GESTOR DE DOCUMENTS

DOCUMENTACIÓ

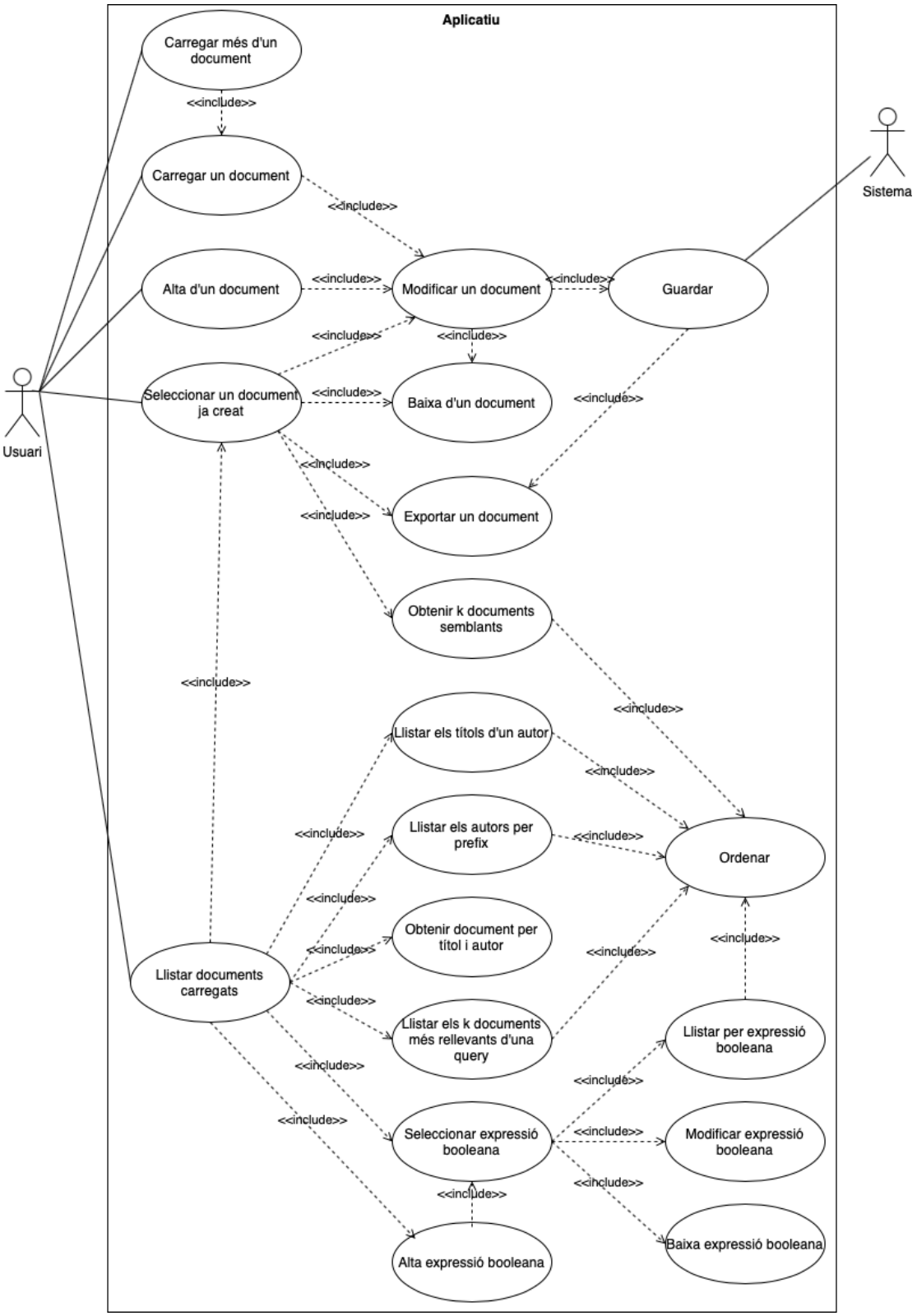
Prop grup 33.1

ariadna.cortes.danes
pau.duran.manzano
marc.lopez.domenech
marc.valls.camps

TAULA DE CONTINGUTS

1. Casos d'ús

1.1. Diagrama de casos d'ús



1.2. Especificació dels casos d'ús

1.2.1. Carregar un document

Actor	Usuari
Descripció	L'usuari carrega un document del seu ordinador al sistema
Precondicions	El document conté dades correctes de títol i autor
Entrada	"path" del document a carregar
Sortida/Postcond.	El sistema crea un nou document al sistema amb autor, títol i contingut corresponent al path rebut i actualitza les seves dades en conseqüència.
Errors/Cur.Alt.	Si ja existia el fitxer amb títol i autor, no es crea un nou Fitxer al sistema sinó que es modifica la informació del ja existent. Si el format no és vàlid (txt o xml) es mostra un avís a l'usuari i no es modifica l'estat del sistema.

1.2.2. Carregar més d'un document

Actor	Usuari
Descripció	L'usuari carrega més d'un document de cop des del seu ordinador al sistema. El sistema comença el cas d'ús "carregar document" amb cada un dels path de documents indicats.
Precondicions	Els documents contenen dades correctes de títol i autor
Entrada	Llistat de paths del document a carregar
Sortida/Postcond.	El sistema crea els nous documents al sistema cada un amb autor, títol i contingut corresponent al seu path concret i actualitza les seves dades en conseqüència.

Errors/Cur.Alt.	Si ja existia algun dels documents, no es crea un nou document al sistema sinó que es modifica la informació del ja existent. Si algun dels formats no és vàlid (txt o xml) es mostra un avís a l'usuari i el document que hagi generat l'error no s'afegeix al sistema.

1.2.3. Alta Document Buit

Actor	Usuari
Descripció	L'usuari introdueix títol i autor d'un nou document que serà creat sense contingut al sistema.
