Gestor de documents

Primera entrega de Projecte de Programació

Q1 2022-23 | Subgrup 31.2

Pere Carrillo

Pol Garrido

Marc Ordóñez

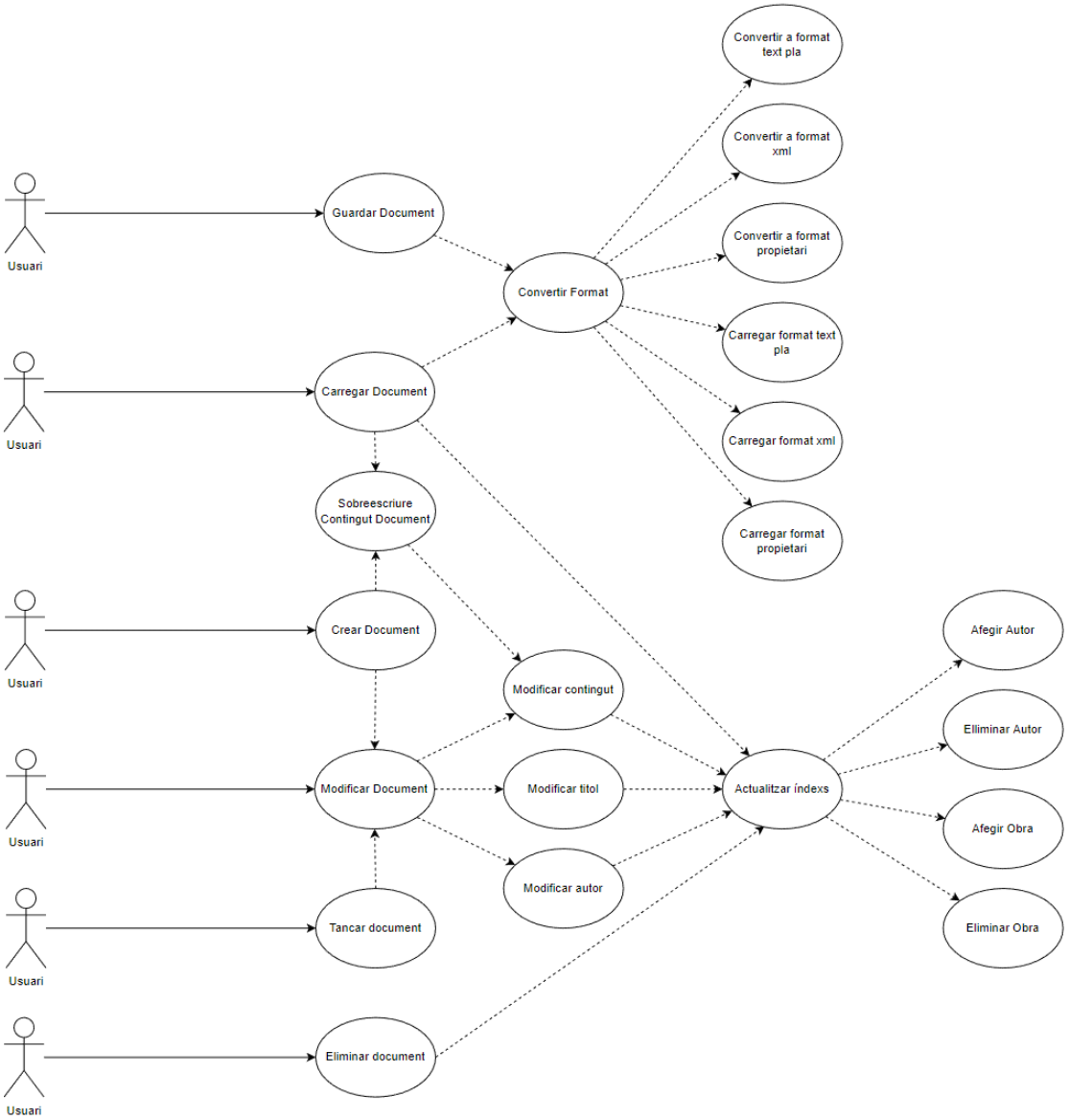
Laura Pérez

Índex

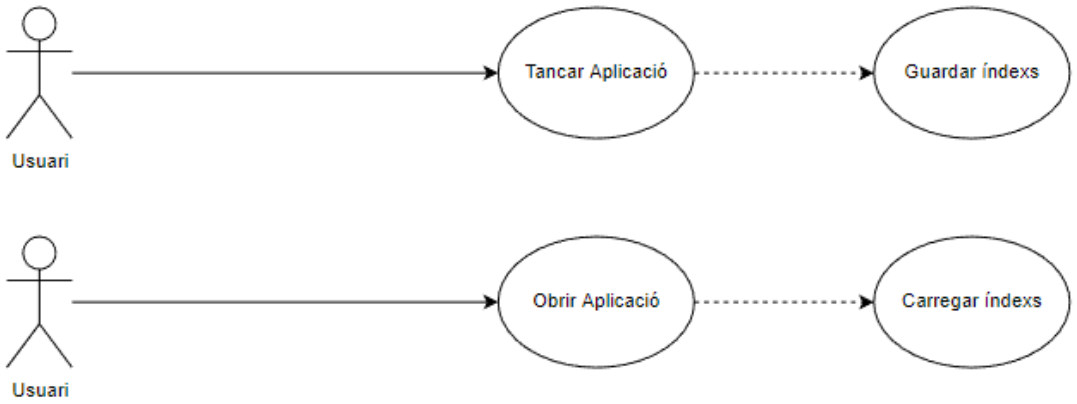
1 Casos d'ús

1.1 Actors

1.2 Diagrama




Usuari
Fer Consultes
Mostrar Contingut
K Documents semblants
K Documents rellevants per paraules
Llistar títols autor
Llistar autors per prefix
Consulta booleana
Guardar expressió booleana
Mostrar consultes booleanes guardades
Modificar consultes booleanes guardades
Gestionar expressions booleanes
Eliminar expressió booleana guardada
Modificar expressió
Mostrar historial consultes


Usuari
Tancar Aplicació
Guardar índexs
Usuari
Obrir Aplicació
Carregar índexs

1.3 Descripció

1.3.1 Gestor de documents

NOM	**Carregar document**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica el document o conjunt de documents a carregar de disc 2. El sistema escull quin tipus de conversió fer - Carregar document tipus txt - Carregar document tipus xml - Carregar document en format propietari 3. Els documents són afegits al sistema 4. S'actualitzen els índexs adientment - Si l'autor no existeix s'afegeix a l'índex d'autors
ERRORS I CURSOS ALTERNATIUS	1A. Si algun document ja estava carregat, es pregunta a l'usuari si vol sobreescriure, en cas afirmatiu, s'executa el cas d'ús "Sobreescriure document" 1B. Si ja no es poden suportar més documents dóna l'error corresponent

NOM	**Sobreescriure document**
ACTOR	Sistema
COMPORTAMENT	1. El sistema sobreescriu el document antic amb el nou 2. Es crida a modificar contingut

NOM	**Carregar document txt**
ACTOR	Sistema
COMPORTAMENT	1. El sistema llegeix el document en format txt i n'extreu la informació
ERRORS I CURSOS ALTERNATIUS	1A. La informació dins del txt és incorrecta. El sistema avisa a l'usuari

NOM	**Carregar document xml**
ACTOR	Sistema
COMPORTAMENT	1. El sistema llegeix el document en format xml i n'extreu la informació
ERRORS I CURSOS ALTERNATIUS	1A. El document no té les etiquetes adequades. El sistema avisa a l'usuari

NOM	**Carregar document format propietari**
ACTOR	Sistema
COMPORTAMENT	1. El sistema llegeix el document en el format propietari i n'extreu la informació

NOM	**Guardar Document**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica el document a guardar. A més, indica el format en que ho vol. - Els formats possibles són: text pla, xml i el format propietari 2. El sistema crida al conversor adequat - Convertir a txt - Convertir a xml - Convertir a format propietari
ERRORS I CURSOS ALTERNATIUS	1A. Si el document no existeix, retorna l'error corresponent

NOM	**Convertir Format**
ACTOR	Sistema
COMPORTAMENT	1. A partir d'una crida de l'usuari, el sistema farà la conversió adient
ERRORS I CURSOS ALTERNATIUS	1A. Si no s'ha pogut convertir el format, el sistema avisa de l'error

