS a servidors web. L'ús més popular, és el de proporcionar contingut dinàmic i actualitzacions asíncrones en pàgines web.

3.8.1 Disseny del web

La decisió d'utilitzar un entorn web sorgeix de donar una facilitat extra a l'usuari. És a dir, oferir un sistema integral i distribuït que no requereix de cap instal·lació ni configuració d'equips. S'ha dissenyat un entorn simple, intuïtiu i minimalista per facilitar-ne la navegació.

Il·lustració 16: Pantalla d'inici de l'aplicació.

La imatge superior mostra la pàgina d'inici de la nostra aplicació, formada per tres parts diferenciades.

- **TextPlagi.** El títol diu molt d'un producte i en aquest cas és curt, clar i directe.
- **Component caixa.** Conscients de la gran penetració del HTML5, la decisió ha esta utilitzar-ho per mostrar el seu gran potencial, utilitzant una funció tant de moda com és el "drag and drop", usat per aplicacions tant destacades com Gmail i Dropbox entre d'altres.
- **Footer.** Informació al peu de pàgina.

3.9 Upload File

L'usuari, com hem exposat, gràcies a la funció "drag and drop", podrà arrossegar dins la caixa l'arxiu i automàticament penjar-lo al servidor. Aquest procés es porta a terme gràcies als Servlets que podem implementar en el servidor d'aplicacions JBoss.

Els Servlets és una tecnologia que ens permet crear aplicacions web interactives, és a dir, dinàmiques. Són petits programes escrits en JAVA que resolen peticions a través del protocol HTTP. Aquests reben instàncies des d'un navegador web, les processen i en retornen una resposta. En el nostre projecte la resposta és HTML5.

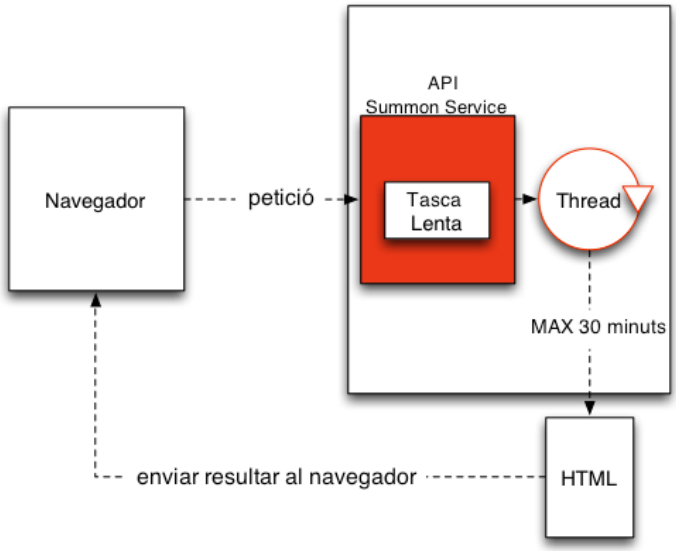
Un cop l'usuari deixi anar l'arxiu a analitzar sobre la caixa, s'executa un codi en XHR, aquesta interfície ens proporciona contingut dinàmic i el més important, contingut asíncron. Un procés asíncron es caracteritza pel fet que qui envia la petició, continua en el seu estat d'execució immediatament després d'enviar el missatge al receptor.

```
// now post a new XHR request
  if (tests.formdata) {
   var xhr = new XMLHttpRequest();
   xhr.open('POST', 'FileUploadAsync',true);
   xhr.onload = function() {
    progress.value = progress.innerHTML = 100;
   };
   if (tests.progress) {
    xhr.upload.onprogress = function (event) {
     if (event.lengthComputable) {
      var complete = (event.loaded / event.total * 100 | 0);
      progress.value = progress.innerHTML = complete;
     }
    };
   }
   xhr.send(formData);
```

Com es pot veure en el fragment de codi, és crea un objecte de tipus XMLHttpRequest i enviem la informació, en aquest cas l'arxiu. Aquest és enviat de forma asíncrona al Servlet FileUploadAsync sol·licitat mitjançant Post.

```
xhr.open('POST', 'FileUploadAsync',true);
```

Un cop enviada la petició, com es realitza de forma asíncrona, l'usuari pot seguir navegant i utilitzant l'aplicació sense que aquesta es bloquegi esperant que l'arxiu es pengi al servidor. El Servlet encarregat de penjar-ho al servidor utilitza la llibreria de Apache fileupload i a l'igual que la informació que se li envia, el Servlet també treballa de forma asíncrona, gràcies a l'especificació Servlet 3.0.



Il·lustració 17: Esquema que mostra el funcionament dels Servlets 3.0 aplicat al nostre projecte.

