Gestor de Documents

Identificador de l'equip: 32.3

Joan Caballero Castro: joan.caballero@estudiantat.upc.edu

Jeremy Comino Raigón: jeremy.comino@estudiantat.upc.edu

Marc Gonzalez Vidal: marc.gonzalez.v@estudiantat.upc.edu

Oriol Miró López-Feliu: oriol.miro.lopez-feliu@estudiantat.upc.edu

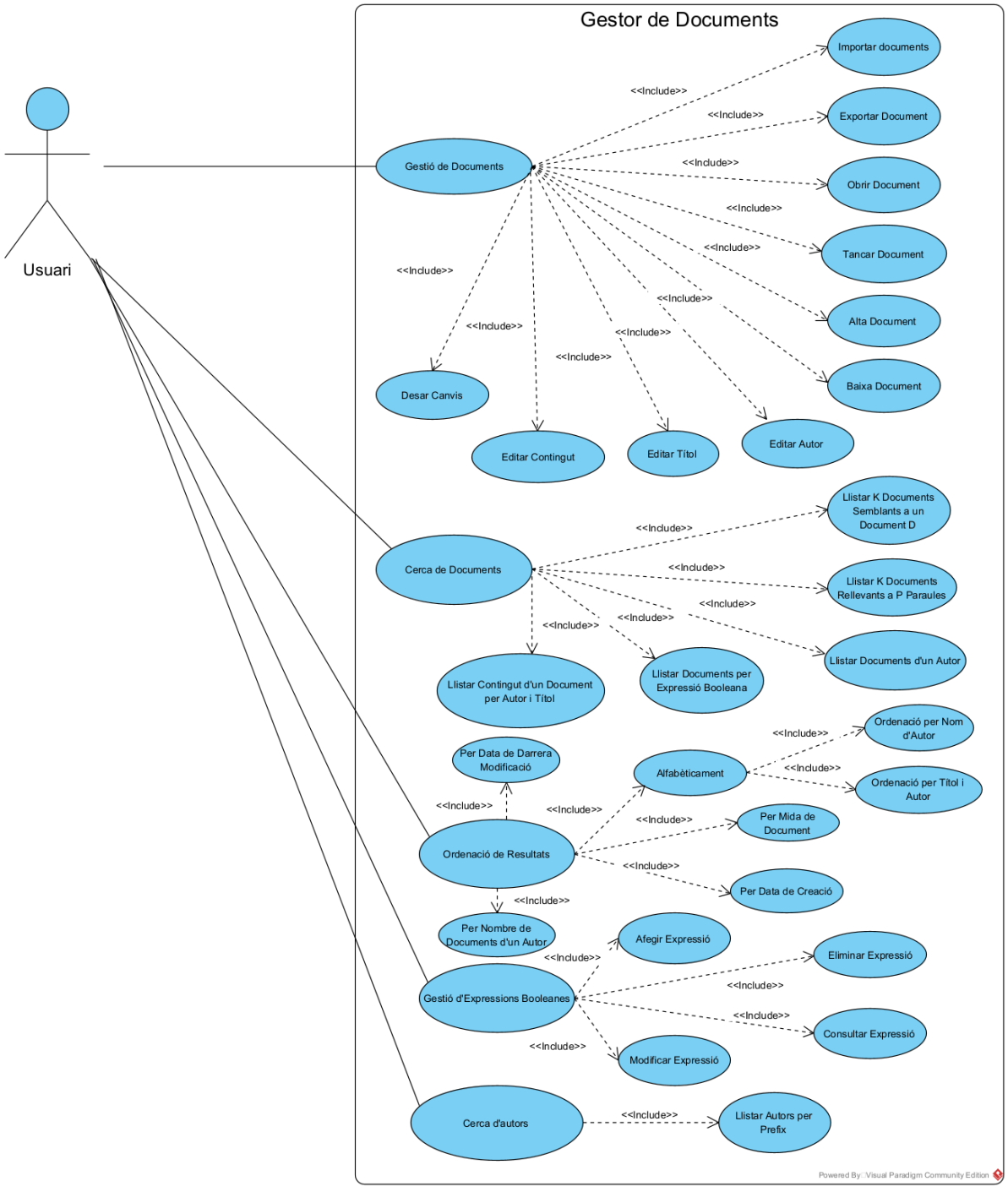
Versió del lliurament: 1.0

1ª entrega PROP

Índex

1. Diagrama de Casos d'Ús

A continuació mostrem el Diagrama de Casos d'Ús. Recomanem baixar el diagrama que es troba ubicat en el directori DOCS en format svg (DiagramaCasosDUS.svg) o pdf (DiagramaCasosDUS.pdf) per a poder veure-ho amb millor resolució.



1.1 Descripció Casos d'Ús

Gestió de Documents

- **Nom:** Importar Documents
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica al sistema que vol importar documents (en format text pla, xtml o pdf) i selecciona els documents a importar.
    - El gestor crea la representació interna del(s) document(s) importat(s).
- **Errors possibles i cursos alternatius:**
    - Un dels documents seleccionats és d'un tipus no reconegut pel gestor, és d'extensió invàlida o és corrupte. En aquest cas aquest document no s'importa i s'informa l'usuari de l'error.

- **Nom:** Exportar Document
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica al sistema que vol exportar un document a un cert format acceptat pel gestor (text pla, xtml o pdf).
    - El gestor processa el document de la representació interna a la representació en el format indicat per l'usuari.
    - El gestor guarda una còpia del document en el format seleccionat.
- **Errors possibles i cursos alternatius:**
    - No hi ha suficient espai per guardar la còpia o s'ha donat un error al processar un el document a exportar. En aquest cas el document no s'exportarà i s'informarà l'usuari de l'error.

- **Nom:** Alta Document
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica al sistema que vol donar d'alta un document, indicant el seu títol, autor i path.
    - El sistema dona d'alta el document i crea la seva representació interna.
- **Errors possibles i cursos alternatius:**

    - No hi ha suficient espai per guardar el document o ja existeix un document amb el mateix títol i autor al sistema o amb el mateix path, en aquest cas no es dona d'alta el document i s'informa l'usuari de l'error.

- **Nom:** Baixa Document
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica al sistema que vol donar de baixa un document i també indica si els vol esborrar del sistema operatiu.
    - El gestor deixa de reconèixer aquell document i la informació associada a aquest s'esborra.
- **Errors possibles i cursos alternatius:**
    - Si hi ha hagut un problema en esborrar un document del sistema operatiu, s'ha intentat donar de baixa un document que està obert pel mateix gestor o s'ha intentat donar de baixa un document no reconegut pel gestor o inexistent, no es donarà de baixa el document i s'informarà l'usuari de l'error.

- **Nom:** Obrir Document
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica al sistema que vol obrir un document mitjançant una direcció de l'ordinador (path).
    - El gestor mostra per pantalla el contingut del document.
- **Errors possibles i cursos alternatius:**
    - Si el Document seleccionat no forma part del gestor o és d'una extensió no reconeguda pel gestor, s'informarà l'usuari de l'error i no s'obrirà el document.

- **Nom:** Tancar Document
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica al sistema que vol tancar el document actualment obert.
    - El gestor tanca per pantalla el contingut del document, i guarda el seu contingut abans de tancar en cas que sigui necessari.
- **Errors possibles i cursos alternatius:**
    - No hi ha cap document actualment obert, en aquest cas s'informarà l'usuari que no s'ha pogut tancar el document.

- **Nom:** Editar Contingut
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - Precondició: Cas d'ús Obrir Document.
    - L'usuari modifica el contingut del document actualment obert.
    - El gestor crea la representació del nou document modificat.
- **Errors possibles i cursos alternatius:**
    - No hi ha cap document actualment obert, en aquest cas el gestor informa l'usuari de l'error.

- **Nom:** Editar Títol
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari selecciona un document i indica el seu nou títol.
    - El gestor sobreescriu aquesta informació sobre l'anterior.
- **Errors possibles i cursos alternatius:**
    - Ja existeix un document amb el mateix nou títol i autor, en aquest cas no es deixa editar el document i s'informa l'usuari de l'error.

- **Nom:** Editar Autor
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari selecciona un document i indica el seu nou autor.
    - El gestor sobreescriu aquesta informació sobre l'anterior.
- **Errors possibles i cursos alternatius:**
    - Ja existeix un document amb el mateix nou autor i títol, en aquest cas no es deixa editar el document i s'informa l'usuari de l'error.

Ordenació de Resultats

- **Nom:** Per Data de Creació (Opcional)
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol ordenar els resultats d'una cerca de documents per data de creació (creixentment o decreixentment).
    - El gestor mostra els documents ordenats per la seva data de creació.
- **Errors possibles i cursos alternatius:**

    - Cap.

- **Nom:** Per Data de Darrera Modificació (Opcional)
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol ordenar els resultats d'una cerca de documents per data de la darrera modificació (creixentment o decreixentment).
    - El gestor mostra els documents ordenats per la data de la darrera modificació.
- **Errors possibles i cursos alternatius:**
    - Cap.

- **Nom:** Per Mida de Document (Opcional)
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol ordenar els resultats d'una cerca de documents per la mida del document (creixentment o decreixentment).
    - El gestor mostra els documents ordenats per la seva mida.
- **Errors possibles i cursos alternatius:**
    - Cap.

- **Nom:** Alfabèticament per Títol i Autor
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol ordenar els resultats d'una cerca de documents prioritàriament per títol i secundàriament per nom d'autor (creixentment o decreixentment per títol).
    - El gestor mostra els documents ordenats alfabèticament per títol i nom d'autor.
- **Errors possibles i cursos alternatius:**
    - Cap.

- **Nom:** Alfabèticament per nom d'Autor
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol ordenar els resultats d'una cerca d'autors per nom d'autor (creixentment o decreixentment).

    - El gestor mostra els resultats ordenats alfabèticament per nom d'autor.
- **Errors possibles i cursos alternatius:**
    - Cap.

- **Nom:** Per Nombre de Documents d'un Autor (Opcional)
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol ordenar els resultats d'una cerca d'autors per nombre de títols d'un autor (creixentment o decreixentment).
    - El gestor mostra els autors ordenats pel nombre de títols d'un autor.
- **Errors possibles i cursos alternatius:**
    - Cap.

Gestió d'Expressions Booleanes

- **Nom:** Afegir Expressió
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol afegir una expressió booleana, donant el nom i el contingut de la nova expressió.
    - S'afegeix la nova expressió a la col·lecció d'expressions guardades.
- **Errors possibles i cursos alternatius:**
    - Ja existeix una expressió amb el mateix nom a la col·lecció, en aquest cas no s'afegeix la nova expressió i s'informa l'usuari de l'error.
    - L'expressió donada és errónea sintàcticament, en aquest cas no s'afegirà i s'informarà l'usuari de l'error.

- **Nom:** Modificar Expressió
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol modificar una expressió booleana ja existent, indicant el nom de l'expressió que vol modificar i la nova expressió booleana.
    - El gestor deixa l'usuari modificar el contingut de l'expressió seleccionada.
- **Errors possibles i cursos alternatius:**
    - No existeix cap expressió amb el nom indicat, en aquest cas no es modificarà l'expressió i s'informarà l'usuari de l'error.

    - El nou contingut de l'expressió és erroni sintàcticament, en aquest cas no es modificarà l'expressió i s'informarà l'usuari de l'error.

- **Nom:** Eliminar Expressió
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica que vol esborrar una de les expressions booleanes guardades.
    - El gestor elimina l'expressió seleccionada per l'usuari.
- **Errors possibles i cursos alternatius:**
    - No existeix l'expressió booleana seleccionada per l'usuari, en aquest cas no s'esborrarà cap expressió i s'informarà l'usuari de l'error.

- **Nom:** Consultar Expressió
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari indica el nom de l'expressió booleana que vol consultar.
    - El gestor mostra el contingut de l'expressió booleana a consultar.
- **Errors possibles i cursos alternatius:**
    - No existeix cap expressió amb el nom donat, en aquest cas no es mostrarà el seu contingut s'informarà l'usuari de l'error.

Cerca de Documents

- **Nom:** Llistar Documents d'un Autor
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari selecciona el nom d'un autor.
    - El sistema mostra tots els títols de l'autor.
- **Errors possibles i cursos alternatius:**
    - L'autor no existeix, per tant, s'informarà l'usuari i no es farà la consulta.

- **Nom:** Llistar Contingut d'un Document per Títol i Autor
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari selecciona un document donat el seu títol i nom d'autor.
    - El sistema mostra el contingut del document seleccionat.