NOM	**Convertir a txt**
ACTOR	Sistema
COMPORTAMENT	1. El sistema converteix el document a txt
ERRORS I CURSOS ALTERNATIUS	1A. El sistema no ha pogut convertir el document a txt, es mostra un missatge d'error

NOM	**Convertir a xml**
ACTOR	Sistema
COMPORTAMENT	1. El sistema converteix el document a xml
ERRORS I CURSOS ALTERNATIUS	1A. El sistema no ha pogut convertir el document a xml, es mostra un missatge d'error

NOM	**Convertir a format propietari**
ACTOR	Sistema
COMPORTAMENT	1. El sistema converteix el document al format propietari
ERRORS I CURSOS ALTERNATIUS	1A. El sistema no ha pogut convertir el document al format propietari, es mostra un missatge d'error

NOM	**Crear document**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica el document a crear (títol + autor+contingut) 2. Es crea un document 3. S'actualitzen els índexs adientment - Si l'autor és nous, s'afegeix als índexs - S'afegeix el document identificat per títol i autor a l'índex obres
ERRORS I CURSOS ALTERNATIUS	1A. Si el document ja existeix, avisar a l'usuari si el vol sobreescriure, en cas afirmatiu es crida a "Sobreescriure document"

NOM	**Eliminar document**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica el document a donar de baixa (títol + autor) 2. S'actualitzen els l'índexs adientment - S'elimina el document identificat per l'autor i el títol - Si l'autor ja no té obres, s'elimina l'autor de l'índex d'obres i d'autors 3. S'elimina el document del sistema
ERRORS I CURSOS ALTERNATIUS	1A. Si el document no existeix, el sistema avisa de l'error 3A. Si no s'ha pogut eliminar el document, el sistema avisa de l'error

NOM	**Tancar documents**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari prem un botó per tancar el document 2. El document es tanca
ERRORS I CURSOS ALTERNATIUS	2A. El document ha estat modificat però no s'han desat els canvis. El sistema pregunta a l'usuari si vol desar els canvis. En cas afirmatiu es crida a la funció que actualitza el document.

NOM	**Modificar document**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica el document a modificar (títol + autor) 2. El sistema mostra la informació del document 3. L'usuari modifica la informació 4. L'usuari decideix si vol desar els canvis o no
ERRORS I CURSOS ALTERNATIUS	1A. Si no existeix el document a modificar, el sistema avisa de l'error 4A. Si l'usuari desa els canvis - Si l'usuari ha canviat el títol, es crida a "Modificar títol" - Si l'usuari ha canviat l'autor, es cria a "Modificar autor" - Si l'usuari ha canviat el contingut, es crida a "Modificar Contingut" - Es crida a "Actualitzar Índexs" 4B. Si no es desen els canvis, no es modifica res

NOM	**Modificar títol**
ACTOR	Sistema
COMPORTAMENT	1. L'usuari ha modificat el títol 2. Es canvia el títol pel nou 3. Es desa el document ja modificat

NOM	**Modificar autor**
ACTOR	Sistema
COMPORTAMENT	1. L'usuari ha modificat l'autor 2. Es canvia l'autor pel nou 3. Es desa el document ja modificat

NOM	**Modificar contingut**
ACTOR	Sistema
COMPORTAMENT	1. L'usuari ha modificat el contingut 2. Es canvia el contingut pel nou 3. Es desa el document ja modificat

NOM	**Mostrar contingut**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica un títol i un autor 2. Es crida a l'índex d'obres 3. El sistema mostra el contingut del document identificat amb el títol i l'autor
ERRORS I CURSOS ALTERNATIUS	2A. L'autor no existeix. El sistema retorna un missatge d'error 2B. Si l'autor no té obres no hauria d'aparèixer al sistema però es mostraria un avís indicant que no té documents al seu nom 2C. L'autor no té l'obra amb el títol indicat. El sistema retorna un missatge d'error

NOM	**K Documents semblants**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica un enter k i el títol i l'autor d'un document D 2. El sistema crida a l'espai vectorial 3. Es mostren k documents semblants a D
ERRORS I CURSOS ALTERNATIUS	2A. No existeixen k documents semblants a D, es mostren n < k documents 2B. El document identificat amb títol i autor no existeix, el sistema mostra l'error

NOM	**K Documents rellevants per paraules**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica un enter k i introdueix un conjunt de paraules 2. El sistema crida a l'espai vectorial 3. Es mostren k títols i autors de documents rellevants segons les paraules introduïdes
ERRORS I CURSOS ALTERNATIUS	2A. No existeixen k documents semblants a D, es mostren n < k documents 2B. No existeix cap document semblant, es mostra un missatge d'error

1.3.2 Gestor d'autors

NOM	**Afegir autor**
ACTOR	Sistema
COMPORTAMENT	1. Se li afegeix un nou autor al sistema
ERRORS I CURSOS ALTERNATIUS	1A. L'autor ja existeix, es mostra un missatge d'error, informant a l'usuari que ja existeix l'autor indicat

NOM	**Eliminar autor**
ACTOR	Sistema
COMPORTAMENT	1. El sistema elimina l'autor
ERRORS I CURSOS ALTERNATIUS	1A. L'autor no existeix, es mostra un missatge d'error indicant que l'autor no existeix

NOM	**Llistar títols autors**
ACTOR	Usuari
COMPORTAMENT	2. L'usuari indica el nom d'un autor 3. Es crida al gestor d'índexs per buscar tots els documents d'aquest autor 4. El sistema mostra a l'usuari tots els títols dels documents de l'autor
ERRORS I CURSOS ALTERNATIUS	2A. L'autor no existeix. El sistema retorna un missatge d'error 3A. Si l'autor no té obres no hauria d'aparèixer al sistema però es mostraria un avís indicant que no té documents al seu nom

NOM	**Llistar autors per prefix**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica el prefix 2. Es crida a l'índex d'Autors 3. El sistema mostra tots els autors que comencen amb el prefix indicat
ERRORS I CURSOS ALTERNATIUS	2.A No existeix cap autor amb el prefix indicat, el sistema mostra a l'usuari que no existeix cap autor que comenci amb el prefix

1.3.3 Gestor d'obres

NOM	**Afegir obra**
ACTOR	Sistema
COMPORTAMENT	1. S'afegeix l'obra a l'índex, identificat pel títol i l'autor
ERRORS I CURSOS ALTERNATIUS	1A. L'obra ja existeix

NOM	**Eliminar obra**
ACTOR	Sistema
COMPORTAMENT	1. S'elimina l'obra de l'índex d'obres
ERRORS I CURSOS ALTERNATIUS	1A. L'obra no existeix, el sistema avisa de l'error

1.3.4 Gestor consultes booleanes

NOM	**Consulta booleana**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari introdueix una expressió booleana 2. El sistema crida a conjunt consultes booleanes 3. Es mostren els títols i els autors dels documents que compleixen l'expressió booleana
ERRORS I CURSOS ALTERNATIUS	1A. L'expressió booleana és incorrecta, es mostra un missatge de l'error

NOM	**Guardar expressió booleana**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari selecciona una expressió booleana 2. El sistema es guarda l'expressió booleana amb el resultat
ERRORS I CURSOS ALTERNATIUS	2A. La consulta ja existeix, es mostra un missatge avisant que la consulta ja existeix

NOM	**Eliminar expressió booleana guardada**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica l'expressió a eliminar 2. El sistema elimina l'expressió seleccionada
ERRORS I CURSOS ALTERNATIUS	2A. La consulta no existeix, el sistema mostra un missatge d'error

NOM	**Modificar Consulta booleana guardada**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari selecciona les consultes booleanes guardades 2. L'usuari modifica una consulta booleana 3. El sistema es guarda la consulta modificada

NOM	**Mostrar consultes booleanes guardades**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari indica que es mostrin les consultes booleanes guardades 2. Es crida al conjunt de consultes booleanes 3. El sistema mostra les consultes booleanes guardades
ERRORS I CURSOS ALTERNATIUS	2A. No hi ha consultes guardades. El sistema informa a l'usuari de la inexistència de consultes guardades.