El funcionament és el següent: tenim una operació lenta que no sabem el temps que pot tardar; el servidor passat un temps cancel·la l'operació. Això és crític, perjudicaria el funcionament correcte de l'aplicació. Solució, aquesta operació que resulta lenta, es fa en un thread a banda.

```
final AsyncContext async = request.startAsync();
async.start(new Runnable() {
        @Override
        public void run() {
        try {
        uploadServerFile(request);
        } catch (IOException e) {
                e.printStackTrace();
        }
        async.complete();
        }
```

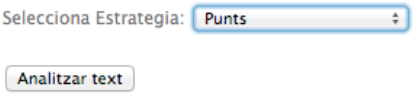
Com es pot veure, l'operació lenta uploadServerFile és realitza en un thread a part, un cop acabada l'operació ho comunica. I ja tenim el fitxer que l'usuari a seleccionat, penjat a la carpeta que hem seleccionat al mètode uploadServerFile que volem que es guardi. Al mateix temps que s'aplicava aquest mètode hem creat una variable session per poder gestionar els documents, amb un identificador únic.

```
HttpSession session = request.getSession(true);
```

3.9.1 Aplicar estratègies

Amb l'arxiu ja penjat al servidor, el següent pas és seleccionar el tipus d'estratègia que és vol aplicar per dividir el text.

Il·lustració 18: Vista un cop l'arxiu s'ha penjat al servidor.



Il·lustració 19: Vista de les opcions per analitzar els arxius.

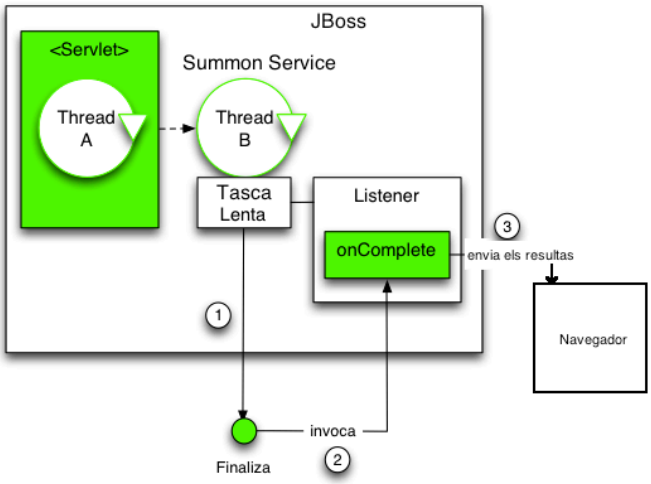
La connexió entre estratègies, arxiu penjat i la resposta web, s'utilitza XHR que envia la informació al Servlet estAsync. Com el seu nom indica, aquest també s'executa de forma asíncrona. L'aplicació està preparada per a suportar l'anàlisi d'un arxiu sempre i quan no superi un temps de trenta minuts. Aquest Servlet és important ja que serà l'encarregat d'executar la classe que cerca a l'API Summon Service i un cop finalitzat generar la resposta visual per a l'entorn web.

Quan l'usuari polsa "Analitzar text", es recull el valor de l'estratègia seleccionada utilitzant JQuery, en el cas de la imatge el seu valor serà "punts" i s'envia mitjançant XHR al Servlet.

```
var estrategia = $j("#opciones").val();
        if(estrategia == 'punts'){
                xmlhttp.open("POST",'estAsync?estrategia='+estrategia,true);
                xmlhttp.send();
        }
```

Il·lustració 20: Menu d'opcions de l'estratègia Paraules.

Si el seleccionat és l'estratègia de Paraules, es pot veure que l'aplicació mostra un camp més, referent al número de paraules. Igual en la resta d'estratègies.



Il·lustració 21: Esquema del funcionament del Servlets 3.0 funtament amb un Listener aplicat al nostre projecte.

Enviada la informació al Servlet aquest executarà l'operació lenta d'enviar les nostres peticions a Summon Service. Afegint un listener al nostre Servlet asíncron aconseguim que un cop hagi acabat l'operació lenta, automàticament realitzi una operació. Aquest operació serà la resposta i mentre l'aplicació recull informació de l'API, al web es mostrarà una imatge animada, més conegut com gif. Així aconseguim que l'usuari vegi que s'està treballant amb la seva petició i no s'ha cancel·lat per qualsevol causa, ja que si fos així es mostraria un text indicant l'error trobat.

