Entrega 1 PROP
Q1 2022-23

Integrants:
Lucas Ares, Miquel Muñoz, Carles Pedrals i Pau Cuscó.

1. Casos d’ús

Diagrama casos d’ús:

Especificació de casos d’ús:

Nom: Alta document

Actor: Usuari

Comportament: l’usuari informa de l’autor, títol (identificadors) i contingut d’un document. El sistema guarda les dades introduïdes a memòria en un objecte “document”.

Errors possibles i cursos alternatius:

→ El sistema informa d’error si ja existeix un document a memòria amb els mateixos identificadors títol i autor.

Nom: Alta Autor
Actor: Usuari
Comportament: el sistema, en la classe documents afegeix un nou identificador (autor) del document en cas que no hi hagi cap document amb aquell nou autor.
Errors possibles i cursos alternatius:

Nom: Baixa Autor
Actor: Usuari
Comportament: el sistema, en la classe documents suprimeix l' identificador (autor) en cas que un autor que havia tingut documents al seu nou ja no en tingui.
Errors possibles i cursos alternatius:

Nom: Baixa document
Actor: Usuari
Comportament: l'usuari informa de l'autor, títol (identificadors). El sistema esborra les dades introduïdes a memòria del document identificat.
Errors possibles i cursos alternatius:
→ El sistema informa d'error si no existeix un document a memòria amb els identificadors títol i autor introduïts.

Nom: Modificar document
Actor: Usuari
Comportament: l'usuari informa de l'autor, títol (identificadors) del document que vol modificar. Aleshores informa del nou contingut. El sistema modifica el contingut.
Errors possibles i cursos alternatius:
→ El sistema informa d'error si no existeix un document a memòria amb els identificadors títol i autor introduïts.

Nom: Llistar per prefix
Actor: Usuari
Comportament: l'usuari informa d'un prefix d'un autor. El sistema mostra tots els autors que comencen per aquell prefix.
Errors possibles i cursos alternatius:
→En cas que el prefix sigui buit el sistema mostra tots els autors que hi ha a memòria.

Nom: Mostrar contingut
Actor: Usuari
Comportament: l'usuari informa de l'autor, títol (identificadors) del document. El sistema mostra el contingut del document.
Errors possibles i cursos alternatius:
→ El sistema informa d'error si no existeix un document a memòria amb els identificadors títol i autor introduïts.

Nom: Llistar documents autor
Actor: Usuari
Comportament: l'usuari informa de l'autor. El sistema mostra els títols que hi ha a memòria associats amb aquest autor.
Errors possibles i cursos alternatius:

→ El sistema informa d'error si no existeix a memòria el autor introduït.