NOM	**Mostrar historial de les últimes consultes realitzades**
ACTOR	Sistema
COMPORTAMENT	1. L'usuari està a la pestanya de consultes booleanes 2. El sistema mostra les últimes consultes realitzades

1.3.5 Gestor de l'aplicació

NOM	**Obrir aplicació**
ACTOR	Usuari
COMPORTAMENT	1. L'usuari ha obert l'aplicació 2. Es crida a carregar índexs de l'última sessió

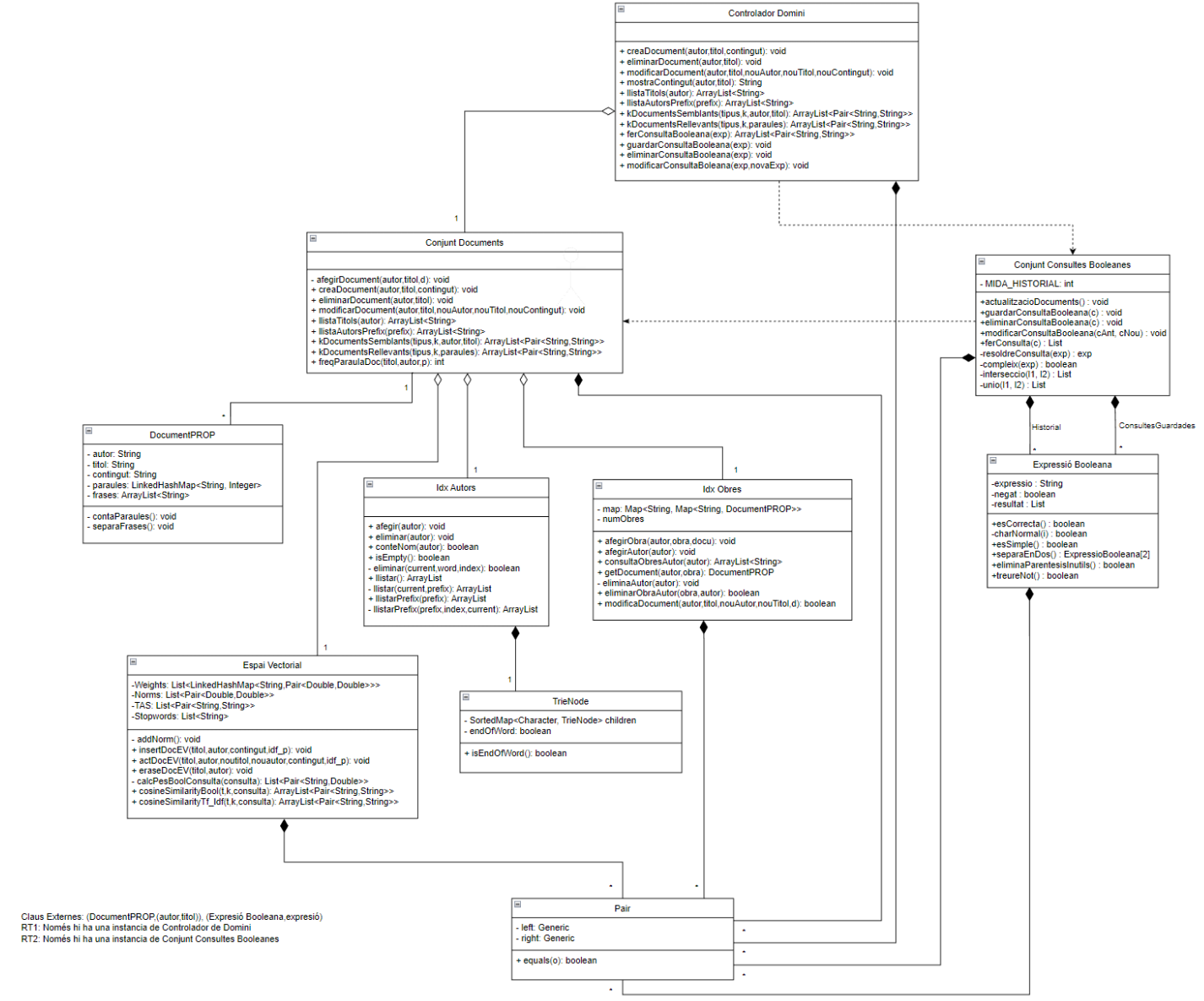
NOM	**Carregar índexs**
ACTOR	Sistema
COMPORTAMENT	1. Es carreguen els índexs de l'última sessió
ERRORS I CURSOS ALTERNATIUS	1A. Ha hagut un error carregant els índexs, el sistema mostra l'error

NOM	**Guardar índexs**
ACTOR	Sistema
COMPORTAMENT	1. Es guarden els índexs
ERRORS I CURSOS ALTERNATIUS	1A. Error al guardar els índexs, el sistema avisa a l'usuari de l'error

NOM	**Tancar aplicació**
ACTOR	Usuari
COMPORTAMENT	3. L'usuari prem un botó per tancar l'aplicació 4. Es desen els índexs 5. L'aplicació es tanca
ERRORS I CURSOS ALTERNATIUS	3A. En el moment de tancar l'aplicació s'estava modificant un document però no s'han desat els canvis. El sistema pregunta a l'usuari si vol desar els canvis. En cas afirmatiu es crida a la funció que actualitza el document.

2 Model conceptual de dades

2.1 Diagrama



2.2 Descripció

2.2.1 Classe Espai Vectorial

Aquesta classe ens permet emmagatzemar la informació necessària per poder resoldre les consultes de "*k* documents similars respecte a un document D" i "*k* documents similars donat un conjunt de paraules". La informació necessària per aquestes consultes són els pesos de les paraules de tots els documents registrats, amb alguna manera d'identificar cada document. D'això se'n carreguen *Weights* i TAS, essent *Weights* els pesos (amb estrategia booleana i amb tf-idf) i TAS la manera d'identificar conjunt de pesos ràpidament per títol i autor. Com per fer aquestes consultes necessitem computar la similaritat del cosinus, també es guarda la norma dels diferents pesos a *Norms* per no haver de calcular-los a cada consulta. Finalment a l'atribut *Stopwords* tenim totes les *stopwords* en català, castella i angles per no comptar-les a l'espai vectorial.

Les funcions són, descomptant *getters* i *setters*:

- **addNorm(): void**
    - Aquesta funció s'encarrega de calcular i afegir a les estructures de dades la norma dels vectors de pesos d'un document nou a afegir.
    - Cost: O(n), sent n el nombre de paraules del document

- **insertDocEV(titol,autor,contingut,idf_p): void**
    - Aquesta funció s'encarrega d'afegir un nou document a l'espai vectorial, afegint els pesos i calculant les seves normes. A més comprova si ja existeix un document amb el mateix nom i llença una excepció si és el cas
    - Cost: O(n), sent n el nombre de paraules del document, pel càlcul dels pesos

- **actDocEV(titol,autor,noutitol,nouautor,contingut,idf_p): void**
    - Aquesta funció s'encarrega d'actualitzar un fitxer ja registrat a l'espai vectorial en cas que es produeixi una modificació del mateix. Si no

existeix un document prèviament afegit amb aquests identificadors (títol,autor) llença una excepció. En cas contrari, actualitza adientment el document, actualitzant pesos i normes.

- Cost: O(n), sent n el nombre de paraules del document, pel càlcul dels pesos

- **eraseDocEV(titol,autor): void**
  - Aquesta funció s'encarrega d'eliminar tota la informació d'un document de l'espai vectorial quan aquest deixa de formar part del sistema. Si no existeix un document amb identificador (titol,autor) llença una excepció.
  - Cost: O(1), ja que es suma de costos constants d'eliminar objectes

- **calcPesBoolConsulta(consulta): List<Pair<String,Double>>**
  - Aquesta funció s'encarrega de transformar una consulta, que pot ser tot el contingut d'un document o un conjunt de paraules, en pesos seguint l'estratègia booleana (tots a 1 si no són *stopwords*) per poder realitzar les consultes.
  - Cost: O(n), sent n el nombre de paraules de la consulta

- **cosineSimilarityBool(t,k,consulta): ArrayList<Pair<String,String>>**
  - Aquesta funció s'encarrega de calcular la similaritat del cosinus, utilitzant les dades emmagatzemades a l'espai vectorial, per tal de decidir els k documents més similars respecte la consulta i retornar-los com a resposta a la consulta de "k similars". El paràmetre *t* s'utilitza per discernir si la consulta es tracta d'un document o d'un conjunt de paraules. Com el nom indica, utilitza l'estrategia booleana per resoldre les consultes.
  - Cost: O(nm), sent n el nombre de documents registrats i m el nombre de paraules de la consulta

- **cosineSimilarityTf_Idf(t,k,consulta): ArrayList<Pair<String,String>>**
    - Aquesta funció s'encarrega de calcular la similaritat del cosinus com l'anterior, però utilitza l'estratègia tf-idf d'assignació de pesos per resoldre les consultes.
    - Cost: O(nm), sent n el nombre de documents registrats i m el nombre de paraules de la consulta

2.2.2 Classe Índex Obres

Aquesta classe és una estructura en la que s'emmagatzemen els documents en funció de l'autor i el títol. Els documents s'emmagatzemen en ordre alfabètic per poder facilitar el llistat i la consulta de la informació. La idea és que si en futures entregues els documents s'han de guardar en memoria, en comptes de guardar el document es guardi només el path. D'aquesta forma a partir d'un autor i un títol podem obtenir el path (o en aquesta entrega el document) amb la informació associada. Creiem que no serà bona idea guardar el document en aquesta estructura perquè al tractar-se d'un gestor de documents el nombre total d'arxius i la seva mida pot ser molt gran, lo qual pot fer que no càpiga tota aquesta informació a memòria RAM.