- **Errors possibles i cursos alternatius:**
    - El document no existeix, per tant, no es farà la consulta i s'informarà l'usuari de l'error.

- **Nom:** Llistar K Documents Semblants a un Document D
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari selecciona un document D i un natural K.
    - El gestor mostra els K documents més semblants a D.
- **Errors possibles i cursos alternatius:**
    - Si hi ha menys de K documents donats d'alta al gestor, s'efectuarà la consulta, però es retornaran menys de K resultats.
    - Si el document D no existeix o és d'una extensió no reconeguda pel gestor, no es farà la consulta i s'informarà l'usuari de l'error.

- **Nom:** Llistar documents per Expressió Booleana
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari selecciona una expressió booleana mitjançant el seu nom o directament dona el contingut d'una expressió.
    - El gestor mostra una llista de documents que compleixen l'expressió booleana introduïda per l'usuari.
- **Errors possibles i cursos alternatius:**
    - Si dona l'expressió directament, pot ser invàlida sintàcticament i, per tant, s'informarà l'usuari i no es farà la consulta
    - Si es busca pel nom de l'expressió i aquesta no existeix, s'informarà l'usuari de l'error i no es farà la consulta.

- **Nom:** Llistar K Documents Rellevants P Paraules
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari selecciona els naturals K i P i un conjunt de P paraules.
    - El gestor mostra una llista dels K documents més rellevants al conjunt de paraules introduït.
- **Errors possibles i cursos alternatius:**
    - Si K és més gran que el nombre de documents donats d'alta al gestor, la consulta retornarà menys de K resultats.

    - Si K o P no són naturals s'informarà l'usuari de l'error i no s'efectuarà la consulta.

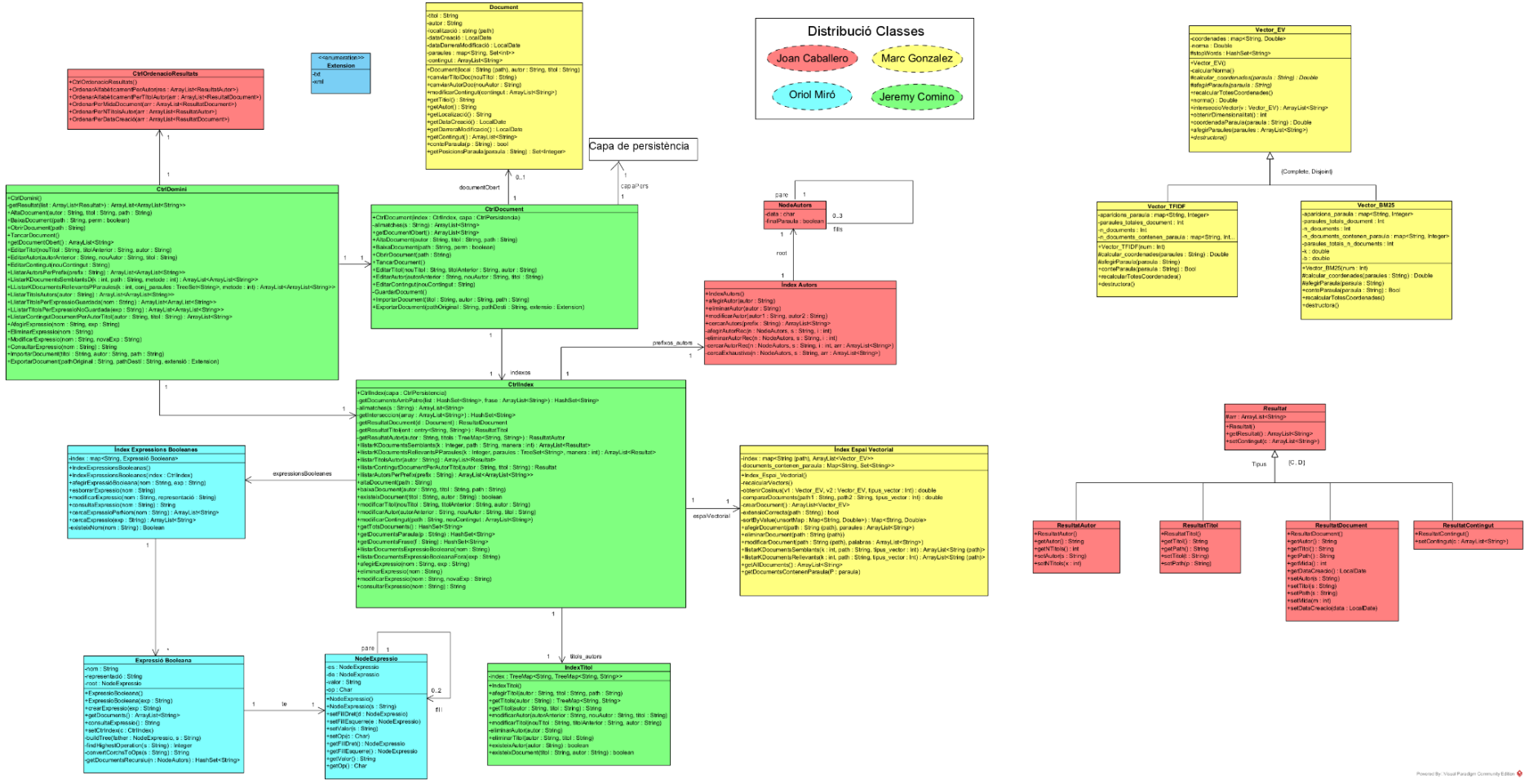
Cerca d'autors

- **Nom:** Llistar Autors per Prefix
- **Actor:** Usuari
- **Comportament (diàleg entre els actors i el sistema):**
    - L'usuari escriu al cercador d'autors un prefix.
    - El sistema mostra una llista amb els autors que comencen pel prefix.
- **Errors possibles i cursos alternatius:**
    - Cap.

2. Diagrama del model conceptual

2.1. Disseny del diagrama de model conceptual

A continuació mostrem el diagrama del model conceptual. Recomanem baixar el diagrama que es troba ubicat en el directori DOCS en format svg (DiagramaUML.svg) o pdf (DiagramaUML.pdf) per a poder veure-ho amb millor resolució.


Distribució Classes
Joan Caballero
Marc Gonzalez
Oriol Miró
Jeremy Comino
Capa de persistència
CtrlDomini
CtrlDocument
CtrlIndex
Document
Index Autors
Index Expressions Booleanes
Index Espai Vectorial
Expressió Booleana
NodeExpressio
IndexTitol
NodeAutors
Vector_EV
Vector_TFIDF
Vector_BM25
Resultat
ResultatAutor
ResultatTitol
ResultatDocument
ResultatContingut
documentObert
indexos
prefixos_autors
expressionsBooleanes
espaiVectorial
títols_autors
Tipus
(Complete, Disjoint)

2.2 Descripció de les classes

2.2.1 Document

La classe Document és la classe que conté la representació de tot el que necessita un document dins del nostre projecte per poder dur a terme totes les funcionalitats descrites a l'enunciat del projecte.

2.2.2 CtrlDomini

El controlador de Domini s'encarrega d'instànciar la resta de controladors de la capa de domini i proporciona una interfície més pròxima l'usuari dels mètodes de la resta de controladors com el controlador d'índex o de Document.

2.2.3 CtrlDocument

El controlador de document s'encarrega de tota la gestió de les funcionalitats on s'ha de fer alguna manipulació amb els documents, això inclou la donada d'alta de Documents, Baixa de Documents i canviar el contingut d'aquest. També es comunica amb el controlador de índex per poder introduir de manera correcta totes aquestes modificacions als índexs.

2.2.4 CtrlIndex

El controlador d'Index s'encarrega de totes les funcionalitats relacionades amb la consulta i modificació dels índexs.

2.2.5 CtrlOrdenacioResultats

El controlador CtrlOrdenacióResultats és l'encarregat d'ordenar els resultats de les consultes segons certs criteris.

2.2.6.IndexAutors

La classe IndexAutors s'encarrega de gestionar les cerques d'autors per prefix. Emmagatzema els noms dels autors en una estructura de dades enfocada en la cerca per prefixos, de tal manera que consultar autors és eficient tant en temps com espai.

2.2.7 IndexEspaiVectorial

Aquesta classe s'encarrega de representar l'espai vectorial al qual se li faran totes les consultes relacionades amb temes de similitud de documents i obtenir els documents que contenen una certa paraula. Utilitzar el model de l'espai vectorial es permet fer aquestes consultes d'una forma molt eficient.

2.2.8 IndexExpressionsBooleanes

Aquesta classe té la funció de preservar totes les expressions booleanes, per poder fer les consultes per aquestes. Es guarden associades a un nom, i es permet fer la cerca d'una expressió guardada o no guardada.

2.2.9 IndexTitol

Aquesta classe té la funció de preservar tots els autors i títols d'aquest per poder fer les consultes dels títols d'autors. A més a més, per cada autor i títol s'adjunta la seva direcció de l'ordinador, per poder mostrar-ho l'usuari en diferents consultes.

2.2.10 ExpressioBooleana

ExpressioBooleana és la classe que s'encarrega de gestionar les consultes per expressió booleana. Per poder realitzar aquesta funcionalitat de manera eficient hem decidit usar un arbre binari com a estructura de dades.

2.2.11 NodeAutors

NodeAutors és una classe auxiliar per la classe IndexAutors. En ella s'emmagatzema informació necessària fer efectuar la cerca d'autors per prefix.

2.2.12 NodeExpressio

NodeExpressio és una classe auxiliar per la classe ExpressioBooleana. En ella s'emmagatzema la informació necessària per efectuar la cerca per expressions booleana.

2.2.13 Vector_EV

És una classe abstracta que ens permet representar un document amb el model espai vectorial. Aquesta classe, com és abstracta, té per mètodes el mínim nombre de funcions que ha de tenir qualsevol estratègia de representació dels vectors, que es vulgui implementar.

2.2.14 Vector_TFIDF

Aquesta classe hereda de Vector_EV, i implementa tots els mètodes necessaris per implementar l'estratègia de pesos del TFIDF (representació obligatòria del projecte).

2.2.15 Vector_BM25

Aquesta classe hereda de Vector_EV, i implementa tots els mètodes necessaris per implementar l'estratègia de pesos del BM25 (representació opcional del projecte).

2.2.16 Resultat

Resultat és una classe que s'utilitza per a poder fer l'ordenació dels resultats de consultes segons certs criteris. Té un total de 4 subclasses: ResultatAutor, ResultatTítol, ResultatDocument i ResultatContingut, on cadascuna d'elles conté la informació necessària per poder realitzar les seves respectives consultes.

3. Relació de les classes implementades per cada membre de l'equip

A continuació mostrem una taula amb les classes que ha implementat cada membre del grup.

Joan Caballero	Jeremy Comino	Marc Gonzalez	Oriol Miró
CtrlOrdenacioResultats	CtrlDomini	Document	IndexExpressionsBooleanes
IndexAutors	CtrlDocument	IndexEspaiVectorial	ExpressioBooleana
NodeAutors	CtrlIndex	Vector_EV	NodeExpressio
Resultat + Subclasses	IndexTitol	Vector_TFIDF	
DriverDomini		Vector_BM25	

4. Estructures de dades i algorismes utilitzats

4.1 Estructures de dades

A continuació explicarem per cada classe que conforma el domini, les estructures hem considerat. Tot això amb l'objectiu de crear el nostre gestor de documents.

4.1.1 Document

El document és l'estructura que té tota la informació relacionada amb un Document. Aquesta classe té principalment 3 atributs importants:

- L'autor del Document
- El títol del Document
- El contingut del Document

En el nostre cas, el principal problema que vam tenir era trobar una estructura de dades que fos relativament assequible de modificar i que no ocupi una gran quantitat de memòria. Tot això per guardar el contingut del Document.
Per tant, vam pensar en aquestes 3 alternatives:

- **Path del Document associat en el sistema operatiu**
  Aquesta tècnica consisteix en l'operació directament amb el document del sistema operatiu que conté tota la informació associada a un document del gestor. D'aquesta manera, en el gestor no teníem un document que té un pes molt gran en memòria. Únicament teníem en memòria la seva direcció en el sistema operatiu (path).
  El principal problema d'aquesta tècnica era que trenca el principi de 3 capes, on cada capa és independent de l'altre, en aquest cas la capa de Domini sap com s'està guardant en el sistema els documents. A més a més, tractar amb el contingut és difícil, ja que no és una estructura dinàmica. És a dir, si volguéssim introduir una nova paraula entremig del nostre contingut actual, primer hauríem de moure tot el contingut posterior a aquesta nova paraula i després introduir aquesta nova paraula.
  Per tant, aquest mètode no és el millor, encara que el cost en memòria sigui de les millors opcions.
- **String que conté tot el contingut.**
  Aquesta estructura té cost ***O(n)*** sent ***n*** el nombre de caràcters que conté el contingut del Document. Per tant, el principal problema d'aquesta estructura és un dels mencionats anteriorment. Aquest és la modificació quan hi ha paraules a afegir o eliminar. Per tant, vam optar que en comptes de modificar una string. Sempre que l'usuari vulgui fer una modificació del contingut, sempre ens doni una string amb el contingut total.