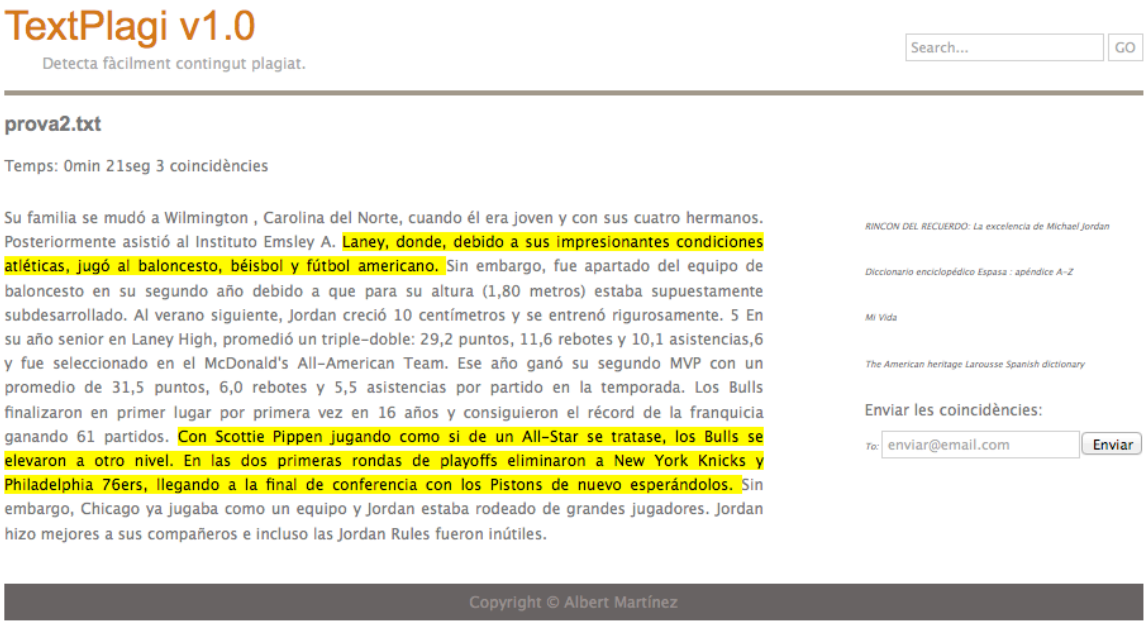
Il·lustració 22: Vista de l'aplicació mentre esta analitzant en busca de coincidències.

3.9.2 Visualització dels resultats

Quan es tenen totes les peticions finalitzades, extretes, i en el cas de que s'hagi trobat plagi, es guardaran en estructures individuals i s'executarà el listener, encarregat de crear l'estructura web que es mostrarà a l'aplicació. El codi col·locat en aquest listener serà el que recollirà XHR i mostrarà el resultat.

```
xmlhttp.onreadystatechange=function(){
        if (xmlhttp.readyState==4 && xmlhttp.status==200){
        pageContents = xmlhttp.responseText;
        document.getElementById("vfile").innerHTML = pageContents;
```

Onreadystatechange és un esdeveniment predeterminat de XHR que es dispara quan hi ha un canvi d'estat.



Il·lustració 23: Vista un cop l'aplicació acabat d'analitzar i a trobat coincidències.

El resultat és bolcat al contenidor central, mantenint l'entorn simple, intuïtiu i minimalista d'un bon inici. Aquest resultat és pot dividir en cinc parts. El primer element que se'ns mostra és el títol del document que hem analitzat. Seguit del temps que s'ha tardat en buscar i mostrar les coincidències i el número d'aquestes.

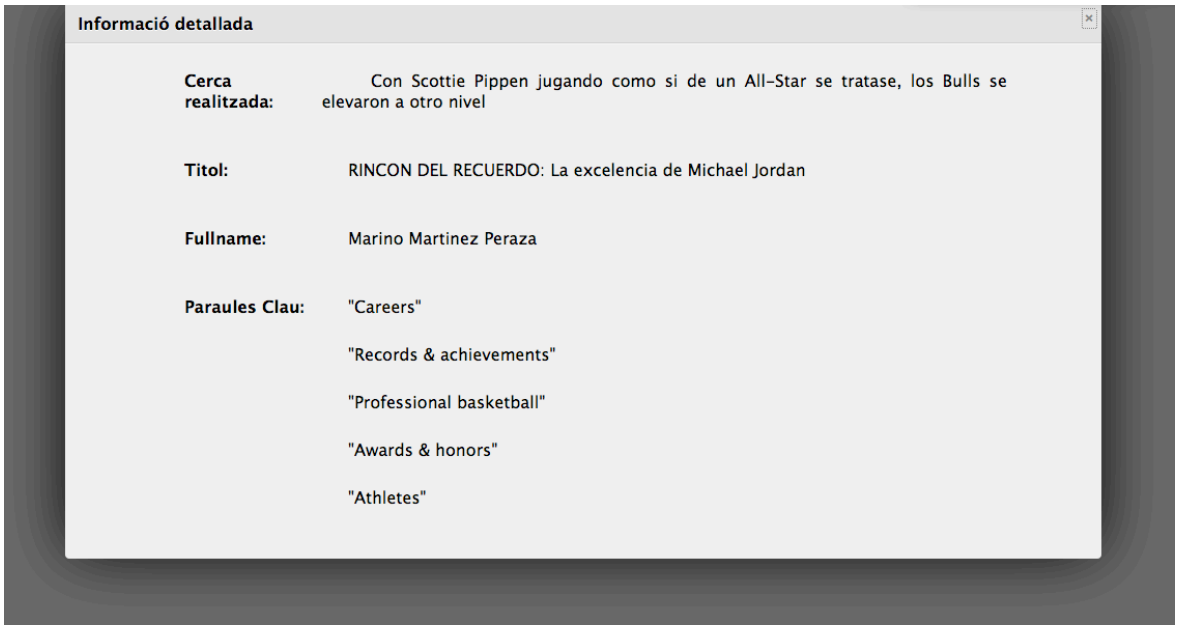
El sistema crea un objecte de tipus Date de la classe JAVA, en el moment que s'executa el Servlet estAsync, encarregat d'enviar les peticions a Summon Service i es repeteix quan acaba l'operació en el listener obtenint la durada total d'aquesta.