Les funcions són, descomptant *getters* i *setters*:

- **afegirObra(String autor, String obra, DocumentPROP docu): void**
    - Donat un autor i una obra afegeix el Document corresponent.
    - Cost: O(log n + log m), sent n el nombre d'autors i m el nombre d'obres de l'autor.

- **afegirAutor(String autor): void**
    - Afegeix un nou autor que no té cap obra.
    - Cost: O(n), sent n el nombre d'autors.

- **consultaObresAutor(String autor): ArrayList<String>**
    - Retorna un ArrayList de Strings amb totes les obres d'un autor ordenades alfabèticament.
    - Cost: O(log n + m), sent n el nombre d'autors i m el nombre d'obres de l'autor. (log n de trobar l'autor i n de recorregut lineal per llistar).

- **getDocument(String autor, String obra): DocumentPROP**
    - Ara mateix retorna el Document amb tota la informació donat un autor i un títol (obra). Més endavant la idea és que retorni el path amb la direcció de memòria que conté la informació del document.
    - Cost: O(log n + log m), sent n el nombre d'autors i m el nombre d'obres de l'autor.

- **eliminarAutor(String autor): void**
    - S'elimina a l'autor de l'estructura de dades. Aquesta funció només es cridarà quan eliminem una obra d'aquest autor i ja no tingui cap obra al sistema. Només en aquest cas es cridarà a la funció. Per tant, ens assegurem de no estar eliminant a un autor que encara tingués obres. La funció és privada per aquest motiu.
    - Cost: O(log n), sent n el nombre d'autors.

- **eliminarObraAutor(String obra, String autor): boolean**
    - Elimina l'obra d'un autor i retorna true si aquell autor ja no té cap obra. En aquest cas també s'eliminarà a l'autor.
    - Cost: O(log n + log m), sent n el nombre d'autors i m el nombre d'obres de l'autor.

- **getTitolsAutors(): ArrayList<Pair<String, String>>**
    - Retorna un ArrayList de Pairs amb totes les obres d'un autor. El primer element del pair és el títol i el segon l'autor.
    - Cost: O(log n + log m), sent n el nombre d'autors i m el nombre d'obres de l'autor.

- **modificaDocument(String autor, String titol, String nouAutor, String nouTitol, DocumentPROP d): boolean**
    - Modifica un document. Donat un autor i un títol busca el document original. Si l'autor i el títol nous són els mateixos als actuals, només modifica el contingut del document sobreescrivint amb el nou document.
    - En cas de que canviï l'autor o el títol, primer s'elimina el document amb l'autor i títols originals i després s'afegeix el nou document amb el títol i l'autor nous. Si a l'eliminar el document antic l'autor antic es queda sense obres, s'eliminarà a l'autor antic i es retornarà true. En cas contrari es retorna false.
    - Cost: O(log n + log m), sent n el nombre d'autors i m el nombre d'obres de l'autor. En cas pitjor és el cos d'eliminar obra més el d'afegir obra i tots dos són O(log n + log m).

2.2.3 Classe Índex Autors

Aquesta classe és una estructura (Trie) en què s'emmagatzemen els noms dels autors. Aquesta estructura ens permet mantenir els noms en ordre alfabètic i llistar per prefixos.

Les funcions són, descomptant *getters* i *setters*:

- **IdxAutors()**
    - Constructora. Crea un nou TrieNode que serà l'arrel de l'índex.
    - Cost: O(1)

- **afegir(String autor): void**
    - Afegeix l'autor a l'índex.
    - Cost: O(m), sent m el nombre de lletres de l'autor (perquè el nombre de fills està limitat al nombre de lletres de l'abecedari).

- **conteNom(String autor): boolean**
    - Retorna cert si l'autor pertany a l'índex i fals si no existeix.
    - Cost: O(m), sent m el nombre de lletres de l'autor

- **isEmpty(): boolean**
    - Retorna cert si no hi ha cap autor i fals en cas contrari.
    - Cost: O(1)

- **eliminar(TrieNode current, String word, int index): boolean**
    - Funció recursiva per eliminar una paraula donat un node, una paraula i un índex per indicar la lletra de la paraula que volem eliminar. Si el node actual conté la lletra de la paraula word que està en la posició index, es mira si conté un fill amb el següent caràcter de la paraula i fa una crida recursiva a aquest node fins que arriba a l'última lletra. Si pel camí arriba a un node que cap dels seus fills no és el següent caràcter de la paraula vol dir que la paraula no existeix i llençarà l'excepció. Si a l'eliminar la paraula ja no penja cap paraula d'aquell node, s'elimina el node sencer.
    - Cost: O(m-index), sent m el nombre de lletres de word.

- **llistar(): ArrayList**
    - Retorna un ArrayList de Strings amb tots els autors de l'índex ordenats alfabèticament.
    - Cost: O(n), sent n el nombre d'autors que es troben a l'índex.

- **llistar(TrieNode current, String prefix): ArrayList**
    - Llista tots els autors (ordenats alfabèticament) que es troben a partir d'un node. És a dir, tots els autors que comencen pel prefix que va des de l'arrel fins a aquell node.
    - Cost: O(n), sent n el nombre d'autors que es troben a partir del node current.

- **llistarPrefix(String prefix): ArrayList**
    - Llista tots els autors que comencen pel prefix donat. Es va iterant pels nodes fins arribar a la profunditat de l'última lletra del prefix i a partir d'allà es llisten tots els autors que compleixin el prefix.
    - Cost: O(m+n), sent n el nombre d'autors que comencen pel prefix i m el nombre de caràcters del prefix.

- **llistarPrefix(String prefix, int index, TrieNode current): ArrayList**
    - Aquesta funció serveix per arribar fins al node que conté l'última lletra del prefix. índex indica la posició de la lletra del prefix en la que ens trobem. La funció mirarà si un dels fills d'aquest node és la següent lletra del prefix i cridarà a la funció recursiva amb aquell fill i la següent posició de l'índex. Si no troba cap fill amb la següent lletra del prefix, no hi ha cap autor que comenci per aquell prefix i retorna una llista buida.
    - Cost: Cost: O(m+n), sent n el nombre d'autors que comencen pel prefix i m el nombre de caràcters del prefix.

2.2.4 Classe Document PROP

Aquesta classe emmagatzema els atributs de tots els documents, és a dir, el nom de l'autor, el títol del document i el contingut del document, a partir del qual calculem les paraules i les seves aparicions, les frases i el nombre de frases.

Cada cop que canviem el contingut es cridaran aquestes dues funcions per assignar-li els valors corresponents als atributs numFrases, paraules i frases.

- **contaParaules(): void**
    - Aquesta funció agafa totes les paraules que estan al contingut, treu els signes de puntuació i conta les seves aparicions. També treu les paraules que comencen amb una lletra i apòstrof.
    - Cost: O(n), n la mida del contingut
- **separaFrases(): void**
    - Separa les frases, tenint en compte que frases són totes les que acaben amb un punt, una exclamació o un interrogant. També elimina les exclamacions i els interrogants invertits.
    - Cost: O(n), n la mida del contingut

Per consultar les paraules del document, tenim aquestes dues funcions:

- **teParaula(String p): boolean**
    - Retorna cert si la paraula p pertany al document
    - Cost: O(1)
- **aparicionsParaula(String p): int**
    - La paraula p ha d'existir, retorna el nombre d'aparicions de la paraula
    - Cost: O(1)

2.2.5 Classe Expressió Booleana

Aquesta classe guarda una consulta booleana. Una consulta booleana la definim com: La consulta en sí, guardada en un String, un booleà per saber si està negada, i el resultat de la consulta, si el sabem.