- **ArrayList de Strings**
  Aquesta és l'estructura que hem utilitzat com a representació del contingut d'un document. Usarem també com a tècnica de modificació del contingut, la mencionada anteriorment. La raó és degut al fet que d'aquesta manera, el tractament del contingut per part de l'índex de l'espai vectorial es pot fer de manera òptima.

4.1.2 IndexAutors

IndexAutors és la classe que s'encarrega de gestionar les consultes d'autors per prefix. Per poder realitzar aquesta funcionalitat de manera eficient hem decidit utilitzar un Trie com a estructura de dades.
Un Trie és una estructura de dades en forma d'arbre n-ari de cerca. La idea bàsica de qualsevol Trie és muntar un arbre de manera que cada node representa un prefix dels seus fills. D'aquesta manera es poden emmagatzemar de manera eficient paraules (com strings).

Característiques dels Trie

- Permeten operacions eficients per prefixos, a diferència d'altres estructures de dades.
- Són flexibles, permetent afegir i esborrar dades guardades.
- Poden emmagatzemar de forma molt eficient en memòria i temps paraules o seqüències que comparteixen prefixos.

Funcionament

Ja que en el projecte hem implementat un arbre de cerca ternari (o TST per a abreujar), explicarem el comportament d'aquest. Encara que cal destacar que el seu funcionament és semblant a altres tipus de trie i, per tant, es podria generalitzar la idea bàsica.

L'arbre estarà compost per diferents nodes. Cadascun amb els següents atributs:

- La dada a emmagatzemar (per exemple, un caràcter de la paraula).
- Un punter a cadascun dels seus 3 nodes fills (denominats left, mid i right).
- Un booleà que indica si el node marca el final d'una paraula o seqüència.

La manera de guardar paraules dins de l'arbre serà mitjançant els nodes centrals o mid.
Així doncs, només estarem adquirint informació addicional sobre la seqüència quan avancem pel fill central en el recorregut de l'arbre. Si, en canvi avancem pels fills left i right no haurem afegit informació sinó que haurem passat a fer la cerca en un dels subàrbres dels fills.

Cost

Creació

Crear un TST solament implica crear un node arrel buit, amb cost O(1).

Cerca

Quan busquem en un arbre ternari podem diferenciar dues operacions: Primer, recorrem k (on k és la grandària en caràcters de la nostra paraula o seqüència) vegades els fills centrals de nodes de l'arbre per a formar la paraula en qüestió. Segon, fem cerques recursives en els subàrbres esquerre i dret de cada node (per a trobar el prefix que necessitem) amb cost logarítmic en funció del nombre d'autors guardats n.
Per tant, el cost d'aquesta operació serà O(log(n) + k) en arbres degudament equilibrats.

Inserció

El procés d'inserció és molt semblant al de cerca, ja que simplement es fa un recorregut de l'arbre com s'ha explicat anteriorment i s'assumeix que la paraula ja es troba dins d'aquest. Si tots els nodes ja existeixen de manera que es pugui fer la paraula, es marca l'últim com a final de paraula. En cas contrari, es creen nous nodes a mesura que es vagin necessitant (amb cost O(1)) fins a formar l'estructura de nodes requerida per a representar la paraula.
Per tant, el cost serà O(log(n) + k) en cas mitjà i O(n + k) en cas pitjor.

Esborrat

De la mateixa forma que la inserció, l'esborrat també depèn fortament de l'operació de cerca, perquè esborrar un element de l'arbre suposa buscar si està i la seva posició i després processar qualsevol d'aquests 4 possibles casos:

- La paraula no es troba en l'arbre: No es fan canvis en aquest. Cost O(1)
- La paraula es troba en l'arbre i és prefix d'altres paraules: Simplement es posa a fals l'atribut que marca final de paraula. Cost O(1)
- La paraula es troba en l'arbre i té altres paraules com a prefix: S'eliminen recursivament els nodes que formen la paraula, començant pel final fins a arribar a un node final del prefix o esgotar el subàrbre que la conté. Cost O(k)
- La paraula es troba en l'arbre i no és prefix ni té prefixos: S'eliminen recursivament tots els seus nodes, començant pel final fins a esgotar el subàrbre que la conté. Cost O(k).

Com que els casos a processar tenen cost menor al de cerca i es fan seqüencialment després d'executar-la, el cost final és el de fer una cerca: O(log(n) + k) en cas mitjà i O(n + k) en cas pitjor.

4.1.3 IndexTitol

IndexTitol és la classe que s'encarrega de gestionar les consultes de contingut donat un autor i un títol i donar tots els títols d'un autor. Per tant, hem optat per utilitzar un TreeMap<String,TreeMap<String,String>> com que que d'aquesta manera ens assegurem que totes les consultes siguin ***O(log(n))*** sent ***n*** el nombre d'autors que hi ha en l'índex.

4.1.4 IndexEspaiVectorial

IndexEspaiVectorial és la classe que ens permet representar documents dins d'un espai vectorial i fer diferents operacions que ens són interessants per saber per exemple, d'una forma relativament eficient, el grau de coincidència d'un document a un altre o d'una query als documents. Els atributs d'aquesta classe són índex, que és un map<String,ArrayList<Vector_EV>> que ens permet obtenir en un temps O(log(n) el Vector d'un document. És un ArrayList, ja que hi ha més d'una estratègia de pesos implementada per tant retorna els Vectors en les diferents representacions, en aquest cas 2. També guardem a un map<String, ArrayList<String>> els documents que contenen una paraula, per poder fer les cerques amb temps O(log(n)).

4.1.5 IndexExpressioBooleana

IndexExpressioBooleana és la classe que ens permet guardar les representacions de les expressions booleanes. Per tal de fer això, disposem d'un índex, utilitzant l'estructura de dades HashMap.

Funcionament

Aquest índex relaciona el nom de les expressions booleanes amb la seva representació, és a dir, un string amb una instància de la classe ExpressioBooleana.

Justificació de l'elecció de l'ED

Hem decidit implementar l'índex amb aquesta ED, ja que suposa una complexitat temporal molt petita per totes les operacions que necessitem efectuar.

Complexitat

Les operacions que es requereixen del l'índex són: afegir una expressió, modificar-la, esborrar-la, consultar-la o cercar per expressió (ja sigui donant aquesta directament o donant el nom d'una guardada).

Afegir

Afegir en un HashMap té un cost mitjà de O(1), pese a que en el pitjor cas es O(k), sigui k la quantitat d'elements del HashMap. També s'ha de tenir en compte el temps per crear la representació de l'expressió booleana, estudiat al apartat ExpressioBooleana, que es O(n), sigui n la longitud de la string que representa l'expressió booleana, i per tant el cost d'afegir es O(n) en mitjà i O(n*k) en pitjor cas.

Modificar

Modificar suposa el mateix cost que afegir, deut al funcionament intern de les classes, i per tant en temps mitjà es O(n) i en en el pitjor cas O(n*k).

Esborrar

Esborrar suposa un cost de O(1) en temps mitjà, i O(k) en el pitjor cas.

Consultar

Consultar suposa un cost de O(1) en temps mitjà, i O(k) en el pitjor cas.

Cercar

En la cerca, es requereix distinguir dos casos: cerca donant l'expressió i cerca donant el nom d'una guardada. En ambdós casos caldrà una cerca, amb la complexitat documentada a ExpressioBooleana sent O(n*m), prenent m com la quantitat de documents que hi han.

- Donant l'expressió: Hem de primer crear l'expressió (temps O(n)) i després cercar-la (O(n*m)). Per tant el cost total sería O(n^2*m).
- Donant el nom: Hem de primer buscar l'expressió (consultar-la, O(1) en temps mitjà, O(k) en pitjor cas) i després cercar-la (O(n*m)). Per tant, el cost total serà O(n*m) en cas mitjà i O(k*m*n) en pitjor cas.

4.1.6 ExpressioBooleana

ExpressioBooleana és la classe que s'encarrega de gestionar les consultes per expressió booleana. Per poder realitzar aquesta funcionalitat de manera eficient hem decidit utilitzar un Arbre Binari com a estructura de dades. Ens guardem l'arrel de dit arbre a la classe com atribut.

Funcionament

L'arbre estarà compost per diferents nodes. Cadascun amb els següents atributs:

- Un punter a cadascun dels seus 2 nodes fills (esquerre i dret).
- Una string que, en cas de que el node actual sigui fulla, conté un predicat.
- Un char que indica de quin tipus és el node:
    - 0: Node fulla i valor es el predicat.
    - ": Node fulla i valor es una frase a buscar
    - !: Node operació '!', només el fill dret té valor.
    - & o |: Node operació '&' o '|', ambdòs fills tenen valor.

Llavors a l'hora de crear l'arbre, busquem l'operació amb màxima precedència i actuem d'acord amb quina és aquesta, donant valor al tipus de node i si cal, donat valor a l'atribut valor i/o cridant recursivament a la funció pel sub-arbre dret i esquerre.

Justificació d'elecció de la ED

Hem escollit aquesta estructura de dades com que és una representació completa, i permet realitzar fàcilment les consultes sub-partint el problema.

Complexitat

Com que l'ED per representar una constant, l'expressió booleana definida per la classe, només tenen sentit les operacions de creació i cerca, ja que les altres es gestionen des de l'índex, sobre les instàncies d'aquesta classe.

<u>Creació</u>

Sigui n el mida de la string que representa l'expressió booleana, el cost de transformar-la en arbre es O(n)

Cerca

Quan busquem tots els documents que compleixen expressió booleana definida per la classe, hem de fer com a molt O(n) crides a subarbres. Tot i això, l'operació més costosa és la intersecció. Hem utilitzat un HashSet per anar "pujant" recursivament els resultats per l'arbre, i el cost d'intersecció amb HashSet és O(m), sigui m la quantitat de documents que hi ha. Per tant, el cost de cerca és O(n*m), però en la majoria de casos serà molt millor.

4.1.7 Vector_EV

El Vector_EV conté el mínim nombre de mètodes que necessitem per representar un document en el model d'espai vectorial. Cada herència d'aquesta classe implementa la seva estratègia de pesos. Utilitzant aquesta classe abstracta ens és molt fàcil afegir noves estratègies de pesos al nostre projecte. Només caldria fer una herència i implementar els diferents mètodes que necessiti. Els atributs que conté són un map<String,Double> on hi haurà les coordenades de cada paraula i podrem consultar la coordenada d'una paraula en temps O(logn). També té la norma del Vector i finalment unes stopWords que són paraules que s'han d'ignorar.

4.1.8 Vector_TFIDF

El Vector_TFIDF és una herencia de Vector_EV implementa una estratègia de pesos utilitzant el model del TFIDF. Per fer-ho, d'atributs es guarda les aparicions d'una paraula dins d'un map<String, Integer> i el nombre de paraules totals que hi ha al document, aquesta part ens serveix per calcular el TF. Per calcular l'IDF tenen 2 atributs static, compartits per totes les classes, un que és el nombre de documents (l'haurem d'incrementar en 1 cada vegada que afegim un nou document i també el nombre de documents que contenen una paraula, aquesta part és per calcular l'idf. He decidit fer-ho amb els static, ja que volia que la responsabilitat de calcular les coordenades d'un vector sigui el mateix vector i no l'espai vectorial.

4.1.9 Vector_BM25

El Vector_BM25 és una herència de Vector_EV implementa una estratègia de pesos utilitzant el model del BM25. Com podem veure comparteix tots els atributs amb Vector_TFIDF i a més afegim les paraules totals que hi ha als n documents i dos paràmetres que s'usaran per calcular les coordenades (k i b).

5. Testing

5.1 Test DriverDomini

testAltaDocument

En aquest test provem els mètodes relacionats amb les altes dels documents, que són afegirTitol(autor,z titol, path) de la classe IndexTitol, afegirAutor(autor) de la classe IndexAutors i afegirDocument(path, paraules) de la classe IndexEspaiVectorial.