Precondicions	
Entrada	Títol i autor del nou document
Sortida/Postcond.	El sistema crea un nou document al sistema amb el títol i l'autor especificats i el contingut buit, i actualitza les estructures de dades del sistema en conseqüència.
Errors/Cur.Alt.	Si ja existia el document amb el títol i l'autor donats, es permet a l'usuari: - Editar el document amb títol i autor donats (cas d'ús Modificar un document). - Sobreescriure el document amb títol i autor donats (el sistema esborra el document existent i es dona d'alta un de nou amb el títol i autor donats i sense contingut). - Tornar a introduir el títol i l'autor (tornar a l'inici d'aquest cas d'ús).

1.2.4. Seleccionar un document

Actor	Usuari
Descripció	L'usuari tria un document sobre el qual fer les possibles operacions. Pot seleccionar un document d'un llistat mostrat pel sistema o bé cercant-lo per títol i autor. Aquest cas d'ús donarà peu a començar a algun cas d'ús d'operacions sobre un document (baixa, modificació, llistar k semblants, exportar)
Precondicions	
Entrada	Títol, autor del document sobre el qual volem realitzar alguna operació
Sortida/Postcond.	El sistema informa l'usuari de quin document té seleccionat
Errors/Cur.Alt.	Si el document cercat per títol i autor no existeix en el sistema, es mostra un avís a l'usuari.

1.2.5. Exportar un document

Actor	Usuari
Descripció	L'usuari vol descarregar un document (que ha seleccionat) en un format concret.
Precondicions	Hi ha un document seleccionat que és vàlid. El format serà txt o xml.
Entrada	Títol i autor que identifiquen el document seleccionat Format a descarregar (xml o txt)
Sortida/Postcond.	L'usuari té el document descarregat a l'ordinador en el format triat. L'estat del sistema no ha canviat.
Errors/Cur.Alt.	