Hem pres la decisió de només poder buscar caràcters especials (&, |, {, (, !,), }) si els posem entre cometes, ja que si no es produeixen moltes ambigüitats i potser una consulta on l'usuari s'ha equivocat pot seguir donant un resultat.

L'objectiu d'aquesta classe serà facilitar la feina al Conjunt d'Expressions Booleanes, tenint els següents mètodes principals:

- **esCorrecta() : boolean**
    - Aquesta consulta retorna cert si l'expressió sobre la que es crida és correcta i fals en cas contrari. Per saber si una expressió és correcta comprovem els següents casos:
        - Que tota l'expressió siguin seqüències de literal - operador - literal tenint en compte els parèntesis. (Un literal pot ser una paraula sola, una seqüència de caràcters delimitada per "" o un conjunt de paraules delimitats per {}).
        - Que no quedi cap parèntesi, claudàtor o cometes sense tancar i que no se'n tanquin de més.
        - Que les NOT s'apliquin sobre un literal, sobre un parèntesis o sobre una altra NOT.
    - Aquesta consulta utilitza la funció charNormal(int), per comprovar els caràcters que hi ha després d'una NOT
    - Cost: O(n) sent n el nombre de caràcters de la consulta, ja que hem de recórrer tota la consulta caràcter per caràcter en cas pitjor. En el cas que aquest sigui una NOT haurem de comprovar els següents caràcters diferents d'un espai, però com que l'únic cas en que s'itera per tots els elements restants a charNormal() és en el cas que siguin espais, per les següents no s'iterarà (perquè no seran NOTs) i per tant no fa augmentar el cost.

- **esSimple() : boolean**
    - Aquesta consulta retorna cert si l'expressió sobre la que es fa és simple, és a dir, conté només un literal (descrit a la funció anterior).
    - Cost: O(n), ja que en cas pitjor haurem de recórrer tot el String.
- **separaEnDos() : ExpressioBooleana[2]**
    - Aquesta consulta és la que permet aplicar el nostre algorisme de Dividir i Vèncer. Retorna l'expressió booleana sobre la que s'ha cridat dividida en 2. Intenta dividir-la primer en dos per una OR exterior, i si no n'hi ha cap, per una AND. Sempre ens trobarem en algun d'aquests dos casos, ja que si no, o la consulta era incorrecte o la hem resolt just abans, però si no passa res d'això retornem la consulta a resultat[0] i null a resultat[1].
    - Cost: O(n), ja que en cas pitjor farem dos recorreguts pel String, un per buscar ORs i l'altre per buscar ANDs.
- **eliminaParentesisInutils() : boolean**
    - Aquesta funció intenta eliminar un possible parèntesis general que afecti a tota la funció. Si ho aconsegueix retorna true, si no false.
    - Cost: O(n), ja que en cas pitjor haurem de recórrer tot el String. Tot i que si la consulta no comença per parèntesi el cost és O(1).
- **treureNot() : boolean**
    - Aquesta funció fa el mateix que la anterior però buscant una NOT. A l'algorisme s'executen les dues fins que les dos retornen fals, per simplificar totes les expressions i poder cridar a separaEnDos().
    - Cost: El mateix que la funció anterior.

La classe té altres mètodes de poca importància, que son necessaris per fer funcionar la resta de l'algorisme. Alguns d'aquests son: getters i setters, equals(), trim(), length(), subString()...

2.2.6 Classe Conjunt Consultes Booleanes

Aquesta classe és l'encarregada de resoldre les consultes booleanes. Ha de poder rebre una consulta i retornar un resultat, mantenir un historial i permetre guardar un conjunt de consultes a decidir per l'usuari.

L'historial el guardem en una LinkedList, ja que ens interessa poder fer ràpidament insercions al principi i eliminacions de l'últim element com en una cua, però podent tenir accés a qualsevol dels elements del mig. La mida màxima de l'historial la guardem en un atribut static i final de la classe. Quan fem una consulta i es supera aquest nombre, s'elimina la consulta feta més antigament i s'afegeix la nova al principi. A l'historial guardem les expressions i els resultats, per poder donar un resultat ràpidament si la consulta ja s'havia fet anteriorment, però per tant, cada cop que s'afegeix, elimina o modifica un document hem d'invalidar totes les consultes i marcar-les per resoldre de nou si les criden.

Per oferir una gestió de les consultes booleanes permetem a l'usuari poder guardar consultes, modificar alguna que ja tenia guardada o eliminar-ne una. El conjunt de consultes de l'usuari les guardem en un ArrayList, ja que és una estructura que ens permet poder afegir, eliminar i consultar un element en qualsevol posició. Tenim, per això, el mateix problema que amb l'historial, i haurem d'invalidar totes les consultes cada cop que s'afegeix, modifica o elimina un document.

Per aquesta funcionalitat tenim les següents funcions:

- **guardarConsultaBooleana(String) : void**
    - Funció que comprova que la consulta sigui correcta i la guarda. En cas que no sigui correcta llança una excepció.
    - Cost:

- **modificarConsultaBooleana(String, String) : void**
    - Aquesta funció rep dos paràmetres, la consulta antiga i la nova. El que fa la funció és eliminar la antiga i crear i guardar la nova, fent saltar excepcions en els casos corresponents.
- **eliminarConsultaBooleana(String) : void**
    - Aquesta funció elimina una consulta de les guardades per l'usuari. En cas que no existeixi fa saltar una excepció.
    - Cost:

- **actualitzacioDocuments() : void**
    - Aquesta funció invalida totes les consultes guardades per l'usuari i les guardades a l'historial.
    - Cost:

Per resoldre les consultes booleanes utilitzem un algorisme basat en Dividir i Vèncer que veurem en un apartat posterior (4.3.1). L'algorisme fa servir les següents funcions:

- **ferConsulta(String) : ArrayList<Pair<String, String>>**
    - És la consulta bàsica de l'algorisme, i la que es crida des de fora. Retorna un conjunt de Pair<Titol, Autor> que son els documents que compleixen la consulta demanada. En cas que la consulta sigui incorrecta llança l'excepció adient. També afegeix la consulta a l'historial. Després crida a resolConsulta(expBool) i es guarda el que retorna en un set, per eliminar els documents repetits, retornant-ho després en forma d'ArrayList<Titol, Autor>.
    - Cost: El cost d'aquesta funció és molt difícil de calcular teòricament, ja que depèn de molts factors diferents, com el nombre de documents al sistema, la mida d'aquests, el nombre de literals a la consulta, la mida de la solució de cada subconsulta...

- **resolConsulta(ExpressioBooleana) : ExpressioBooleana**
    - Aquesta funció és la que implementa la recursivitat de l'algorisme. Té dos casos base: El primer, si l'expressió ja tenia solució retorna, i el segon, si l'expressió és simple, la resol i retorna.

      En cas que no es compleixi cap dels dos casos base separa en dos l'expressió i es crida a si mateixa per cada subexpressió.

      Finalment, si havia separat per una OR, fa la unió dels resultats, i si havia separat per una AND, la intersecció. En cas que la consulta estigués negada s'inverteixen les funcions per calcular el resultat.

    - Cost: El cost d'aquesta funció és el mateix que el de ferConsulta(), molt difícil de calcular teòricament.