Introduïm el nom de l'autor, títol i path del nou document a crear i si les dades introduïdes són correctes podrem donar d'alta el document.

```
S'està provant l'alta d'un nou document.
Introdueix el nom de l'autor: Joan
Introdueix el títol: titolJoan
Introdueix el path: pathJoan
S'ha donat d'alta el nou document correctament.
```

Figura 1: testAltaDocument() amb dades correctes.

En canvi, si introduïm les dades de tal manera que ja existeixi un document amb el mateix autor i títol o amb el mateix path veurem com ens salten errors.

```
S'està provant l'alta d'un nou document.
Introdueix el nom de l'autor: Joan
Introdueix el títol: titolJoan
Introdueix el path: pathJoan2
ERROR: Ja existeix un document amb títol titolJoan i autor Joan.
```

Figura 2: testAltaDocument() amb mateix títol i autor.

```
S'està provant l'alta d'un nou document.
Introdueix el nom de l'autor: Marc
Introdueix el títol: titolMarc
Introdueix el path: pathJoan
ERROR: Ja existeix un document amb path pathJoan.ojmj.
```

Figura 3: testAltaDocument() amb mateix path.

testBaixaDocument

En aquest test provem els mètodes relacionats amb l'eliminació de documents, que són eliminarTitol(autor, titol) de la classe IndexTitol, eliminarAutor(autor) de la classe IndexAutors i eliminarDocument(path) de la classe IndexEspaiVectorial.

Si introduïm el path d'un document existent veurem com s'elimina satisfactòriament del gestor.

```
S'està provant la baixa d'un document
Introdueix el path del document a eliminar: pathjoan
Vols eliminar el document completament del Sistema Operatiu? si/no: si
El document pathjoan s'ha eliminat correctament.
```

Figura 4: testEliminarDocument() amb path existent.

En canvi, si intentem eliminar un document que no existeix ens saltarà l'error.

```
S'està provant la baixa d'un document
Introdueix el path del document a eliminar: pathJoan2
Vols eliminar el document completament del Sistema Operatiu? si/no: no
ERROR: El document pathJoan2 no existeix.
```

Figura 5: testEliminarDocument() amb path inexistent.

testObrirDocument

En aquest test provem el mètode d'obtenir un document de la Capa de Persistència mitjançant el CtrlDocument amb la funció ObrirDocument(path).

Si introduïm un document existent veurem com ens deixa obrir-lo, però en canvi si intentem obrir un document que no existeix veurem com ens salta l'error.

```
S'està provant el mètode d'obrir un document.
Insereix el path del document a obrir: pathjoan
El document pathjoan.ojmj s'ha obert correctament.
```

Figura 6: testObrirDocument() amb path existent.

```
S'està provant el mètode d'obrir un document.
Insereix el path del document a obrir: pathJoan2
ERROR: No existeix cap document amb path pathJoan2.ojmj.
```

Figura 7: testObrirDocument() amb path inexistent.

testTancarDocument

En aquest test provem el mètode TancarDocument() de la classe CtrlDocument. Veiem com si tanquem un document obert tot és correcte però si cridem a aquesta funció sense cap document obert ens salta l'error.

```
S'està provant el mètode de tancar el document actualment obert.
El document pathjoan.ojmj s'ha tancat correctament.
```

Figura 8: testTancarDocument() amb document obert.

```
S'està provant el mètode de tancar el document actualment obert.
ERROR: No hi ha cap document obert.
```

Figura 9: testTancarDocument() sense document obert.

testEditarAutor

En aquest test provem els mètodes relacionats amb l'edició de l'autor d'un document, que són eliminarAutor(autor) i afegirAutor(autor) de la classe IndexAutors i modificarAutor(autorAnterior, nouAutor, titol) de la classe IndexTitol.

Si introduïm les dades correctament veiem com ens deixa editar l'autor del document, en canvi si volem editar un document inexistent o el volem modificar a un document amb títol i autor ja existent ens saltarà error.

```
S'està provant l'edició de l'autor d'un document.
Introdueix el títol del document: titolJoan
Introdueix el nom de l'autor del document a modificar: Joan
Introdueix el nom del nou autor del document: Jeremy
L'autor del document s'ha canviat correctament a Jeremy.
```

Figura 10: testEditarAutor() amb dades correctes.

```
S'està provant l'edició de l'autor d'un document.
Introdueix el títol del document: titoljoan
Introdueix el nom de l'autor del document a modificar: Marc
Introdueix el nom del nou autor del document: Oriol
ERROR: No existeix un document amb títol titoljoan i autor Marc.
```

Figura 11: testEditarAutor() amb document inexistent.

```
S'està provant l'edició de l'autor d'un document.
Introdueix el títol del document: titoljoan
Introdueix el nom de l'autor del document a modificar: Jeremy
Introdueix el nom del nou autor del document: Joan
ERROR: Ja existeix un document amb títol titoljoan i autor Joan.
```

Figura 12: testEditarAutor() a un document que ja existeix.

testEditarTitol

En aquest test provem el mètodes relacionat amb l'edició del títol d'un document, que és modificarTitol(nouTitol, titolAnterior, autor) de la classe IndexTitol.

Si introduïm les dades correctament veiem com ens deixa editar el títol del document, en canvi si volem editar un document inexistent o el volem modificar a un document amb títol i autor ja existent ens saltarà error.

```
S'està provant l'edició del titol d'un document.
Introdueix el nou nom del document: titoljoan2
Introdueix el nom antic del document: titoljoan
Introdueix el nom de l'autor del document: Joan
El títol del document s'ha canviat correctament a titoljoan2.
```

Figura 13: testEditarAutor() amb dades correctes.

```
S'està provant l'edició del titol d'un document.
Introdueix el nou nom del document: titolOriol
Introdueix el nom antic del document: titolMarc
Introdueix el nom de l'autor del document: Joan
ERROR: No existeix un document amb títol titolMarc i autor Joan.
```

Figura 14: testEditarAutor() amb document inexistent.

```
S'està provant l'edició del titol d'un document.
Introdueix el nou nom del document: titoljoan
Introdueix el nom antic del document: titolJoan2
Introdueix el nom de l'autor del document: Joan
ERROR: Ja existeix un document amb títol titoljoan i autor Joan.
```

Figura 15: testEditarAutor() a un document que ja existeix.

testEditarContingut

En aquest test provem els mètodes relacionats amb la modificació del continguts del document obert. Si tenim un document obert veurem com ens deixa modificar el seu contingut i després mostra els canvis satisfactòriament. En canvi si no tenim cap document obert ens saltarà l'error.

```
3
S'està provant el mètode d'obrir un document.
Insereix el path del document a obrir: pathjoan
El document pathjoan.ojmj s'ha obert correctament.
7
S'està provant la modificació del contingut del document actualment obert.
Insereix el nou contingut:
Marc Oriol Jeremy Joan
El contingut del document pathjoan.ojmj s'ha modificat correctament.
14
S'està provant la consulta d'un document donat el seu títol i autor.
Introdueix el nom de l'autor: Joan
Introdueix el nom del títol: titoljoan
Marc Oriol Jeremy Joan
```

Figura 16: testEditarContingut() amb document obert.

```
7
S'està provant la modificació del contingut del document actualment obert.
Insereix el nou contingut:
Aquest és el nou contingut del document.
ERROR: No hi ha cap document obert.
```

Figura 17: testEditarContingut() sense document obert.

testLlistarAutorsPerPrefix

En aquest test provem el mètode cercarAutors(prefix) de la classe IndexAutors. Per tal de provar aquesta funcionalitat primer creem un conjunt de documents amb diferents autors i després realitzem la cerca per prefix. En la imatge es poden veure com la cerca per prefix ens retorna el conjunt d'autors corresponents.

Autors afegits: Joan, Marc, Jeremy, Oriol.

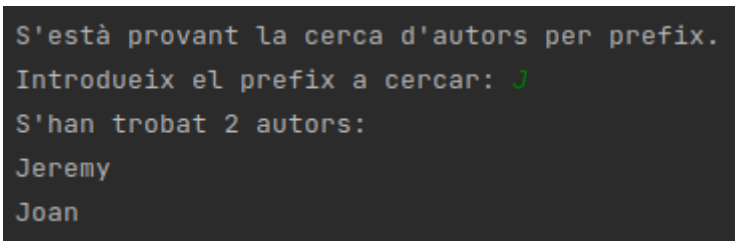


Figura 18: testLlistarAutorsPerPrefix() amb prefix.

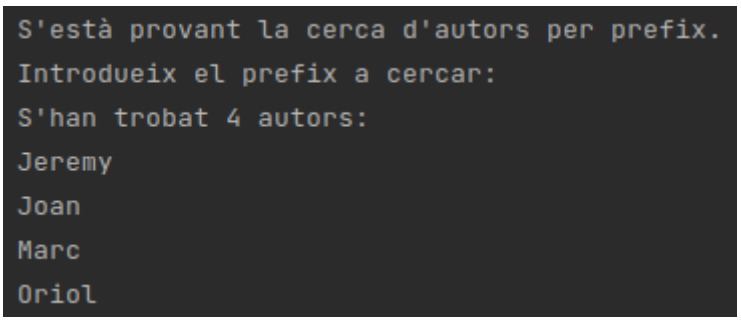


Figura 19: testLlistarAutorsPerPrefix() amb prefix buit.

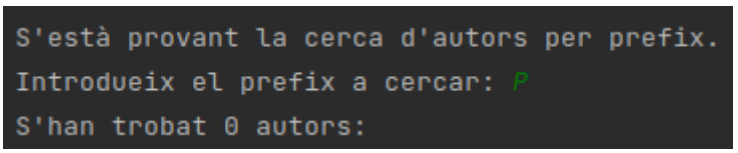


Figura 20: testLlistarAutorsPerPrefix() sense cap resultat.

testLlistarKDocumentsSemblantsD

En aquest test provem el mètode LListarKDocumentsSemblantsD. Per això, hem creat 2 Documents. Un Document tindrà com a atributs “JEREMY”, “PROP”, “p.ojmj”, “hola com estas”. L’altre Document tindrà com a atributs “JOAN”, “PROP”, “pp.ojmj”, “molt be”.
Farem la consulta amb k = 2, Document = “p.ojmj”.

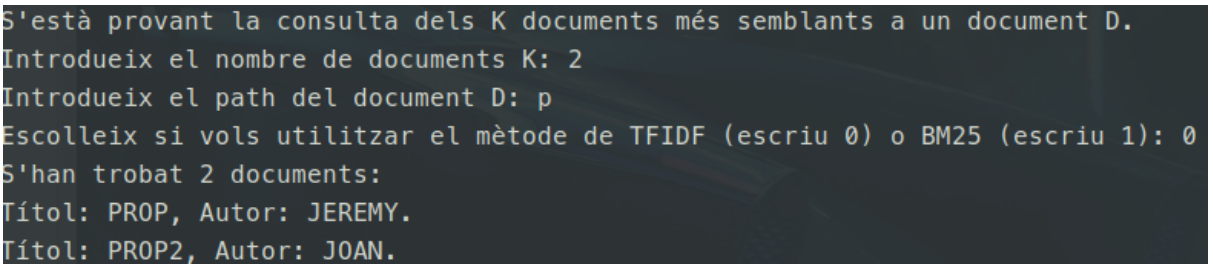


Figura 21: Resultat de la consulta amb TFDF

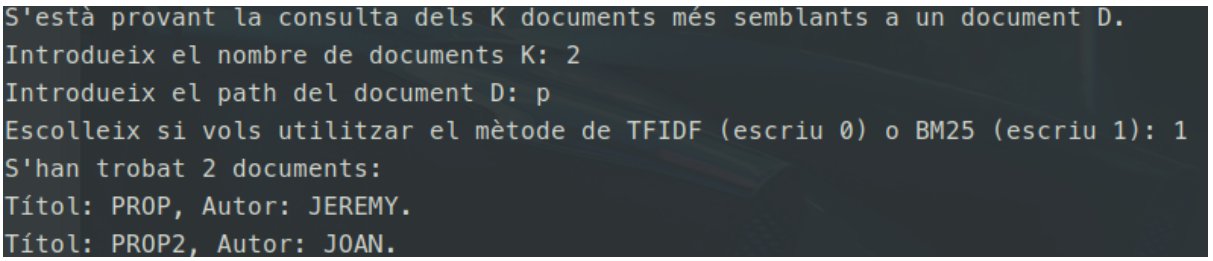


Figura 22: Resultat de la consulta amb BM25

Com es pot apreciar en l'anterior figura, com en aquest cas existeixen sols 2 Documents al gestor. Mostrarà tot el contingut del gestor ordenat per la seva semblança. El primer Document és el propi i el segon que no té gaire bé semblança. És el document de PROP2

testLlistarKDocumentsRellevantsPParaules

En aquest test provem el mètode LListarKDocumentsDonadesPParaules. Hem utilizat a part dels Documents utilitzats en el test anterior, hem afegit el document "MARC", "PROP3", "ppp.ojmj", "si si si si". I hem fet la consulta dels amb k = 2, les paraules són {hola, molt}.