L'altra part característica és el contingut del document. Aquest, si conté parts plagiades es ressalten per una millor visualització. Aquest apartat s'aconsegueix comparant el text original amb les cerques on hem obtingut plagi. En cas que així sigui, aquestes se'ls hi aplica l'estil de subratllat de color groc. Vist així, el document no ens mostra cap tipus d'informació sobre el plagi, només visualitzem les parts on ha trobat una possible còpia.

En un principi, l'usuari quan polsava damunt el plagi el navegador el redirigia a una altra pagina de l'aplicació on es mostrava la informació més detallada. Aquesta informació era obtinguda a través del Servlet Search. Aquest mètode va ser suprimit en veure que no era molt còmode per l'usuari tenir que navegar entre pestanyes, amb l'inconvenient afegit que alguns navegadors al polsar endarrere tornaven a l'índex de l'aplicació, la qual cosa perjudicava el bon funcionament.

La solució trobada ha estat aplicar ModalBox[10]. Aquesta eina ens proporciona un contenidor per visualitzar els resultats sense moure's de la pròpia pantalla. Per utilitzar aquesta eina, hem de descarregar de la pàgina i instal·lar-ho en la nostra aplicació.

[10] Més informació: http://okonet.ru/projects/modalbox/index.html



Il·lustració 24: Vista detallada d'una coincidència utilitzant ModalBox.

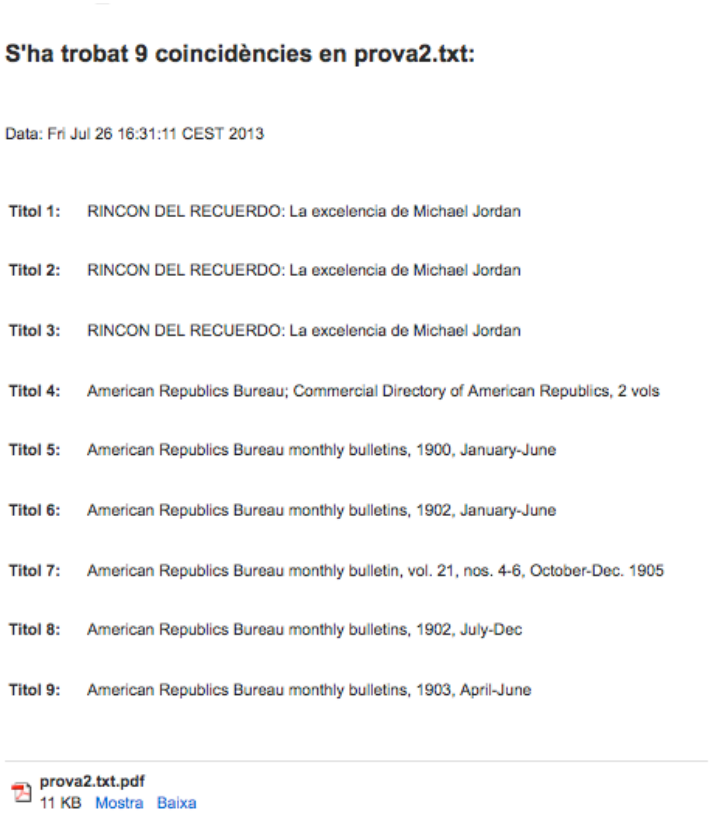
Així mantenim el disseny simple que tant ha caracteritzat aquesta aplicació.

L'altra part que podem diferenciar és el llistat de la dreta. Aquest llistat, conté els títols que ha trobat sense eliminar els elements repetits. És a dir, quan rebem la resposta de Summon Service l'aplicació te *n* llibres, molts d'aquest repetits. Per visualitzar-los en aquest apartat hem fet ús de l'estructura de dades linkedHashSet.

Aquesta estructura és molt útil i el seu funcionament també, bolquem les dades que tenim guardades dels títols a l'estructura linkedHashSet. Quan les extraiem d'aquesta, ja ha eliminat tots els elements duplicats. Per tant, aquest llistat amb els títols ens dona informació a primer cop d'ull del que ha trobat l'aplicació.

L'última part de la visualització dels resultats és la caixeta de correu electrònic. Aquesta permet a l'aplicació enviar els resultats i adjuntant un pdf amb un resum de tot el plagi detectat al document.

Per enviar el correu i adjuntar un fitxer s'ha fet ús del framework de JAVA JavaMail, aquest ens permet introduir un servidor de correu per enviar-los.