1.2.6. Llistar k més semblants

Actor	Usuari
Descripció	Llistar els k documents més semblants/rellevants (pel que fa a contingut) del document que té seleccionat l'usuari
Precondicions	Hi ha un document seleccionat que és vàlid. k és un nombre natural.
Entrada	Títol i autor que identifiquen el document seleccionat Un nombre natural k
Sortida/Postcond.	Llista de k documents més rellevants o una llista buida
Errors/Cur.Alt.	Si k > nombre de documents trobats, es retornen tots els documents que s'hagin trobat al sistema.

1.2.7. Modificar un document

Actor	Usuari
Descripció	L'aplicació permet l'edició de documents, permetent la inserció i l'esborrat de caràcters al contingut del document seleccionat. Si l'usuari vol guardar el nou contingut, començarà el cas d'ús "Guardar". Si l'usuari decideix no guardar les modificacions, es restablirà l'estat original del document.
Precondicions	Existeix el document seleccionat.
Entrada	Contingut original del document seleccionat en format text
Sortida/Postcond.	Es mostra el contingut modificat a l'usuari, sense produir cap canvi al sistema
Errors/Cur.Alt.	

1.2.8. Baixa d'un document

Actor	Usuari
Descripció	L'usuari vol eliminar de l'aplicatiu un document que ha seleccionat, i el sistema el suprimeix de les seves dades.
Precondicions	Hi ha un document seleccionat que és vàlid.
Entrada	Títol i autor que identifiquen el document seleccionat
Sortida/Postcond.	El document identificat per títol i autor s'esborra del sistema
Errors/Cur.Alt.	

1.2.9. Guardar el contingut d'un document

Actor	Usuari Ocasionalment, el sistema pot autoguardar de forma periòdica
Descripció	Es salven al sistema els canvis fets al document seleccionat durant l'edició del seu contingut.
Precondicions	Existeix un document seleccionat, i s'estava editant.
Entrada	Nou contingut del document que es vol salvar.
Sortida/Postcond.	Es reemplaça el contingut antic del document pel nou, canviant les dades associades al contingut que guardi el sistema
Errors/Cur.Alt.	

1.2.10. Llista documents

Actor	Usuari
Descripció	L'usuari especifica uns paràmetres de cerca i ordenació per a filtrar els documents carregats al sistema. Si l'usuari introdueix autor, comença el cas d'ús "Llistar els títols d'un autor". Si l'usuari introdueix títol i autor, comença el cas d'ús "Obtenir document per títol i autor". Si l'usuari introdueix una query, comença el cas d'ús "Llistar per Query". Si l'usuari introdueix un prefix d'autor, comença el cas d'ús "Llistar per autor per prefix". Si l'usuari selecciona una expressió, comença el cas d'ús "Llistar per Expressió booleana". Després de filtrar i obtenir el resultat, comença el cas d'ús "Ordenar resultats" en cas que l'usuari ho demani.
Precondicions	Si es tracta d'una cerca per expressió booleana, l'expressió és vàlida.
Entrada	Paràmetres de cerca (query, k , títol, autor/prefix d'autor, expressió booleana) i paràmetres d'ordenació (alfabètic per títol/autor, rellevància)
Sortida/Postcond.	El sistema retorna els documents resultants d'executar el cas d'ús corresponent (ordenats en cas que es demani) o una llista buida.
Errors/Cur.Alt.	Si k no és natural es mostra un missatge d'error a l'usuari. Si k > nombre de documents trobats per la query, es retornen tots els trobats.

1.2.11. Obtenir document per títol i autor

Actor	Usuari
Descripció	Donats un títol i un autor, es retorna el document corresponent
Precondicions	
Entrada	Títol i autor
Sortida/Postcond.	Es mostra a l'usuari el document amb el títol i autor especificats, sense produir canvis en el sistema.
Errors/Cur.Alt.	Si no existeix un document (títol, autor) identificat per les entrades donades, es mostra un missatge d'avís a l'usuari.