- **compleix(String, ExpressióBooleana) : boolean**
    - Retorna cert si la frase que se li passa compleix l'expressió que se li passa. Comprovem els 3 possibles casos d'expressió: Tenir una paraula sola, un conjunt de paraules o una seqüència de paraules.
    - Cost: O(n), sent n el nombre de paraules de la frase

- **intereccio(ArrayList<Pair<Pair<String, String>, String>> docs, ArrayList<Pair<Pair<String, String>, String>> altres): ArrayList<Pair<Pair<String, String>, String>>**
    - Es fa la intersecció de les dues llistes docs i altres. Retorna una llista de títol, autor, frase, que pertanyen tan a docs com a altres. Per a això, en el cas que docs sigui més gran, anem mirant si cada element de altres pertany a docs. En cas contrari es comprova si els elements de docs pertanyen a altres. Si l'element pertany a les dues llistes, s'afegeix a la llista resultant.
    - Cost: O(n*m), n la mida de docs i m la mida de altres

- **unio(ArrayList<Pair<Pair<String, String>, String>> docs, ArrayList<Pair<Pair<String, String>, String>> altres): ArrayList<Pair<Pair<String, String>, String>>**
    - Es fa la unió de les dues llistes docs i altres. Retorna una llista de tots els documents pertanyents a docs i altres. Per a això, clonem docs al resultat i li afegim tots els elements de altres que no pertanyen a docs.
    - Cost: Cost: O(n*m), n la mida de docs i m la mida de altres

2.2.7 Classe Conjunt Documents

Aquesta classe representa el conjunt de tots els documents, ja que manté els índexs dintre seu. Actua com l'element comunicador entre els documents emmagatzemats i l'espai vectorial o el conjunt de consultes booleanes, entre d'altres funcions. Detallem ara les funcions de la classe:

- **afegirDocument(autor,titol,d): void**
    - Afegeix un document als indexs i al espai vectorial donat un autor un titol i un document.
    - Cost: O(n), sent n el nombre de paraules del document, degut al cost del càlcul del idf

- **creaDocument(autor,titol,contingut): void**
    - Crea un document donat un autor, un titol i contingut i crida a afegirDocument
    - Cost: O(n), degut a afegirDocument

- **eliminarDocument(autor,titol): void**
    - Elimina un document donat el seu titol i autor de les estructures de dades
    - Cost: O(1)

- **modificarDocument(autor,titol,nouAutor,nouTitol,nouContingut): void**
    - Modifica un document crida a que s'actualitzi a les diferents estructures de dades (indexs i espai vectorial).
    - Cost: O(n), sent n el nombre de paraules del document, degut al cost del càlcul del idf

- **llistaTitols(autor): ArrayList<String>**
    - Retorna la llista de titols d'un autor
    - Cost: O(1)

- **llistaAutorsPrefix(prefix): ArrayList<String>**
    - Retorna la llista d'autors per prefix
    - Cost: O(1)

- **kDocumentsSemblants(tipus,k,autor,titol): ArrayList<Pair<String,String>>**
    - Retorna els k documents més semblants al consultat amb l'estrategia d'assignació de pesos escollida amb el paràmetre tipus
    - Cost: O(1)

- **kDocumentsRellevants(tipus,k,paraules): ArrayList<Pair<String,String>>**
    - Retorna els k documents més rellevants per la consulta en forma de conjunt de paraules amb l'estrategia d'assignació de pesos escollida amb el paràmetre tipus
    - Cost: O(1)

- **freqParaulaDoc(titol,autor,p): int**
    - Retorna la freqüencia local d'una paraula al document indicat
    - Cost: O(1)

2.2.8 Classe Controlador Domini

Aquesta classe actuara com element de comunicació entre la capa de presentació i la de domini, o entre la de persistencia i la de domini. Es la classe on es reben les consultes i es deriven a les classes especialitzades.

En quant a les funcions d'aquesta clase, es limiten a cridar mètodes de les classes associades, que ja hem descrit a les altres descripcions.

3 Relació de classes

Pere Carrillo:

- ConjuntConsultesBooleanes
- ExpressioBooleana

Pol Garrido:

- EspaiVectorial

Marc Ordóñez:

- IdxAutors
- IdxObres

Laura Pérez:

- ConjuntDocuments
- DocumentPROP
- ControladorDomini

4 Estructures de dades i algorismes

4.1 Estructures de Dades

4.1.1 Espai Vectorial

L'espai vectorial ha estat la forma d'emmagatzemar els documents per la seva comparació, que ha proposat l'enunciat. Aquesta estructura ens permet guardar els documents com a una cadena de les seves paraules amb un pes associat. D'aquesta seqüència de paraules s'eliminen les *stopwords* i també es poden eliminar els nombres, les paraules que només apareixen en un document així com els signes de puntuació.

4.1.2 Trie

Un trie és un tipus d'arbre molt utilitzat per trobar cadenes de paraules.

Cada node d'un trie té assignat un caràcter i conté un punter per cada lletra que segueix la del node actual i un indicador de si la lletra actual és fi de paraula o no. El node arrel no correspon a cap lletra, els seus fills seran les primeres lletres de les paraules.

A l'hora de buscar les paraules es va mirant lletra per lletra, una paraula no existirà si per dues lletres consecutives a i b, b no és fill d'a, o si no està el final de paraula marcat.

Aquesta estructura de dades ens és molt útil a l'hora de buscar els autors que comencen per algun prefix.

4.1.3 Map

TreeMap

Els autors es guarden en un TreeMap. D'aquesta forma podem llistar als autors alfabèticament. La key de cada element del map serà el nom de l'autor i el value serà un altre TreeMap amb totes les obres d'aquell autor. En aquest segon TreeMap la key serà el títol de l'obra i el value serà el DocumentPROP amb la informació de l'obra d'aquell autor. Més endavant ha

LinkedHashMap

Aquesta implementació de map ens permet mantenir l'ordre d'addició facilitant el manteniment de l'espai vectorial pel que fa a la interrelació entre *Weights*, *TAS* i *Norms*. També s'utilitza al passar consultes de similaritat pel mateix motiu, mantenir la coherència i evitar posibles desparellaments de dades.

SortedMap

Aquest tipus de map ens permet ordenar les claus de forma creixent. Ens és útil la l'hora d'emmagatzemar els nodes del Trie, de manera que quan demanem els autors, ens els dona ordenats.

4.1.4 List

Una llista és una estructura molt útil si el que volem és iterar sobre un conjunt d'elements. Permet tenir elements repetits.

ArrayList

És una llista a la que se li pot modificar la mida, per tant, és molt útil per llistes a les que li volem afegir o eliminar elements, com a les expressions booleanes o a les frases d'un document.

4.1.4 Pair

Es tracta d'una classe auxiliar que ens ha permès simplificar algunes estructures de dades. Al no existir com a tal a *Java*, hem decidit implementar-la nosaltres amb un ús identic al seu homònim a C++.

4.2 Estructures de dades a les nostres classes

4.2.1 EspaiVectorial

- **List<LinkedHashMap<String,Pair<Double,Double>>> Weights;**
    - Aquesta estructura ens permet mantenir els pesos de les paraules de cada document assignats amb dos criteris, el booleà (si esta, un 1) i amb l'estratègia del tf-idf. D'aquesta manera podem respondre a les consultes de k documents semblants mitjançant dos criteris diferents.

- **List<Pair<Double,Double>> Norms;**
    - Aqui mantenim la norma dels vectors de pesos (els dos tipus) de tots els documents registrats, d'aquesta manera no s'ha de calcular cada vegada que ens fagin una consulta, sinó només quan es modifiqui un document.

- **List<Pair<String,String>> TAS;**
    - Aquesta estructura ens permet identificar a quin document pertanyen els pesos.

- **List<String> Stopwords;**
    - Aquí guardem la llista *d'stopwords* en català, castellà i anglès, per computar-les a l'espai vectorial, d'aquesta manera no hem de llegir el fitxer *d'stopwords* cada vegada que el necessitem.

4.2.2 DocumentPROP

- **String autor**
    - El nom de l'autor.
- **String titol**
    - El nom del títol.

- **String contingut**
    - El contingut tal com s'ha introduït.
- **LinkedHashMap<String, Integer> paraules**
    - Les paraules amb el nombre d'aparicions en el contingut, eliminant les que comencen per article + apòstrof.
- **ArrayList<String> frases**
    - Una llista de les frases, eliminant els signes de puntuació.
- **int numFrases**
    - El nombre de frases que hi ha en un document.

4.2.3 ConjuntDocuments

- **int numDocs**
    - Nombre de documents que tenim al nostre conjunt de documents.
- **IdxObres idxObres**
    - Tots els documents guardats identificats pel títol i l'autor.
- **IdxAutors idxAutors**
    - Els autors que tenim al nostre sistema.
- **EspaiVectorial espaiVectorial**
    - Els documents guardats amb pesos segons les paraules.

4.2.4 IdxAutors

- **TrieNode root**
    - Un Trie amb els noms dels autors.