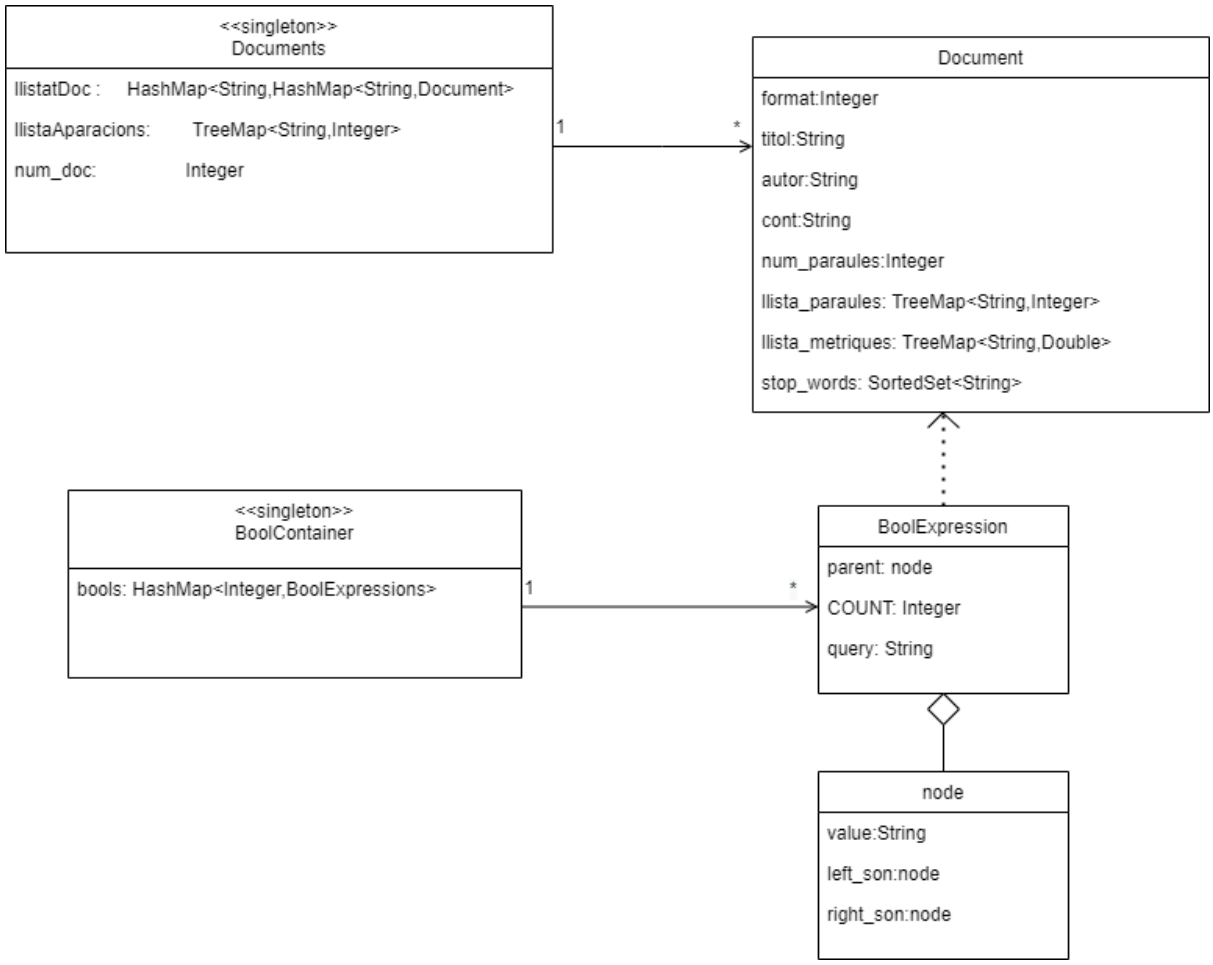
Nom: Actualitzar aparicions
Actor: Sistema
Comportament: El sistema informa de quin document ha canviat. El sistema actualitza en quants documents apareix cada paraula.
Errors possibles i cursos alternatius:
→ El sistema informa de l'error si no existeix el document.

Nom: Calcular mètrica
Actor: Sistema
Comportament: el sistema indica en quin document s'han de realitzar els càlculs. El sistema realitza els càlculs i els guarda.
Errors possibles i cursos alternatius:
→ El sistema informa de l'error si no existeix el document.

Nom: Llistar documents per semblança
Actor: Usuari
Comportament: L'usuari informa del títol, l'autor d'un document i un nombre natural 'K'. El sistema retorna els K documents més semblants al document amb títol i autor indicats.
Errors possibles i cursos alternatius:
→ Si no existeix cap autor amb el nom indicat, el sistema informa d'error
→ Si no existeix cap títol de l'autor amb el nom indicat, el sistema informa d'error.

2. Model Conceptual de dades

Diagrama UML:



Especificació de les classes:

Documents: La classe documents tenim tres atributs que ens permeten identificar cada document per títol i autor. Tenim llistatDoc per tenir de cada autor els seus títols i documents de manera que no hem d'accedir al Document pròpiament. També ens serveix llistaAparicions per saber en quants documents apareix determinada paraula. "num_doc" ens indica fàcilment quants documents tenim a memòria amb cost constant ja que sinó hauríem de recórrer cada mapa de mapes per anar sumant quants documents hi ha en total amb cost *n* (sigui *n* el nombre d'autors).

Document: Document ens serveix per manipular el contingut del propi document i també s'utilitza per dur a terme la cerca de llistar semblants. Títol i autor també son atributs ja que ens serveixen per dur a terme la funcionalitat esmentada. L'explicació de les estructures de dades utilitzades per fer la cerca està més desenvolupada al punt 3.

BoolContainer, node i BoolExpression: BoolContainer ens permet manipular cada una de les expressions booleanes guardades al sistema fent operacions com altes i baixes. BoolExpression utilitza una estructura de nodes (classe d'agregació "node") que permet construir un arbre conservant les precedències. Això permet dur a terme la cerca booleana.