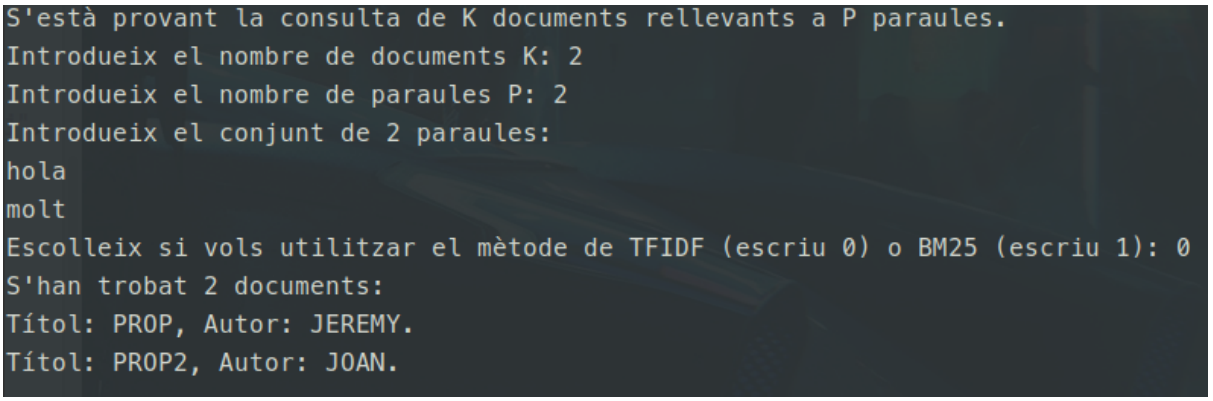


Figura 23: Resultat de la consulta amb TFIDF.

Com es pot veure en la figura anterior sols els documents que tenen hola i molt són els que surten degut al valor k que hem donat.

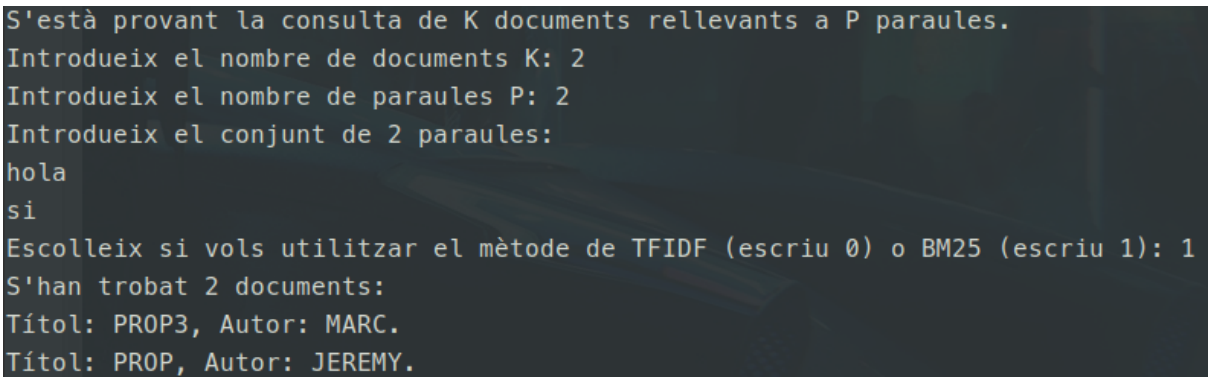


Figura 24: Resultat de la consulta amb BM25.

En aquesta consulta hem volgut veure si agafavem com a valors el hola i el si. Com es pot veure, surt primer el document de MARC degut a que en la totalitat de les paraules, la paraula que més surt és “si”, fent que sigui la primera en sortir. Posteriorment, surt el document JEREMY degut a que és l’únic document que conté hola en el seu contingut.

testLlistarTitolsAutors

En aquest test provem el mètode llistarTitolsAutor(autor) de la classe CtrlIndex. Per tal de provar aquesta funcionalitat primer creem un conjunt de documents per a l’autor “joan” amb mateix títol i path {a, b, c} i per l’autor “oriol” {d, e}.

Si fem la consulta per aquests dos autors veiem com el resultat que ens donen és l’esperat. En canvi, si fem la consulta per un autor que no existeix veiem com ens dona error.

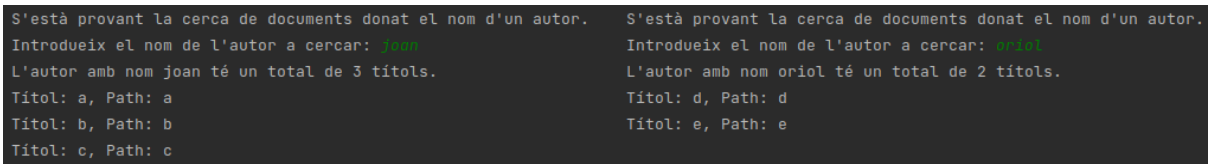


Figura 25: testLlistarTitolsAutors() amb autors existents.

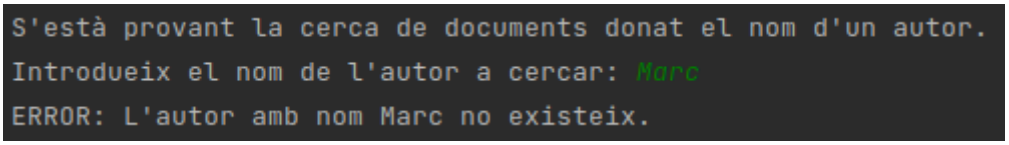


Figura 26: testLlistarTitolsAutors() amb un autor inexistent.

testLlistarTitolsPerExpressioGuardada

En aquest test provem el mètode cercaExpressioPerNom(nom) de la classe IndexExpressionsBooleanes. Primer creem uns documents amb contingut i després fem la consulta i comprovem que els resultats són correctes.

Per provar aquest test primer hem creat dos documents: un amb contingut “hola mama adeu papa” i l’altre amb “hola avia o avi”. A continuació afegim una sèrie d’expressions booleanes amb el mètode afegirExpressio(nom, exp) i després fem les seves consultes i veiem com el resultat que rebem és el desitjat.

```
S'està provant l'addició d'una nova expressió.
Introdueix el nom de la nova expressió: exp1
Introdueix la nova expressió: {hola} & !tieta
Expressió afegida correctament.
12
S'està provant la cerca de documents donat el nom d'una expressió booleana guard
ada.
Introdueix el nom de l'expressió a cercar: exp1
S'han trobat 2 resultats:
Títol: b, Autor: b.
Títol: a, Autor: a.
```

Figura 27: testLlistarTitolsPerExpressioGuardada amb resultat

.

```
S'està provant l'addició d'una nova expressió.
Introdueix el nom de la nova expressió: exp2
Introdueix la nova expressió: !hola
Expressió afegida correctament.
12
S'està provant la cerca de documents donat el nom d'una expressió booleana guard
ada.
Introdueix el nom de l'expressió a cercar: exp2
S'han trobat 0 resultats:
```

Figura 28: testLlistarTitolsPerExpressioGuardada() sense resultat.

testLlistarTitolsPerExpressioNoGuardada

En aquest test provem el mètode cercaExpressio(exp) de la classe IndexExpressionsBooleanes. Primer creem uns documents amb contingut i després fem la consulta i comprovem que els resultats són correctes.

Hem utilitzat els mateixos documents i consultes que en el test anterior i, si els comparem amb els del test anterior, podem veure com ens donen el mateix resultat.

```
Introdueix l'expressió a cercar: {hola} & !tieta
S'han trobat 2 resultats:
Títol: b, Autor: b.
Títol: a, Autor: a.
```

Figura 29: testLlistarTitolsPerExpressioNoGuardada amb resultat.

```
S'està provant la cerca de documents donada una expressió booleana introduïda pe
r l'usuari.
Introdueix l'expressió a cercar: !hola
S'han trobat 0 resultats:
```

Figura 30: testLlistarTitolsPerExpressioNoGuardada sense resultat.

testLlistarContingutDocumentPerAutorTitol

En aquest test provem el mètode getTitol(autor, titol) de la classe IndexTitol i getDocument(path) de la Capa de Persistència. Primer creem una sèrie de documents amb diferents autors i títols i després comprovarem que les consultes ens retornen els resultats esperats.

Documents creats (títol, autor, contingut): (titoljoan, joan, joan s'en va a dormir), (titoloriol, oriol, oriol s'ha despertat).

```
S'està provant la consulta d'un document donat el seu títol i autor.
Introdueix el nom de l'autor: joan
Introdueix el nom del títol: titoljoan
joan s 'en va a dormir
```

Figura 31: testLlistarContingutDocumentPerAutorTitol correcte (1).

```
S'està provant la consulta d'un document donat el seu títol i autor.
Introdueix el nom de l'autor: oriol
Introdueix el nom del títol: titoloriol
oriol esta despert
```

Figura 32: testLlistarContingutDocumentPerAutorTitol correcte (2).

```
S'està provant la consulta d'un document donat el seu títol i autor.
Introdueix el nom de l'autor: jeremy
Introdueix el nom del títol: titoljeremy
ERROR: No existeix cap document amb títol titoljeremy i autor jeremy.
```

Figura 33: testLlistarContingutDocumentPerAutorTitol sense resultat..

testAfegirExpressió

En aquest test provem el mètode afegirExpressio(nom, exp) de la classe IndexExpressionsBooleanes. Introduïm un nom i contingut correctes per a l'expressió booleana i aquesta s'afegeix correctament.

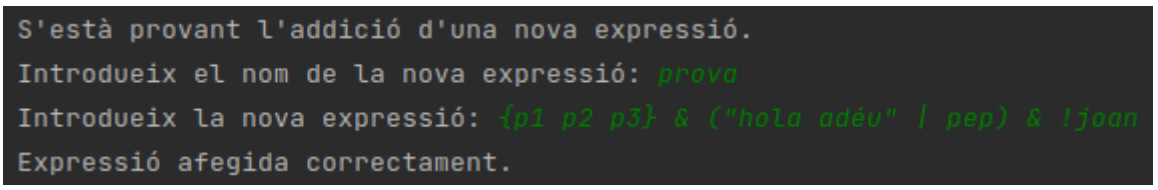


Figura 34: afegirExpressio() amb dades correctes.

Si ara intentem afegir una altra expressió amb el mateix nom o amb un contingut incorrecte ens sortiran excepcions advertint-nos que hem introduit dades incorrectes.

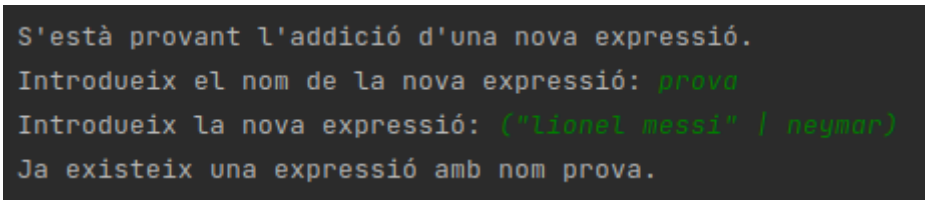


Figura 35: afegirExpressio() amb nom invàlid.

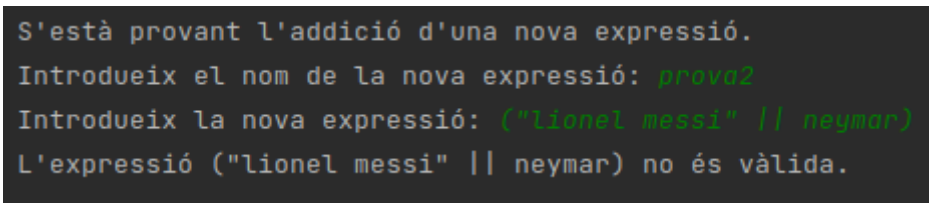


Figura 36: afegirExpressió amb contingut invàlid.

testEliminarExpressio

En aquest test provem el mètode borrarExpressio(nom) de la classe IndexExpressionsBooleanes. Introduïm el nom d'una expressió booleana existent i veiem com aquesta s'elimina correctament.