S'ha trobat 9 coincidències en prova2.txt:

Data: Fri Jul 26 16:31:11 CEST 2013

Titol 1: RINCON DEL RECUERDO: La excelencia de Michael Jordan

Titol 2: RINCON DEL RECUERDO: La excelencia de Michael Jordan

Titol 3: RINCON DEL RECUERDO: La excelencia de Michael Jordan

Titol 4: American Republics Bureau; Commercial Directory of American Republics, 2 vols

Titol 5: American Republics Bureau monthly bulletins, 1900, January-June

Titol 6: American Republics Bureau monthly bulletins, 1902, January-June

Titol 7: American Republics Bureau monthly bulletin, vol. 21, nos. 4-6, October-Dec. 1905

Titol 8: American Republics Bureau monthly bulletins, 1902, July-Dec

Titol 9: American Republics Bureau monthly bulletins, 1903, April-June

prova2.txt.pdf
11 KB Mostra Baixa

Il·lustració 25: Vista del correu plantilla que s'envia.

El correu generat és una plantilla formada per quatre elements. El número de coincidències total de l'arxiu, l'hora en que s'ha buscat la informació per aquest fitxer, el llistat de llibres que s'han trobat eliminant els possibles duplicats i finalment l'arxiu adjunt que és un pdf.

Per generar l'informe en format pdf, s'ha utilitzat la llibreria iText[11] que ens permet generar documents pdf utilitzant el propi llenguatge de JAVA, ja sigui text, taules, gràfics, etc.

3.10 Seguretat i privadesa

L'aplicació implementa dos elements importants pel funcionament però aquest no els veu l'usuari, estem parlant d'un ServletListener i un Servlet Index. Aquest dos, diguem-ne Servlets neixen de la solució a dos problemes. El primer problema que

[11] Més informació: http://itextpdf.com/

se'ns planeja es quan ens preguntem que fem amb els documents pujats al servidor i altres documents generats de forma temporal. La solució és utilitzar un Servlet que implementi un ServletContextListener, aquest funciona com un listener.

```
timer = new Timer();
timer.schedule(this,0,30*60*1000);//Cada 30 minuts
```

Seleccionem el temps cada quan es vol que s'executi aquest Servlet, com es pot comprovar, s'executarà cada trenta minuts. Aquest servlet està en continu procés i en el moment que es posa en marxa el servidor comença a comptar. Comprova els elements guardats en el servidor i si porten més temps que l'establert, s'eliminen.

L'altre problema plantejat és referent a les sessions. Com s'ha comentat, l'aplicació utilitza les sessions de JAVA per garantir a l'usuari un identificador únic per sessió. Però, el problema es presenta quan volem destruir la informació d'aquest usuari. En quin moment ho fem: un cop hem analitzat l'arxiu? Un cop hem visualitzat els resultats? La solució més adient per a la nostra aplicació és destruir tota la sessió un cop es carregui la pagina principal (índex) o en qualsevol moment en que es polsi "TextPlagi", ja que aquest executa el Servlet Index que implementa la destrucció de la sessió. Així aconseguim que cada cop que l'usuari introdueixi un fitxer nou a analitzar aquest obtindrà una sessió nova i única.

4 Fase de proves

S'han aplicat un criteris de tests basats en la funcionalitat i la seguretat del sistema. Bàsicament proves d'unitat mòdul a mòdul per comprovar de forma independent si cadascun d'ells funciona correctament.

Paral·lelament, cada cop que un mòdul superava correctament les proves d'unitat, s'ha anat agrupant en el producte final per aplicar proves d'integració i acceptació per part de l'equip de treball.

4.1 Proves d'unitat Framework Tika

A la següent taula tenim les proves principals a les quals hem sotmès l'eina per transformar els fitxer Tika.

	Requeriment	Resultat esperat	Resultat real	Tester
1	Integrar la llibreria d'Apache Tika dins el sistema.	Instal·lar correctament la llibreria en el nostre projecte.	Com l'esperat.	AMV
2	Transformar qualsevol fitxer a un document de text pla.	Dissenyar algoritme que ens permeti extreure el contingut de qualsevol format i guardar-ho en un text pla.	Com l'esperat	AMV

Taula 20: Taula de proves d'unitat aplicades al framework Tika.

4.2 Proves d'unitat de l'algoritme dividir text

A continuació podem veure la taula de tests principals als quals hem sotmès l'algoritme de dividir el text d'un fitxer.

	Requeriment	Resultat esperat	Resultat real	Tester
1	Dividir el text mitjançant l'estratègia Punts.	Obtenir ArrayList on cada posició són els elements dividits cada cop que l'algoritme es troba amb un punt.	Com l'esperat	AMV
2	Dividir el text mitjançant l'estratègia Paraules.	Obtenir ArrayList on cada posició són els elements dividits segons el número de paraules que determina l'usuari.	Com l'esperat	AMV
3	Dividir el text mitjançant l'estratègia ConcPunts.	Obtenir ArrayList on cada posició són els elements dividits segons el número de punts que determina l'usuari.	Com l'esperat	AMV
4	Dividir el text mitjançant l'estratègia ConcPuntsBin.	Obtenir ArrayList on cada posició són els elements dividits segons el número de punts que determina l'usuari a més a més de decidir si és mostren les posicions parells o senars.	Com l'esperat	AMV

Taula 21: Taula de proves d'unitat sobre l'algoritme dividirText.