3. Descripció de les estructures de dades i dels algoritmes

Estructures de dades de Document.java

llista_paraules (TreeMap<String,Integer>): un *map* on s'emmagatzemen totes les paraules del contigut del document i del títol separades i si alguna es repeteix només surt una vegada i el valor de l'entrada equival a les vegades que apareix.

llista_metriques (TreeMap<String,Double>): un *map* amb totes les paraules que no coincideixen amb cap *stop word* i la mètrica tf-idf de cadascuna.

stop_words (SortedSet<String>): un set amb totes les *stop words* ordenades alfabèticament.

Estructures de dades de Documents.java

llistatDoc (HashMap<String,HashMap<String,Document>>): un *map* on la clau del mapa extern és l'autor del document i el valor és un mapa "intern" on la clau és el títol del document i valor és l'objecte Document.

llistaAparicions (TreeMap<String,Integer>): un *map* ordenat on el primer valor és una paraula que apareix en algun dels documents i la segona en quants documents surt.

Algoritme de llistar documents per semblança

Bàsicament hem fet servir la mètrica tfidf i el model d'espais vectorials per calcular la semblança entre documents.

Primerament hem separat totes les paraules del contingut i del títol i les hem guardat a llista_paraules. Seguidament hem actualitzat llistaAparicions amb les noves paraules que tenim dels documents entrats i llavors ja tenim totes les dades per poder calcular el valor de tf-idf amb la seguent fórmula:

$$w_{t,d} = \mathrm{tf}_{t,d} \cdot \log \frac{|D|}{|\{d' \in D \,|\, t \in d'\}|}$$

On amb un bucle que va recorrent les paraules d'un document on calculem tf que és quantes vegades apareix una paraula en el document (llista_paraules) entre el número total de paraules del document i el idf que és el logaritme de la divisió entre el número de documents que tenim en el sistema entre el número de documents on apareix la paraula (llistaAparicions). Aquests valors els guardem a llista_metriques.

Quan ja tenim totes les mètriques procedim a calcular les distàncies entre documents. Com que el que se'ns demana és que calculem la semblança entre un document i tota la resta el que farem serà anar comparant aquell document un a un entre els altres.

La fórmula utilitzada per calcular la semblança (i per tant la distància segons el model d'espais vectorials) és la següent, on cada una de les paraules és una dimensió.

$$\cos(d_j, q) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{\|\mathbf{d_j}\| \, \|\mathbf{q}\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}}$$

On la part de dalt és la multiplicació de la mètrica tf-idf en el document de referència per la mètrica tf-idf del document amb el que el comparem i la part de baix la norma del vector del document de referència (l'arrel de la multiplicació de totes les mètriques elevades al quadrat) per la norma del vector amb el que el comparem. Anem guardant els valors en una

llista que ordenem per la mètrica i així aconseguim els documents ordenats per més a menys semblants.

Estructures de dades de BoolExpresion.java

La classe BoolExpression representa la expressió booleana mitjançant un Abstract Syntax Tree (AST) amb una estructura per nodes recursiva. Aquesta definició permet respectar les precedències. S'utilitza una funció de treatment que comprova que el input sigui parseable i un parser que converteix la String a un AST. Aquest és després avaluat en un recorregut recursiu.

4. Relació de classes implementades per cada membre

La feina ha estat dividida en funcionalitats. La gestió d'expressions booleanes i la cerca amb booleans està implementada a la classe BoolExpression, node i BoolContainer. Aquestes classes les ha implementat Carles Pedrals.

La funcionalitat relacionada amb la cerca per semblança l'ha dut a terme Pau Cuscó. Aquesta funcionalitat intervé a les classes Document i Document afegint les funcions de ordenar per semblança, calcular la distància entre documents, actualitzar aparicions, inicialitzar i comprovar si es una stop word, calcular mètrica i contar paraules. S'han posat totes a document i documents en comptes de crear una nova classe ja que, com que totes les funcions bàsicament treballaven o amb un document o amb tots els documents no s'ha cregut necessari.

Les funcionalitats llistar títols d'un autor, alta document, baixa document, modificar document, llista d'autors per prefix i contingut de document han estat implementades per Miquel Muñoz i Lucas Ares. Aquestes funcionalitats intervenen les classes Documents i Document. D'aquestes últimes funcionalitats només ens és necessari accedir a Document per utilitzar la creadora i per accedir al atribut *contingut* del Document. Aquest es pot consultar o modificar depenent de la situació. Totes les altres funcionalitats es fan directament des de Documents ja que les claus externes les tenim identificadores al mapa de mapes. Aleshores podem fer consultes de títols i autors sense haver de consultar el propi Document.

5. Llibreries externes utilitzades

Hem utilitzat JUnit i Hamcrest core per testejar.