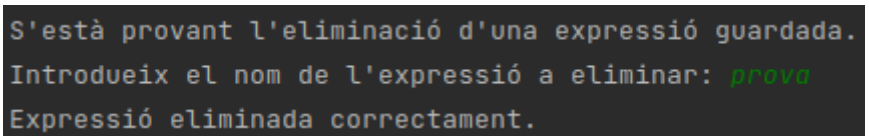


Figura 37: borrarExpressio() amb nom existent.

Si en canvi intentem eliminar una expressió que no existeix veurem com ens salta l'excepció.

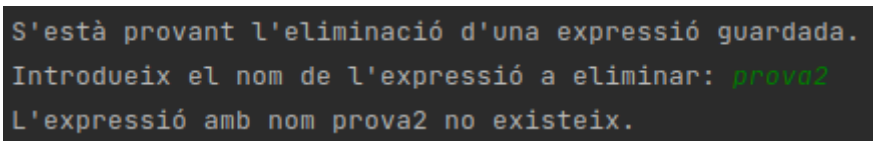


Figura 38: borrarExpressio() amb nom inexistent.

testModificarExpressio

En aquest test provem el mètode modificarExpressio(nom, exp) de la classe IndexExpressionsBooleanes. Introduïm el nom d'una expressió booleana existent i el seu nou contingut i veiem com aquesta s'ha modificat correctament.

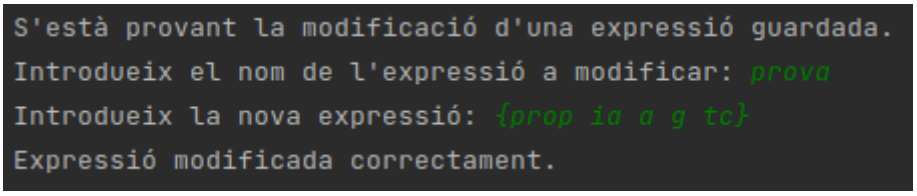


Figura 39: modificarExpressio() amb dades vàlides.

Si en canvi intentem modificar una expressió inexistent o amb un contingut incorrecte veurem com ens salta l'excepció corresponent en cadascun dels dos casos.

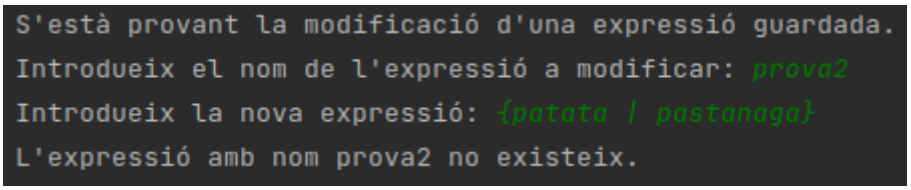


Figura 40: modificarExpressio() amb nom invàlid.

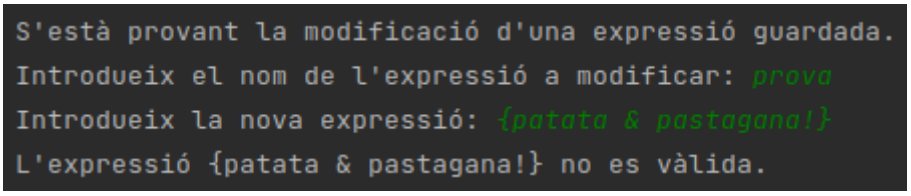


Figura 41: modificarExpressio() amb contingut invàlid.

testConsultarExpressio

En aquest test provem el mètode consultarExpressio(nom) de la classe IndexExpressionsBooleanes. Introduïm el nom d'una expressió booleana existent i veurem el seu contingut correctament.

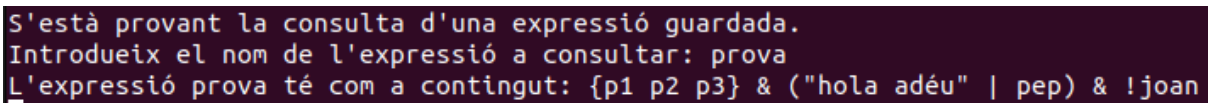


Figura 42: testConsultarExpressio() d'una prova existent.

En canvi si intentem consultar una expressió que no existeix veurem el missatge d'error.

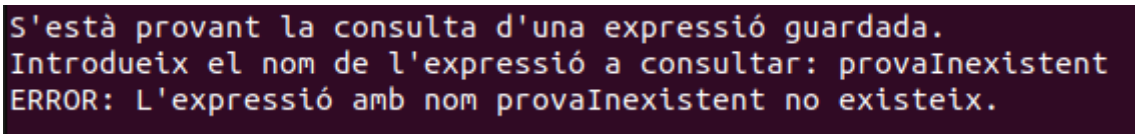


Figura 43: testConsultarExpressio() d'una prova inexistent.

5.2 Test Document

TestCanviarTítolDocument

En aquest test creem un document amb títol "prova de prop" i posteriorment cridem al mètode CanviarTítolDocument i posem "Planetes Del Sistema Solar", posteriorment, al fer un getTítol, hem d'obtenir el títol modificat i no el que hi havia primerament.

TestCanviarAutorDocument

En aquest test creem un document amb autor "prova de prop" i posteriorment cridem al mètode CanviarAutorDocument i posem "Jeremy", posteriorment, al fer un getAutor, hem d'obtenir l'autor modificat i no el que hi havia primerament.

TestCanviarAutorDocument

En aquest test creem un document amb autor "prova de prop" i posteriorment cridem al mètode CanviarAutorDocument i posem "Jeremy", posteriorment, al fer un getAutor, hem d'obtenir l'autor modificat i no el que hi havia primerament.

TestModificarContingut

Aquest test crea un nou document i modifica el contingut dues vegades, comprovant que el contingut que s'ha quedat guardat dins la classe es la de l'última modificació.

TestGetTitol

GetTitol simplement es un getter del títol, comprovem que en retorni el títol que li hem passat a la constructora i ja.

TestGetAutor

GetAutor simplement es un getter de l'autor, comprovem que en retorni l'autor que li hem passat a la constructora.

TestGetLocalització

GetLocalització simplement es un getter de la localització (path) de l'arxiu, comprovem que ens retorni la localització que li hem passat a la constructora i ja.

TestGetDataCreació

GetDataCreació es un getter de la data de creació de l'arxiu que va ser calculada al crear l'arxiu. Comprovem que ens retorni la data actual i ja.

TestGetDataDarreraModificació

GetDataDarreraModificació es un getter de la data de la darrera modificació de l'arxiu que per tant per comprovar-ho simplement modifiquem un arxiu i mirem que la data sigui la mateixa que la actual.

//Observació

És possible que aquests dos últims tests fallin si es fan prop de les 23:59. No fer durant aquest període.

TestGetContingut

Aquest mètode és un getter del contingut que te un document i ho retorna en forma de llista.

TestGetContingut1

Simplement comprovem que un document al qual no li hem afegit contingut retorna una llista buida.

TestGetContingut2

Comprovem que un document guarda bé el contingut de unes quantes paraules.

TestGetContingut3

Comprovem que si fem una modificació el contingut és l'última modificació que fem.

TestContéParaula

Comprovem que un document guarda bé el contingut de unes quantes paraules i a més ho fem amb modificacions comprovant si conte o no paraules.

TestGetPosicionsParaula1

Cridem a la funció get posicions paraula d'un vector buit, de manera que ens a de retornar una llista buida.

TestGetPosicionsParaula2

Cridem a la funció get posicions paraula per a un vector amb diferents elements i comprovem que les paraules siguin correctes.

5.3 Test IndexEspaiVectorial

Com Index Títol és una classe que no depèn d'altres. No necessitarem stubs ni mocks per fer els tests.

testAfegirDocument

En aquest test farem farem tests amb diferents casos que pot tenir el mètode AfegirDocument.

testAfegirDocument1

Afegim un document "/Disc/prova.ojmj" sense paraules i comprovem que l'únic document que hi ha dins de l'espai vectorial és aquest.

testAfegirDocument2

Afegim un document "/Disc/prova.ojmh" i comprovem que ens llença l'excepció de format no reconegut, ja que es .ojmh en lloc de .ojmj.

testAfegirDocument3

Afegim un document "/Disc/provaojmj" i comprovem que ens llença l'excepció de format no reconegut, ja que es ojmj en lloc de .ojmj.

testAfegirDocument4

En aquest cas intentem afegir un document que ja havia estat creat anteriorment. Per tant, esperem que es llenci l'excepció DocumentJaExisteix.

testAfegirDocument5

Afegim dos documents amb diferent path i comprovem que s'hagin afegit els 2 correctament.

testAfegirDocument6

Afegeixo un document, en aquest cas amb text per comprovar que no hi ha cap error i que s'està afegint correctament.

testEliminarDocument

En aquest cas comprovem diferents casos per la funcionalitat EliminarDocument().

testEliminarDocument1

En aquest primer cas, afegeixo un document i després l'elimino. Posteriorment comprovo que l'espai vectorial estigui buit, és a dir, que s'hagi eliminat correctament el document.

testEliminarDocument2

En aquest cas intento eliminar un document que no existeix, per tant ha de saltar l'excepció DocumentNoExisteix.

testEliminarDocument3

En aquest cas, intento eliminar un document que té un format incorrecte, enlloc de ser .ojmj es .ojmh, per tant ha de saltar l'excepció FormatNoReconegut. En cas que no ho comprobessim també saltaria que el document no existeix, però ens ha semblat més interessant FormatNoReconegut, ja que en un futur se li podrà indicar l'usuari que s'ha equivocat a l'hora d'introduir el format.

testEliminarDocument4

En aquest cas, intento eliminar un document que té un format incorrecte, enlloc de ser .ojmj es ojmj, per tant ha de saltar l'excepció FormatNoReconegut

testEliminarDocument5

En aquest cas comprovo borrar un document quan dins de l'espai vectorial hi ha més d'un document. Per comprovar que es borra el correcte.

testModificarDocument

En aquest cas provarem la funcionalitat de ModificarDocument.

testModificarDocument1

Per provar aquesta funcionalitat, el primer test que farem serà el d'afegir un document sense contingut, i posteriorment el modificarem per a poder afegir-li contingut. Per fer la comprovació, mirarem d'obtenir els documents que tingui una paraules de les que hem afegit posteriorment.

testModificarDocument2

En aquest cas intentem modificar un document que no existeix, per tant s'ha de llençar l'excepció DocumentNoExisteix.

testModificarDocument3

En aquest cas modifiquem un document amb una ExtensioNoReconeguda i comprovem que ens llença l'excepció de format no reconegut, ja que es .ojmh en lloc de .ojmj.

testModificarDocument4

En aquest cas modifiquem un document amb una ExtensioNoReconeguda i comprovem que ens llença l'excepció de format no reconegut, ja que es ojmj en lloc de .ojmj.

testModificarDocument5

En aquest cas, afegim un document amb paraules i posteriorment modifiquem aquest contingut, posteriorment demanem al índex espai vectorial que ens retorni els documents que contenen una de les paraules que s'ha afegit al fer la modificació.

testModificarDocument6

En aquest cas afegeixo 2 documents i modifico el segon, preguntant per una paraula que anteriorment estava als 2 documents però que després de la modificació només ha d'estar a 1.

TestLlistarKDocumentsSemblants

En aquests test provarem la funcionalitat LlistarKDocumentsSemblants.Fins que no es digui el contrari, utilitzarem l'estratègia de pesos TFIDF.

TestLlistarKDocumentsSemblants1

En el primer test comprovem que el número de documents que se'ns passa per paràmetre es correcte, en aquest cas és més gran que el número de documents que conté, per tant, hem de retornar l'excepció NumDocumentsIncorrecte.

TestLlistarKDocumentsSemblants2

En el primer test comprovem que el número de documents que se'ns passa per paràmetre es correcte, en aquest cas passem un nombre de documents negatiu, per tant, hem de retornar l'excepció NumDocumentsIncorrecte.

TestLlistarKDocumentsSemblants3

En aquest cas preguntem per una representació que no existeix del vector, ja que recordem que només tenim dos representacions op == 0 implica TFIDF i op == 1 implica BM25, i en aquest cas preguntem per la tercera implementació op == 2, i per tant no està implementada i ha de retornar l'excepció RepresentacioVectorNoExisteix.

TestLlistarKDocumentsSemblants4

En aquest cas preguntem per una representació que no existeix del vector op == -1 com que això no és una representació vàlida, ha de retornar l'excepció RepresentacioVectorNoExisteix.

TestLlistarKDocumentsSemblants5

En aquest cas preguntem per obtenir els documents semblants a un document que no existeix, per tant hem de retornar l'excepció DocumentNoExisteix.