4.3 Proves d'unitat de l'API Summon Service

A la següent taula tenim les proves principals a les quals hem sotmès l'API de Summon Service.

	Requeriment	Resultat esperat	Resultat real	Tester
1	Connexió amb el servidor.	Dissenyar algoritme que permeti la connexió amb el servidor de Summon Service	Com l'esperat	AMV
2	Enviar les posicions dels l'ArrayList dividits mitjançant les estratègies.	Algoritme que enviï un volum gran d'ArraysList i que la connexió es realitzi satisfactòriament.	Com l'esperat	AMV
3	Modular la resposta en una estructura JSON.	Dissenyar algoritme que rebi un volum gran de d'estructures JSON i modular-ne el resultat.	Com l'esperat	AMV

Taula 22: Taula de proves d'unitat sobre l'API Summon Service.

4.4 Proves d'unitat de l'entorn web

L'objectiu principal de l'entorn web és que fos un sistema funcional, senzill, ràpid i segur.

	Requeriments	Resultat esperat	Resultat real	Tester
1	Integrar en l'entorn de desenvolupament el servidor d'aplicacions JBoss.	Instal·lar correctament el servidor d'aplicacions en el nostre entorn de desenvolupament.	Com l'esperat.	AMV i JDC
2	Executar el servidor web en local	Accedir al web usant un navegador a la direcció localhost:8080	Correcte .	AMV i JDC
3	L'aplicació és compatible en diferents sistemes operatius.	Obtenir una imatge del disseny del web inamovible i independent del sistema operatiu.	No superat. L'aplicació ha estat desenvolupada sota OS X i testejada i per	AMV

			tant funciona correctament sota Linux. No s'ha pogut comprovar el seu funcionament en un entorn windows	
4	Accedir a l'aplicació des de qualsevol navegador.	Obtenir el mateix disseny independentment del navegador utilitzat.	No superat. L'aplicació ha estat testejada amb els principals navegadors: Firefox, Chrome, Safari obtenint el resultat esperat. No s'ha comprovat el seu funcionament amb InternetExplorer.	AMV i JDC
5	Proveir accés per a dispositius mòbils	Que es pugi navegar per l'aplicació web utilitzant un dispositiu mòbil.	No superat. No es disposa d'un servidor.	AMV

Taula 23: Taula de proves d'unitat sobre l'entorn web.

5 Conclusions

En aquest capítol finalitzarem amb l'avaluació dels objectius un cop finalitzat el projecte així com un llistat de futures línies de treball, repassarem les desviacions de la planificació i exposarem una valoració personal de l'experiència obtinguda.

5.1 Valoració d'objectius assolits

Els nostre objectiu principal era oferir una eina per detectar text plagiat d'un fitxer prèviament seleccionat. Un cop realitzada l'aplicació que ofereix la funcionalitat esperada i acompleix els objectius marcats, podem afirmar que és viable el seu desenvolupament.

Si en repassem els objectius marcats fins de l'estudi de viabilitat (apartat. 2.3). Veiem que consta de set objectius:

1. Pujar els documents a tractar al servidor. **(Assolit)**
2. Dividir els documents segons estratègies, sense importar la seva extensió. **(Assolit)**
3. Cercar coincidències a l'API de Summon Service. **(Assolit)**
4. Visualitzar la informació obtinguda en un gestor de continguts web. **(Assolit)**
5. Els procediments de pujar els documents al servidor, dividir els documents i buscar coincidències a Summon Service es realitzin de manera asíncrona. **(Assolit)**
6. Pujar els documents a tractar al servidor utilitzant la tecnologia "drag and drop". **(Assolit)**
7. Opció d'enviar els resultats mitjançant correu electrònic i crear un arxiu pdf amb tota la informació obtinguda per aquell document. **(Assolit)**

5.2 Futures línies de treball

- Creació d'una base de dades al sistema que permetrà a l'usuari tindre un historial de les cerques i recuperar tota la informació tractada. Permetrà guardar perfils i configuracions més utilitzades pels usuaris, com pot ser auto enviar correu electrònic al finalitzar la cerca.
- Replantejar el projecte d'una altra manera. Un cop penjat l'arxiu, es mostri una visualització del contingut i enlloc d'analitzar-lo complet, com funciona ara, sigui l'usuari el que seleccioni el fragment a analitzar.
- Introduir noves estratègies a l'aplicació o millorar les existents, fent-les més eficients i per tal de poder disminuir el temps.
- Integració de l'aplicació per a dispositius mòbils (smartphones i tablets).
- Millores de cara l'usuari com pot ser traduir l'aplicació a múltiples idiomes, permetre a l'usuari seleccionar el color en que es ressalta el text plagiat, etc.