4.2.5 IdxObres

- **Map<String, Map<String, DocumentPROP>> map**
    - Ens permet guardar els documents identificats per títol i autor. Al ser un TreeMap guarda els valors creixentment, de manera que quan vulguem llistar o realitzar alguna consulta se'ns retornaran els valors en ordre.

- **int numObres**
    - Nombre d'obres que tenim a l'índex. Ens serveix per fer càlculs a l'espai vectorial.

4.2.6 ExpressióBooleana

- **String expressió**
    - Expressió booleana formada pels operadors & | i !, conjunts de paraules (delimitats per {}), seqüències de paraules (delimitades per “”), o paraules soltes com a operands i parèntesis.
- **boolean negat**
    - Indica si l'expressió està negada.
- **ArrayList<Pair<Pair<String, String>, String>> resultat**
    - Ens permet guardar el títol, l'autor i la frase de tots els documents tals que la frase satisfà l'expressió.

4.2.7 ConjuntConsultes

- **int MIDA_HISTORIAL**
    - Mida màxima de l'historial.
- **LinkedList<ExpressioBooleana> historial**
    - Guardem les últimes “MIDA_HISTORIAL” consultes fetes per l'usuari, permetent reutilitzar les més recents. Necessitem poder afegir una consulta al principi i eliminar-ne la última de forma ràpida (com una cua), però també accedir a un element aleatori, per tant una LinkedList és la millor opció. Per mantenir la coherència cada cop que s'afegeix, elimina o modifica un document invalidem totes les expressions booleanes que tenim guardades a l'historial.
- **ArrayList<ExpressioBooleana> consultesGuardades**
    - Guardem totes les consultes que l'usuari ha demanat de guardar, en una estructura bàsica i senzilla com un ArrayList. Per mantenir la coherència cada cop que s'afegeix, elimina o modifica un document invalidem totes les expressions booleanes que tenim guardades.
- **ConjuntDocuments conjuntDocuments**
    - Necessitem una referència al conjunt de documents per poder fer les consultes.

4.3 Algorismes

4.3.1 Per a consultes booleanes

Tractament de les negacions

Per tractar les operacions NOT, cada expressió booleana es guarda un atribut que ens diu si està negada o no. Durant la funció treureNot(), si troba una NOT que afecta a tota l'expressió la treu i inverteix l'atribut negat. Això ens permet poder seguir operant amb l'expressió sense la NOT i saber si estava negada o no.

Algorisme per resoldre les consultes booleanes

L'estratègia que seguim per resoldre cada consulta es basa en Dividir i Vèncer. Dividim cada consulta en consultes més petites fins a arribar a un cas trivial i després anem ajuntant els resultats fins a obtenir de nou la consulta inicial.

Per començar intentem treure els parèntesis i les NOT que afectin a tota la consulta.

Comprovem si la consulta és bàsica, és a dir, conté només una paraula, un conjunt de paraules entre claudàtors o una seqüència de paraules entre cometes. Si ho és la resolem i retornem.

A continuació intentem separar-la en dos per una OR exterior, i si no n'hi ha cap, per una AND. Sempre ens trobarem en algun d'aquests dos casos, ja que si no, o la consulta era incorrecte o la hem resolt just abans.

Després cridem a resoldre les dues subconsultes de manera recursiva.

Finalment, si l'operador era una OR retornem la unió de les dues subconsultes, si era una AND retornem la intersecció. En el cas que l'expressió original estigués negada s'intercanvien les AND per OR i les OR per AND. Al separar la consulta de dos en dos ens assegurem que el resultat seguirà sent correcte tot i estar negada (Per les lleis de De Morgan).

4.3.2 Per a l'espai vectorial

Similaritat del cosinus

Aquesta representació dels documents com a vectors de pesos ens serveix per poder fer consultes de similaritat amb altres documents o amb conjunts de paraules. Per fer-ho utilitzem l'algorisme de similaritat del cosinus que surt de la idea de donar l'angle que separa dos documents, l'angle que ens representa com de diferents són dos conjunts de paraules.

$$\cos(\theta) = \frac{d \cdot q}{||d|| \cdot ||q||}$$

Aquesta és la fórmula amb la que treballem, s'utilitza el cosinus en comptes de l'angle, ja que és més fàcil de calcular. La d representa el vector de pesos del document, la p el vector de pesos de la *query* que pot ser un altre document i no té perquè utilitzar la mateix estratègia d'assignació de pesos. El producte del numerador es defineix d'aquesta manera:

$$d \cdot q = \sum_{i=1}^{N} w_{i,d} \cdot w_{i,q}$$

És a dir, el sumatori dels productes entre el pes de cada paraula al document i a la *query*. En el denominador tenim el producte de les normes que es calculen d'aquesta manera:

$$|q| = \sqrt{\sum_{i=1}^{N} w_{i,q}}$$

És a dir l'arrel quadrada de la suma dels quadrats dels pesos de la *query* o del document depenent de la norma que vulguem calcular.

En el nostre cas, computem la similaritat del cosinus per donar els k documents més similars a un donat o per donar els k documents més rellevants donat un conjunt de paraules. Aquestes consultes es poden fer amb l'estratègia d'assignació de pesos booleana o la de tf-idf pels documents, escollint a la hora de fer la consulta. El conjunt de paraules que son les consultes, se'ls hi assignen pesos amb l'estrategia booleana.

Tf-idf

Finalment parlem d'aquesta estratègia d'assignació de pesos que té en compte si una paraula és freqüent però també si ho és a través de tots els documents. D'aquesta manera paraules molt freqüents com "cosa" no obtindran tanta rellevància com si ho farien si només tinguéssim en compte la freqüència local a cada documents. Aquesta estratègia busca potenciar la importància de les paraules no tan freqüents que moltes vegades són les que ens permeten diferenciar el contingut de dos textos.

La fórmula per calcular els pesos és aquesta:

$$tf\_idf = tf(t,d) \cdot idf(t,D)$$

- $tf(t,d) = f_{t,d}$, es pot definir com el nombre de vegades que apareix una paraula al document , i ha diverses maneres de quantificar el *tf* però aquesta és la més àgil de calcular junt amb la booleana (1 si està 0 si no).
- $idf(t,D) = log\frac{N}{1+|\{d \in D : t \in d\}|}$, es defineix com el logaritme del quocient entre el nombre total de documents i el nombre de documents on apareix *t*. L'1 del denominador està per si el mot no apareix a cap document.

5 Tests

5.1 Conjunt Consultes Booleanes

Com que aquesta classe és una agregació de Expressions booleanes no farem mocks d'aquesta classe en concret.

5.1.1 Creadora

Comprovem que els atributs de la classe creats per la creadora són correctes.

5.1.2 Actualització Documents

Comprovem que s'anulen (posant a null el resultat) les consultes guardades per l'usuari i les de l'historial al cridar a actualitzacioDocuments().

5.1.3 Guarda Consulta Booleana

Comprovem que guardarConsultaBooleana() afegeix la consulta al conjunt de consultes de l'usuari.

5.1.4 Eliminar Consulta Booleana

Comprovem que eliminarConsultaBooleana() elimina la consulta del conjunt de consultes de l'usuari.

5.1.5 Modificar Consulta Booleana

Comprovem que modificarConsultaBooleana() modifica correctament la consulta que li passem.

5.1.6 Fer Consulta

Utilitzant un mock de ConjuntDocuments sobreescrivint les funcions getAllTitolsAutors i getFrasesDocument pels paràmetres que ens interessen, comprovem que el resultat de varies consultes és correcte.

5.1.7 Fer Consulta Incorrecta

Comprovem que al fer una consulta incorrecta salta l'excepció corresponent.

5.1.8 Resol Consulta

Comprovem que aquesta funció resol correctament una consulta. Com que la funció anterior utilitza aquesta, no cal comprovar molts més casos.

5.1.9 Compleix

Comprovem que la funció compleix() funciona per tots els casos.

5.1.10 Intersecció

Comprovem que la llista resultant és la intersecció de les dues llistes que passem com a paràmetres.

5.1.11 Unió

Comprovem que la llista resultant és la unió de les dues llistes que passem com a paràmetres.

5.2 Expressió Booleana

5.2.1 Creadora

Comprova que la creadora funciona correctament i inicialitza tots els atributs al que nosaltres volem.

5.2.2 Es Correcta

Comprovem tots els tipus diferents de consultes per assegurar-nos que aquesta funció funciona bé. Primer comprovem consultes correctes i després consultes incorrectes.