TestLlistarKDocumentsSemblants6

En aquest cas preguntem per obtenir els k documents que es semblen a un document que no conté una extensió vàlida, per tant a de retornar ExtensioNoReconeguda, ja que es .ojmh en lloc de .ojmj.

TestLlistarKDocumentsSemblants7

En aquest cas preguntem per obtenir els k documents que es semblen a un document que no conté una extensió vàlida, per tant a de retornar ExtensioNoReconeguda, ja que es ojmj en lloc de .ojmj.

TestLlistarKDocumentsSemblants8

En aquest cas afegim 3 documents, on dos d'aquests tenen paraules compartides. Per tant, preguntem per un dels documents que contenen paraules repetides i per k = 1 i ens ha de retornar l'altre document amb el qual es comparteixen paraules.

TestLlistarKDocumentsSemblants9

Es comprova que la funció no retorna cap document per molt que l'espai vectorial en contingui si k = 0.

TestLlistarKDocumentsSemblants10

En aquest test és comprova que funcioni correctament amb 3 documents, amb k = 2.

TestLlistarKDocumentsSemblants11

Replica de TestLlistarKDocumentsSemblants8 amb estratègia BM25

TestLlistarKDocumentsSemblants12

Replica de TestLlistarKDocumentsSemblants9 amb estratègia BM25

TestLlistarKDocumentsSemblants13

Replica de TestLlistarKDocumentsSemblants10 amb estratègia BM25

TestLlistarKDocumentsRellevants

En aquests test provarem la funcionalitat LlistarKDocumentsRellevants.Fins que no es digui el contrari, utilitzarem l'estratègia de pesos TFIDF.

TestLlistarKDocumentsRellevants1

En el primer test comprovem que el número de documents que se'ns passa per paràmetre es correcte, en aquest cas és més gran que el número de documents que conté, per tant, hem de retornar l'excepció NumDocumentsIncorrecte.

TestLlistarKDocumentsRellevants2

En el primer test comprovem que el número de documents que se'ns passa per paràmetre es correcte, en aquest cas passem un nombre de documents negatiu, per tant, hem de retornar l'excepció NumDocumentsIncorrecte.

TestLlistarKDocumentsRellevants3

En aquest cas preguntem per una representació que no existeix del vector, ja que recordem que només tenim dos representacions op == 0 implica TFIDF i op == 1 implica BM25, i en aquest cas preguntem per la tercera implementació op == 2, i per tant no està implementada i ha de retornar l'excepció RepresentacioVectorNoExisteix.

TestLlistarKDocumentsRellevants4

En aquest cas preguntem per una representació que no existeix del vector op == -1 com que això no és una representació vàlida, ha de retornar l'excepció RepresentacioVectorNoExisteix.

TestLlistarKDocumentsRellevants5

En aquest cas afegim 3 documents, i fem una query per "document", i obtenim l'únic arxiu que la tenia com a paraula.

TestLlistarKDocumentsRellevants6

En aquest cas comprovem que si dos paraules de la query són la mateixa, el resultat que retorna és el mateix.

TestLlistarKDocumentsRellevants7

Replica de TestLlistarKDocumentsRellevants5 amb estratègia BM25.

TestLlistarKDocumentsRellevants8

Replica de TestLlistarKDocumentsRellevants6 amb estratègia BM25.

TestGetAllDocuments

En aquest cas provarem el mètode GetAllDocuments.

TestGetAllDocuments1

En el primer cas fem un get quan no hi ha cap document, de manera que ens a de retornar un llista buïda.

TestGetAllDocuments2

En el segon test afegim un document i fem un get de tots els documents per veure si ens torna aquest document.

TestGetAllDocuments3

En el tercer test afegim 3 documents i comprovem que se'ns retorni tots 3.

TestGetDocumentsContenenParaula

En aquest cas farem un testeig del mètode que ens permet obtenir tots els documents que contenen una paraula.

TestGetDocumentsContenenParaula1

Primer que tot comprovem la funció fent un get d'un espai vectorial que no conté elements, de manera que ha de retornar una llista buida.

TestGetDocumentsContenenParaula2

En aquest test afegeixo un document amb la paraula "hola" i posteriorment, els busco per veure si retorna el document que acabo d'afegir.

TestGetDocumentsContenenParaula3

En aquest test afegeixo dos documents amb la paraula “hola” i posteriorment, els busco per veure si retorna els documents que acabo d’afegir.

TestGetDocumentsContenenParaula4

En aquest cas afegeixo 3 documents dels quals 2 contenen la paraula “hola” i posteriorment faig un get dels Documents que tenen “hola”, això ho faig per comprovar que no hem retorna el test que té la paraula “adeu”.

5.4 Test IndexExpressionsBooleanes

TestAfegirExpressio

Test en el que es prova afegir una expressió booleana a l’índex,on el nom i l’expressió indicats són vàlids.

TestAfegirExpressioNomInvalid

Test en el que es prova afegir una expressió booleana no vàlida, degut a que el nom és invàlid, on salta l’excepció JaExpressioExisteix.

TestAfegirExpressioInvalida

Test en el que es prova afegir una expressió booleana no vàlida, degut a que l'expressió és invàlida, on salta l’excepció ExpressioNoValida.

TestModificarExpressioCorrecte

Test en el que es prova modificar una expressió booleana de manera que el nom indicat i l’expressió nova siguin vàlides..

TestModificarExpressioNomInvalid

Test en el que es prova modificar una expressió booleana incorrectament, degut a que el nom és invàlid, on salta l’excepció NoExisteixExpressio.

TestModificarExpressioExpInvalida

Test en el que es prova modificar una expressió booleana incorrectament, degut a que el la nova expressió indicada no és vàlida, on salta l’excepció ExpressioNoValida.

TestBorrarExpCorrecte

Test en el que es prova a esborrar una expressió booleana de l'índex correctament, donant un nom vàlid.

TestBorrarExpNomInvalid

Test en el que es prova a esborrar una expressió booleana del índex incorrectament, degut a que el nom és invàlid, on salta l'excepció NoExisteixExpressio.

TestConsultaExpCorrecte

Test en el que es consulta una expressió booleana del índex correctament, donant un nom vàlid.

TestConsultaExpNomInvalid

Test en el que es prova a consultar una expressió booleana de l'índex incorrectament, degut a que el nom és invàlid, on salta l'excepció NoExisteixExpressio.

TestCercaExpressioValidaPerNom

Test en el que es cerca una expressió booleana del índex correctament, donant un nom vàlid.

TestCercaExpressioNomInvalid

Test en el que es prova a cercar una expressió booleana del índex incorrectament, degut a que el nom és invàlid, on salta l'excepció NoExisteixExpressio.

TestCercaExpressioVàlida

Test en el que es cerca una expressió booleana explicitada correctament, donant una expressió vàlida.

TestCercaExpressioInvalida

Test en el que es cerca una expressió booleana incorrectament, degut a que l'expressió explicitada no és vàlida, on salta l'excepció ExpressioNoValida.

5.5 Test IndexTitol

Com Index Títol és una classe que no depèn d'altres, no necessitarem stubs ni mocks per fer els tests.

testAfegirTitol

En aquest test, el que es vol provar és si la inserció d'un títol funciona. Donat un autor, títol i path, veurem si després de la inserció existeix la inserció donada. S'utilitza com a autor "Juan", "El MAN" com a títoli"EL WOMAN.ojmj" com el path del Document.

testGetTitols

En aquest test es vol comprovar si en fer una consulta dels títols d'un autor no existent, llença l'excepció NoExisteixAutor. Com a valor introduït, utilitzarem els anteriors mencionats per introduir-los en l'índex, però farem una consulta sobre el nom "maria" fent que s'executi l'excepció.

testGetTitols2

És el mateix test anterior però canviant l'autor a cercar. En aquest cas s'utilitzarà com a autor de cerca l'introduït en l'índex (en aquest cas és "Juan"). Per tant, no s'executa l'excepció i retorna en aquest sols el títol introduït.

testGetTitol

En aquest test volem comprovar si la funció getTítol, pot executar de manera correcta l'excepció NoExisteixTitol. Per dintre d'aquesta funció utilitza la funcionalitat getTitols comprovada anteriorment i, per tant, comprovarem aquesta excepció. Primer de tot farem servir els valors d'autor "Juan", títol "EL MAN" i path "EL WOMAN.ojmj". La consulta es preguntarà per autor "Juan" i títol "wOMAN". Donant així una excepció.

testModificarAutor

En aquest test vol comprovar si la funció de modificació d'autor d'un Document funciona. En aquest cas utilitzarem els valors d'autor "Juan", títol "EL MAN" i path "EL WOMAN.ojmj". Posteriorment, es canviarà d'autor a "pedro" i, per tant, comprovarem que existeix l'autor "pedro" i deixa d'existir l'autor "Juan".

testModificarTitol

En aquest test es vol comprovar si la funció de modificació del títol Document funciona. En aquest cas utilitzarem els valors d'autor "Juan", títol "EL MAN" i path "si.ojmj". Posteriorment, es canvia de Títol a "EL WOMAN" i comprovarem que el path que dona la consulta get títol amb els paràmetres d'autor i nou títol dona exactament el path d'abans.

testModificarTitol2

És el mateix test que abans, però ha de donar una excepció de NoExisteixAutor perquè fem la consulta de get títol amb un autor diferent del proposat, tot això amb l'objectiu de veure si la modificació de títol no crea cap nou autor.

testEliminarTitol

En aquest test, es vol comprovar si la funció eliminarTitol funciona de manera adequada.
En aquest cas els valors d'autor "Juan", títol "EL MAN" i path "EL WOMAN.ojmj". Posteriorment, eliminem aquest Document i comprovem si l'autor ja no existeix.

testEliminarTitol2

És el mateix test d'abans, però abans de fer l'eliminació fem un altra inserció amb el mateix autor però un document diferent. Això fa que encara que s'elimini el primer Document, encara existeix l'autor.

5.6. Test IndexAutors

testAfegirAutor

En aquest test comprovem si el mètode afegirAutor(autor) funciona de manera adequada. Per això afegim un autor amb nom “autor” i després el busquem amb la funció cercarAutor(prefix) amb prefix = “autor. Es passarà el test si el resultat de cercarAutor(“autor”) retorna una sola String amb valor “autor”.

testAfegirAutorExistent

En aquest test comprovem si el mètode afegirAutor(autor) funciona de manera adequada quan s’intenta afegir un autor que ja existeix. Per això afegim dues vegades un autor amb el mateix nom amb el mètode afegirAutor(autor) i esperem que es llenci l’excepció JaExisteixAutor quan fem la segona crida a afegirAutor(autor).

testEliminarAutor

En aquest test comprovem si el mètode eliminarAutor(autor) funciona de manera adequada. Per això afegim un autor amb nom qualsevol i després provem d’eliminar-ho. Si fem la cerca per prefix i no trobem l’autor és que s’ha eliminat correctament.

testEliminarAutorNoExistent

En aquest test comprovem si el mètode eliminarAutor(autor) funciona de manera adequada quan intentem eliminar un autor que no existeix. Per això cridem al mètode d’eliminar un

autor amb un nom qualsevol i hauríem de rebre l'excepció que estem intentant eliminar un autor inexistent.

testModificarAutor

En aquest mètode comprovem si el mètode modificarAutor(autor1, autor2) funciona de manera adequada. Per això creem un autor amb nom qualsevol i l'intentem canviar el nom a un de diferent mitjançant la funció modificarAutor. Si després no trobem cap autor amb nom autor1 però en canvi sí que en trobem amb autor2 és que s'ha modificat correctament.

testModificarAutorNoExistent

En aquest test comprovem si el mètode modificarAutor(autor1, autor2) funciona de manera adequada quan intentem modificar el nom a un autor inexistent. Per això cridem a la funció de modificarAutor amb un nom qualsevol i si ens salta l'excepció que no existeix l'autor és que hem passat el test satisfactòriament.

testModificarAutorAAutorJaExistent

En aquest test comprovem si el mètode modificarAutor(autor1, autor2) funciona de manera adequada quan intentem modificar el nom d'un autor al nom d'un altre autor ja existent. Per això creem dos autors amb noms qualsevol i intentem utilitzar el mètode de moficarAutor enter els dos noms. Si ens salta l'excepció que ja existeix l'autor significa que hem passat el test satisfactòriament.

testCercarAutor

En aquest test comprovem si el mètode cercarAutor(autors) funciona de manera adequada. Per això creem una sèrie d'autors i utilitzem el mètode de cercarAutor amb diferents noms d'autor per comprovar que en tots casos obtenim el resultat esperat.

testCercarAutorNoExistent

En aquest test comprovem si el mètode cercarAutor(prefix) funciona de manera adequada quan intentem cercar un autor que no existeix. Per això creem una sèrie d'autors i després cridem al mètode de cercarAutor amb un nom d'autor inexistent per rebre l'excepció d'autor no existeix.