5.3 Seguiment de la planificació

La planificació dels projectes complexos tendeixen a ser optimistes i no acaben complint amb els plans establerts inicialment. La planificació del nostre projecte s'ha vist modificada en relació a l'inicial. Principalment degut als múltiples problemes amb la connexió a l'API Summon Service, l'eina principal d'aquest projecte.

A continuació podem veure un llistat de problemes que ens hem trobat al llarg del desenvolupament on hem hagut de fer un esforç extra per resoldre'ls i per tant la planificació s'ha vist modificada, sent la data de finalització finals de Juliol de 2013, això representa 27 setmanes.

- Els continus problemes en la connexió a l'API Summon Sercice.
- Mostrar de forma correcta la informació obtinguda de Summon Service en l'entorn web.
- La implementació de les connexions HTTP i els Servlets de manera asíncrona han suposat un esforç elevat.
- La implementació del "drag and drop".
- Enviar un correu electrònic plantilla amb un arxiu pdf adjunt.
- Canviar la forma de visualitzar la informació detallada de les coincidències i fer-ho mitjançant l'eina ModalBox.

Podem veure en el diagrama de Gantt (il·lustració) elaborat un cop realitzat el projecte que aquest s'ajusta al desenvolupament real que hem seguit des de finals d'Octubre de 2012. La primera planificació ha resultat massa optimista.

Cost Personal

	Hores	Preu / Hora	Cost total
Desenvolupador	280h	20€	5.600€
Cap de projecte	54h	40€	2160€

Taula 24: Taula de cost de personal.

Cost dels recursos

	Cost amortització	Cost unitari	Període d'utilització
MacBook Pro	307€	1100€	6,7
Asus Asipre One	84€	300€	6,7
OS X Mountain Lion	5€	17,99€	6,7
Microsoft Office 2011 MAC	106€	379€	6,7

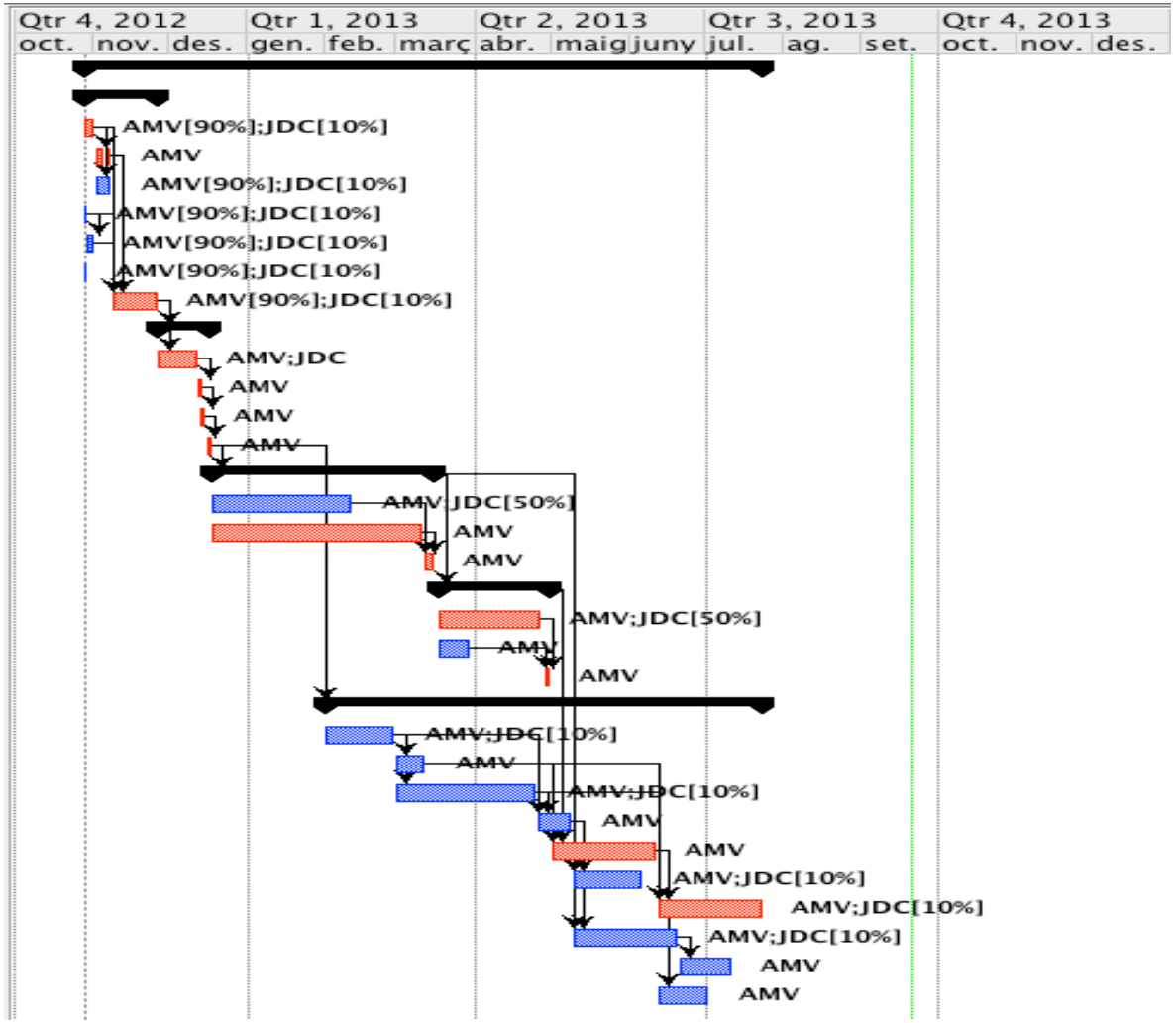
Taula 25: Taula de cost de maquinari i materials.