5.2.3 Char Normal

Comprovem, per molts casos diferents, que la funció charNormal() retorna el resultat esperat.

5.2.4 Es Simple

Comprovem si la funció esSimple() funciona adequadament.

5.2.5 Equals

Comprovem que si creem dos consultes amb el mateix nom i mateixa negació es consideren iguals. També comprovem que (a) i a es consideren la mateixa consulta.

5.2.6 Separa en dos

Comprovem que la funció separaEnDos() retorna el resultat correcte.

5.2.7 Elimina parèntesis inútils

Comprovem que s'eliminen correctament els parèntesis del principi i del final, però no els intermitjos.

5.2.8 Treure not

Comprovem que s'elimina correctament una NOT al principi, però no al mig.

5.3 Document PROP

5.3.1 Constructora buida

Comprovem que es crea el document correctament

5.3.2 Constructora amb paràmetres

Comprovem que la constructora crea correctament un document bàsic i que tots els atributs són els esperats.

5.3.3 Contar Paraules

A la primera part comprovem que s'eliminin correctament els espais innecessaris, noves línies i els signes de puntuació.
A la segona part comprovem que s'eliminin les paraules que comencen per apòstrof
A la tercera part comprovem que es contin correctament les paraules al tenir contingut buit

5.3.4 Separa Frases

La primera part mira un cas bàsic amb punts al final de cada frase.
La segona part comprova casos més extrems, amb caràcters com ! ¡ ¿ ? , . ;
La tercera part comprova que es llegeixi correctament quan el contingut és buit;

5.3.5 Te Paraula

Comprova les aparicions d'una paraula. Es miren casos com prefixos, amb espais i paraules que no estan al contingut.

5.3.6 Aparicions Paraula

Es comprova que hi hagi sobreposició de paraules.
No es comprova una paraula que no està ja que s'ha de fer la comprovació abans

5.4 Espai Vectorial

5.4.1 Constructora buida

En aquest test, revisem que es creïn les estructures de dades buides i que es llegeixi el document *d'stopwords* correctament.

5.4.2 Insertar document (ExceptionObraJaExisteix)

Ara revisem si al insertar un document a l'espai vectorial ho fa correctament. Revisem que els pesos s'han afegir correctament, així com el títol i l'autor del document per poder identificar-lo després. També revisem que s'afegeixi la norma del nou document. Apart d'això, en el contingut del document afegim una stopword per comprovar que no es tingui en compte.

En un test a apart, revisem que el cas extrem on s'intenta inserir un document ja existent, llenci l'excepció pertinent.

5.4.3 Afegir norma

En aquest test afegim diversos documents, revisant totes les estructures de dades que s'actualitzen i especialment que les normes s'afegeixin correctament

5.4.4 Actualitzar document (ExceptionObraNoExisteix)

Ara revisem que al actualitzar un document a l'espai vectorial, aquest modifiqui els atributs adients de les estructures de dades, per fer-ho afegim un document al l'espai i revisem els seus atributs. Després, l'actualitzem i tornem a revisar.

En un test a part revisem el cas extrem en que s'intenta modificar un document no registrat a l'espai vectorial. En aquest cas s'ha de llençar la excepció corresponent.

5.4.5 Eliminar document (ExceptionObraNoExisteix)

En aquest test revisem que tota la informació del document eliminat s'elimina també. Per fer-ho insertem un document que després esborrem. Per complicar-ho una mica més reinsertem el mateix document, l'actualitzem, i finalment l'eliminem. El resultat ha de ser un espai vectorial buit.

En un test a part revisem que en cas de intentar eliminar un document no registrat a l'estructura de dades es llenci l'excepció pertinent.

5.4.6 Similaritat de cosinus (Bool)

En aquest test comprovem la funcionalitat principal de l'espai vectorial que és resoldre les consultes de similaritat. En aquest primer test fem servir els pesos booleans dels documents. Per provar aquesta funció afegim 3 documents mantenint alguna similaritat dos a dos. Computem la similaritat i ha de retornar en primer lloc el document més similar al consultat.

5.4.6 Similaritat de cosinus (Tf-idf)

Finalment fem la mateixa comprovació que al test anterior, però variant el document consultat i utilitzant els pesos tf-idf dels documents.

5.5 Índex Autors

5.5.1 Afegir autors

Afegim molts autors que comencen pel mateix prefix i mirem si els llista en ordre alfabètic.

5.5.2 Eliminar autor

Es creen dos autors: patri i patricia. S'elimina patricia i es comprova que patri (prefix de patricia) segueix estant a l'índex. Després s'elimina a l'altre autor i es comprova que retorna un Array buit.

5.5.3 Eliminar autor no pertanyent

S'elimina a un autor que no pertanyi a l'índex i es comprova que salti l'excepció ExceptionAutorNoExisteix.

5.5.4 Llistar

S'afegeixen una sèrie d'autors i després es llisten. Els autors s'afegeixen en desordre alfabètic i comprovem que els llisti a tots i que els llisti en ordre.

5.5.5 Llistar prefix

S'afegeixen diversos autors que tinguin prefixos molt semblants per assegurar-se que llisti a partir del prefix i no de la lletra anterior. Per exemple si tenim patricia i paula i llistem per pat, no ens retorna paula. També comprova que si llistem per un prefix que contingui cap autor retorna una llista buida.

5.6 Índex Obres

5.6.1 Afegir obra

Afegeix una sèrie d'obres a un únic autor per comprovar que es crea l'autor i que s'afegeixen totes les obres. També s'afegeix una obra d'un altre autor per comprovar que es poden afegir diversos autors. Finalment, es comprova que el nombre d'obres afegides sigui el correcte.

5.6.2 Afegir obra repetida

S'assegura que quan vulguis afegir una obra a un autor que ja tingui un document amb aquell títol salti una excepció per no carregar-se la informació que ja tenies.

5.6.3 Afegir autor

Es comprova el correcte funcionament de la funció afegir autor. No s'utilitzarà per separat però cal assegurar-se que funciona. S'afegeix un nou autor i es comprova que hi hagi un key al map amb el nom de l'autor i que aquest autor s'hagi inicialitzat amb 0 obres.

5.6.4 Consulta obres autor

S'afegeixen una sèrie d'obres a un autor i es comprova que les llisti totes i que ho faci en ordre alfabètic. També s'aprofita per comprovar que si dos autors tenen el mateix títol de document no salta cap error, ja que un document s'identifica per autor i títol (pot haver dos obres amb el mateix títol però de diferent autor).

5.6.5 Consulta autor no existent

Es comprova que quan volem llistar les obres d'un autor i aquest autor no existeix salta l'excepció AutorNoExistent.

5.6.6 Get Document

El test s'assegura de que al afegir una obra es guardi el document que passem com a paràmetre i que quan cridem a la funció per obtenir el document ens retorni el document que havíem afegit anteriorment.

5.6.7 Get document però autor no existeix

Intenta obtenir el document d'un autor que no està a l'estructura de l'índex i fa saltar l'excepció de que l'autor no existeix.

5.6.8 Get document però obra no existeix

A l'igual que en el cas anterior, s'intenta obtenir el document d'un autor que sí que existeix però que no té cap obra amb el títol de la consulta i fa saltar l'excepció obra no existeix.

5.6.9 Elimina obra autor

Afegeix una sèrie d'obres i comprova que el número total d'obres sigui el correcte. Després elimina una obra i comprova que el nombre total d'obres hagi decrementat en una.

5.6.10 Elimina però autor no existent

Intenta eliminar l'obra d'un autor que no existeix i salta l'excepció autor no existent.

5.6.11 Elimina però obra no existent

Intenta eliminar l'obra d'un autor que existeix però l'obra no existeix i salta l'excepció obra no existent.

5.6.12 Modifica document

Es modifica el contingut d'un document. Primer es modifica el títol del document sense canviar l'autor i després es canvia l'autor del document i es comprova que s'ha creat un nou autor amb la nova obra (l'antiga canviada d'autor) i també es mira que l'autor antic ja no tingui l'obra modificada (ha deixat de ser seva).

5.6.13 Modifica document però autor es queda sense obres

Es comprova que si al modificar una sèrie de documents un autor deixa de tenir obres se l'ha d'eliminar de l'índex i si intentem mirar les obres que té aquest autor ens saltarà l'excepció de que no existeix l'autor perquè en el procés de modificar el contingut l'hem eliminat perquè ha deixat de tenir obres associades.