5.7 Test ExpressioBooleana

TestCreadora

En aquest test es prova que es crea correctament una ExpressioBooleana donant-li una expressió vàlida.

TestCrearExpressioValida1

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana bàsica amb l'operador '&', com per exemple:"a&b".

TestCrearExpressioValida2

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana bàsica amb l'operador '|', com per exemple:"a|b".

TestCrearExpressioValida3

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana bàsica amb l'operador '!', com per exemple:"!a".

TestCrearExpressioValida4

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana bàsica amb corxetes, com per exemple:"{a b c}".

TestCrearExpressioValida5

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana bàsica amb cometes, com per exemple:""a b c d e"".

TestCrearExpressioValida6

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana bàsica consistent en tan sols una paraula, com per exemple: "paraula".

TestCrearExpressioValida7

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana més complexe contenint barrejades les diferents operacions, com per exemple: "{a b c} | !d & "frase 12345"".

TestCrearExpressioValida8

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana més complexe contenint barrejades les diferents operacions, quan també hi han molts espais estranys, com per exemple: "a & b | !d & "frase 12345" | !{ p1 p2 p3 }".

TestCrearExpressioValida9

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana més complexe contenint barrejades les diferents operacions, essent aquesta la del enunciat de la pràctica: "{p1 p2 p3} & ("hola adéu" | pep) & !joan".

TestCrearExpressioValida10

En aquest test es comprova que es crea correctament una ExpressioBooleana quan es dona una expressió booleana bàsica consistint en una sola paraula entre corxetes, com per exemple: “{paraula}”.

TestCrearExpressioParentesisInvalids1

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de parèntesis no tancats, com per exemple: "((a & b) | c". Salta l’excepció ExpressioNoValida.

TestCrearExpressioParentesisInvalids2

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de parèntesis buits, com per exemple: "(a & b) | c)". Salta l’excepció ExpressioNoValida.

TestCrearExpressioParentesisInvalids3

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de parèntesis no oberts, com per exemple: "(a & b) & ()". Salta l’excepció ExpressioNoValida.

TestCrearExpressioCorxetesInvalids1

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de corxetes no tancats, com per exemple: "{{a b}". Salta l’excepció ExpressioNoValida.

TestCrearExpressioCorxetesInvalids2

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de corxetes no oberts, com per exemple: "{a b}}". Salta l'excepció ExpressioNoValida.

TestCrearExpressioCorxetesInvalids3

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de corxetes buits, com per exemple: "{}". Salta l'excepció ExpressioNoValida.

TestCrearExpressioCometesInvalides1

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de cometes no tancades, com per exemple: " "a b c d e f ". Salta l'excepció ExpressioNoValida.

TestCrearExpressioCometesInvalides2

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de cometes no obertes, com per exemple: " a b c d e f " ". Salta l'excepció ExpressioNoValida.

TestCrearExpressioCometesInvalides3

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a alguna instància de cometes buides, com per exemple: " " " ". Salta l'excepció ExpressioNoValida.

TestCrearExpressioInvalidaNoOperador1

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a no haber operador i només dues paraules, com per exemple: "paraula1 paraula2". Salta l'excepció ExpressioNoValida.

TestCrearExpressioInvalidaNoOperador2

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a no haber operador entre corxetes, com per exemple: "{a b}{c d}". Salta l'excepció ExpressioNoValida.

TestCrearExpressioInvalidaNoOperador3

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a no haber operador entre cometes, com per exemple: "”a b””c d”"". Salta l'excepció ExpressioNoValida.

TestCrearExpressioInvalidaNoOperador4

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a no haber operador en una expressió més complexe amb cometes i corxetes, com per exemple: ""a b"{c d}". Salta l'excepció ExpressioNoValida.

TestCrearExpressioInvalidaNoOperador5

En aquest test es comprova que la creació falla a l'intentar donar una expressió booleana invàlida, en aquest cas a causa de no haver operador en una expressió més complex amb paraules soltes, cometes i corxetes, com per exemple: "{paraulacorchetes}paraula"paraulacometes"". Salta l'excepció ExpressioNoValida.

TestCrearExpressioInvalidaNoOperand1

En aquest test es comprova que la creació falla al intentar donar una expressió booleana invàlida, en aquest cas degut a no haber operand per l'operador '&', com per exemple: "a&". Salta l'excepció ExpressioNoValida.

TestCrearExpressioInvalidaNoOperand2

En aquest test es comprova que la creació falla a l'intentar donar una expressió booleana invàlida, en aquest cas a causa de no haver operand per l'operador '|', com per exemple: "a|". Salta l'excepció ExpressioNoValida.

TestCrearExpressioInvalidaNoOperand3

En aquest test es comprova que la creació falla a l'intentar donar una expressió booleana invàlida, en aquest cas degut a l'haber un doble operand ("&&" per exemplificar, equivalent als altres). Equival a què un dels dos no té operand. Exemple: "a&&b". Salta l'excepció ExpressioNoValida

TestCrearExpressioInvalidaNoOperand4

En aquest test es comprova que la creació falla a l'intentar donar una expressió booleana invàlida, en aquest cas a causa de no haver operand per l'operador '!', com per exemple: "a & ! ". Salta l'excepció ExpressioNoValida.

TestCercarExpressio1

En aquest test es comprova que es realitza correctament la cerca d'un document definit per una expressió bàsica.

TestCercarExpressio2

En aquest test es comprova que es realitzen correctament les cerques de documents diferenciant els signes de puntuació.

TestCercarExpressio3

En aquest test es comprova que s'obtenen tots els documents en cercar una tautologia.

TestCercarExpressio4

En aquest test es comprova que no s'obté cap document en cercar una contradicció.

TestCercarExpressio5

En aquest test es comprova que les cerques funcionen correctament a l'utilitzar parèntesis per canviar la precedència dels operadors.

TestConsultarExpressio

En aquest test es comprova que es consulta correctament l'expressió booleana definida per l'instància actual de la classe.

5.8 Test NodeExpressio

TestCreadoraBuida

En aquest test es prova que es crea correctament un node sense paràmetres.

TestCreadoraAmbParametre

En aquest test es prova que es crea correctament un node indicant-li el valor d'aquest.

TestSetterGetterFillDret

Test conjunt per setter i getter, com que per testejar un hem de cridar l'altre i, per tant, tests separats són idèntics. Test per comprovar correctament el set i get del fill dret del node.

TestSetterGetterFillEsquerre

Test conjunt per setter i getter, com que per testejar un hem de cridar l'altre i, per tant, tests separats són idèntics. Test per comprovar correctament el set i get del fill esquerre del node.

TestSetterGetterValor

Test conjunt per setter i getter, com que per testejar un hem de cridar l'altre i, per tant, tests separats són idèntics. Test per comprovar correctament el set i get del valor del node.

TestSetterGetterOperacio

Test conjunt per setter i getter, com que per testejar un hem de cridar l'altre i, per tant, tests separats són idèntics. Test per comprovar correctament el set i get de l'operació del node.

5.9 Test Vector_TFIDF i Vector_BM25

Aquests tests els comentarem conjuntament, ja que fan exactament el mateix per una estratègia de pesos com per l'altra, l'únic que canvia és la forma de calcular les coordenades, però l'objectiu dels tests és exactament el mateix. Tampoc hem fet test de la classe Vector_EV, perquè com és una classe abstracta comprovarem els mètodes dins dels seus fills.

TestRecalcularTotesLesCoordenades

Aquesta col·lecció de testos, simplement comprova que l'estratègia de pesos s'hagi implementat bé i calculi la coordenada correcta per cada paraula que no sigui un stopWord.

TestRecalcularTotesLesCoordenades1

En aquest test comprovem que la coordenada a l'afegir una única paraula dins d'un document es calculi correctament.

TestRecalcularTotesLesCoordenades2

En aquest test comprovem que les coordenades a l'afegir dues paraules dins d'un document es calculi correctament.

TestRecalcularTotesLesCoordenades3

En aquest test comprovem que les coordenades a l'afegir dues paraules dins de dos documents es calculin correctament.

TestRecalcularTotesLesCoordenades4

En aquest test comprovem que les coordenades a l'afegir dues paraules dins d'un document i afegint una paraula compartida a l'altre document es calculin correctament.

TestRecalcularTotesLesCoordenades5

En aquest test afegim dos documents un amb dues paraules i l'altre amb un, i comprovem que si "elimino" un vector (crido a la destructora) les coordenades de l'altre document es tornen a calcular correctament.

TestNorma

En aquest test comprovarem si donades unes paraules que afegim al vector, la norma d'aquest vector s'està calculant correctament.

TestNorma1

En aquest test comprovem que la norma d'un vector que no conté elements ha de ser 0.

TestNorma2

En aquest cas comprovem que la norma d'un únic document que té una paraula ha de ser 1 (sempre que no sigui una StopWord).

TestNorma3

Comprovem que la norma d'un vector que conté dues paraules diferents s'està calculant correctament.

TestNorma4

En aquest cas comprovem que la norma de dos documents que contenen algunes paraules compartides i altres que no, s'està calculant correctament.

TestNorma5

En aquest cas comprovem que la norma de dos documents que contenen algunes paraules compartides i altres que no, s'està calculant correctament amb més paraules que el test anterior.

TestIntersecciónVector

En aquest test comprovarem que la intersecció entre dos vectors es fa correctament. La intersecció bàsicament ens permet obtenir una llista dels elements que estan als dos vectors.

TestIntersecciónVector1

Comprovem que la intersecció de dos vectors buits es buida.

TestIntersecciónVector2

Comprovem que la intersecció d'un vector buit i un que no ho és també es buida, amb el vector implícit.

TestIntersecciónVector3

Comprovem el mateix que l'anterior però amb el vector passat per paràmetre en lloc del vector implícit.

TestIntersecciónVector4

Comprovem la intersecció entre dos vectors que comparteixen la paraula "hola".

TestIntersecciónVector5

Comprovem la intersecció entre dos vectors que comparteixen més d'un element "hola" i "adeu" i no tenen cap element no compartit.

TestIntersecciónVector6

Comprovem la intersecció entre dos vectors que comparteixen més d'un element "hola" i "adeu" i tenen algun element no compartit.

TestObtenirDimensionalitat

En aquest test comprovarem que la dimensionalitat que ens retorna un Vector és correcta.

TestObtenirDimensionalitat1

En el primer cas, comprovem que la dimensionalitat d'un vector sense cap coordenada és 0.

TestObtenirDimensionalitat2

En el segon cas, comprovem que la dimensionalitat d'un vector al qual li hem afegit una paraula (una coordenada) és 1.

TestObtenirDimensionalitat3

En el tercer cas, comprovem que la dimensionalitat d'un vector al qual li hem afegit dues vegades la mateixa paraula és 1.

TestObtenirDimensionalitat4

A l'últim cas comprovem que si les paraules són diferents sí que està incrementant la dimensionalitat del vector.

TestConteParaulaIAfegirParaules

Tot i que hem de fer tests unitaris, en aquest cas comprovarem dos mètodes dins del mateix tests, ja que els necessitem els 2 per fer-lo, aquests mètodes són ConteParaula i AfegirParaules.

TestConteParaulaIAfegirParaules1

En el primer cas comprovem que si a un vector buit li preguntem per una paraula que no conté, ConteParaula retorna false.

TestConteParaulaIAfegirParaules2

En aquest, afegim la paraula "hola" al vector, i preguntem si la conté, esperant un true.

TestConteParaulaIAfegirParaules3

En aquest, afegim la paraula "hola" dos cops al vector, i preguntem si la conté, esperant un true. D'aquesta manera veiem si s'estan fent bé les insercions d'una mateixa paraula més d'un cop.

TestParaules

Farem tests per aquest mètode que ens permet obtenir totes les paraules que conté un document sense repeticions.

TestParaules1

Primer que tot, comprovem que un vector buit retorna una llista buida.

TestParaules2

En aquest test, comprovem que si afegim una paraula, se'ns retorna la paraula que hem afegit.

TestParaules3

En aquest test, comprovem que si afegim una paraula més d'un cop, se'ns retorna la paraula sense repetits.

TestParaules4

En aquest test, afegim dues paraules diferents del vector i comprovem que se'ns retornen les dues.

TestDestructora

En aquest test creem dos documents amb paraules i comprovem que un com eliminat un vector, l'altre recalculi les coordenades correctament.