La desviació és considerable, sobre els 5.703€ pressupostats a l'inici en concepte del cost total de les activitats, s'ha augmentat un 36% sobre el valor inicial, això representa en termes absoluts 4.279€.

Aleshores el cost final del projecte un cop portat a terme, suma un total de:

- Desenvolupament de les activitats del projecte: 5.600€
- Costos d'amortització de material: 2.160€

Total : 7.760€



Il·lustració 26: Diagrama de Gantt del projecte un cop desenvolupat.

5.4 Valoració personal

Des d'una perspectiva personal, destacaria que aquest projecte ha estat més que un exercici acadèmic obligatori. Ha resultat ser un treball molt satisfactori a nivell professional que m'ha permès posar a proba tots els coneixements adquirits al llarg dels anys de la carrera. També m'ha permès tindre una visió més realista del que suposa un projecte d'aquesta magnitud i aproximar-me al món laboral.

D'altra banda, i des d'una vesant més tècnica he adquirit una sèrie de coneixements nous i molt valorables, que no els proporciona l'ensenyament universitari estricte. En primer lloc, ressaltar l'ús de Frameworks. Els Frameworks han resultat ser una eina molt útil perquè ens permeten un gran estalvi de temps als desenvolupadors. Perquè crear un algoritme quan ja està creat i en podem fer un ús personal?

En segon terme, realçar la utilitat de les API's, que ofereix als desenvolupadors la possibilitat de fer ús d'un conjunt de procediments per a ser utilitzat per un altre software. La qual cosa habilita una major connectivitat i integració entre aplicacions.

L'altre punt destacable és l'aprenentatge del llenguatge JAVA EE. Aquest llenguatge no forma part del currículum de la carrera, en canvi és molt utilitzat en el sector empresarial.

Per últim i no menys important, realitzar l'aplicació en un entorn web m'ha permès conèixer i aprendre més tècniques de desenvolupament com poden ser: JQuery, Prototype i XMLHTTPRequest.

D'altra banda també he millorat habilitats transversals tals com la planificació del temps i la responsabilitat que implica un projecte tant gran.

6 Glossari

6.1 Relació de taules

6.2 Relació de figures i il·lustracions

7 Bibliografia

7.1 Llibres

- Bruce Eckel. *Thinking in Java* (4a edició, 2006)

  Prentice Hall

- Erich Gamma, Richard Helm, Ralph Johnson i John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software* (1a edició, 10 Novembre 1994)

7.2 Digital

- The Summon Service

  http://www.serialssolutions.com/en/services/summon/, 2013

- Eclipse

  http://www.eclipse.org/, 2013

- Java Platform SE 7 API

  http://docs.oracle.com/javase/7/docs/api/, 2013

- Java EE 7 Specification API

  http://docs.oracle.com/javaee/7/api/, 2013

- Oracle

  http://www.oracle.com/index.html, 2013

- JBoss Community

  http://www.jboss.org/overview/, 2013

- JSON

  http://www.json.org/, 2013

- Ruby on Rails

  http://rubyonrails.org/

- HTML5 Rocks

  http://www.html5rocks.com/es/, 2013

- W3Schools

  http://www.w3schools.com/, 2013

- JQuery

  http://jquery.com/, 2013

- Prototype

  http://prototypejs.org/, 2013

- Project Management, OpenProj

  http://sourceforge.net/projects/openproj/, 2013

- ModalBox

  http://okonet.ru/projects/modalbox/index.html, 2013

- XMLHttpRequest

  https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest, 2013

- Stack Overflow

  http://stackoverflow.com/, 2013

- Wikipedia

  http://www.wikipedia.org/,2013

8 Agraïments

A la família, als amics i la meva parella que m'han donat recolzament i ajuda en molts moments.

Al professor i director del projecte Jordi Duran Cals pel seu temps i dedicació, per haver-se implicat de forma real i la seva ajuda i tenacitat en moments crítics del projecte.

Finalment donar un agraïment especial a l'enginyer de la Universitat Oberta de Catalunya, en Xavier Duran per la seva col·laboració desinteressada i haver-nos cedit el seu projecte "Apila".

Autor del document:

Albert Martínez Vilanova

Setembre de 2013