1.2.12. Llistar els títols d'un autor

Actor	Usuari
Descripció	L'usuari proporciona un nom d'autor al sistema i aquest li retorna el conjunt de títols de documents que estan associats a aquell autor.
Precondicions	
Entrada	Autor
Sortida/Postcond.	Llistat de tots els títols associats a l'autor donat o llista buida.
Errors/Cur.Alt.	Si l'autor no té associat cap títol (és a dir, no està introduït al sistema), s'indicarà aquesta situació a l'usuari.

1.2.13. Llistar autors per prefix

Actor	Usuari
Descripció	L'usuari proporciona un prefix, que pot ser buit, al sistema i aquest li retorna el conjunt de noms d'autors que comencen per aquell prefix.
Precondicions	
Entrada	Prefix
Sortida/Postcond.	Llistat de tots els noms d'autor que comencen pel prefix donat o llista buida
Errors/Cur.Alt.	Si no hi ha cap nom d'autor que comenci pel prefix donat, s'indicarà aquesta situació a l'usuari.

1.2.14 Ordenar resultat

Actor	Usuari
Descripció	L'usuari vol ordenar el llistat que ha obtingut de documents d'acord amb diversos paràmetres
Precondicions	Es disposa d'un llistat de documents a ordenar amb documents vàlids.
Entrada	Llistat de documents i un criteri d'ordenació
Sortida/Postcond.	Es retorna una nova llista de documents ordenats seguint el criteri d'ordenació donat. El sistema queda inalterat
Errors/Cur.Alt.	

1.12.15 Llistar els k documents més rellevants d'una query

Actor	Usuari
Descripció	L'usuari proporciona un seguit de paraules al sistema (denotades com a "query") i un enter k, i el sistema retorna els k documents que tenen el contingut més rellevant pel que fa a la query introduïda.
Precondicions	k es natural
Entrada	Query (p paraules separades per espais) i un natural k
Sortida/Postcond.	Llistat dels k documents amb un contingut més rellevant respecte la query introduïda o llista buida
Errors/Cur.Alt.	Si k > nombre de documents trobats per la query, es retornen tots els trobats.

1.12.16 Llistar per expressió booleana

Actor	Usuari
Descripció	L'usuari selecciona una expressió booleana del sistema i el sistema retorna tots els documents que compleixin l'expressió
Precondicions	L'expressió és vàlida
Entrada	Expressió booleana a avaluar
Sortida/Postcond.	Llista dels documents que compleixin l'expressió seleccionada o una llista buida
Errors/Cur.Alt.	

1.12.17 Alta expressió booleana

Actor	Usuari
Descripció	L'usuari introdueix una expressió booleana que es guarda al sistema
Precondicions	
Entrada	Expressió booleana en forma de cadena de text
Sortida/Postcond.	L'expressió queda registrada al sistema, modificant-ne l'estat en conseqüència.
Errors/Cur.Alt.	Si l'expressió ja existeix al sistema, no s'afegeix Si l'expressió és invàlida*, es mostra un missatge d'error a l'usuari i el sistema es manté inalterat.

* Les regles de validació d'expressions s'exposen en l'apartat 3. Estructura de dades i algorismes

1.12.18 Seleccionar expressió booleana

Actor	Usuari
Descripció	L'usuari selecciona una expressió existent al sistema. L'usuari pot decidir començar els casos d'ús de "modificar expressió booleana", "donar de baixa expressió booleana" o "llistar per expressió booleana"
Precondicions	
Entrada	Cadena de text corresponent a l'expressió a seleccionar
Sortida/Postcond.	El sistema informa l'usuari de quina expressió té seleccionada. No hi ha canvis en l'estat del sistema.

Errors/Cur.Alt.	Si l'expressió no existeix en el sistema, es mostra un avís a l'usuari.

1.12.19 Modificar expressió booleana

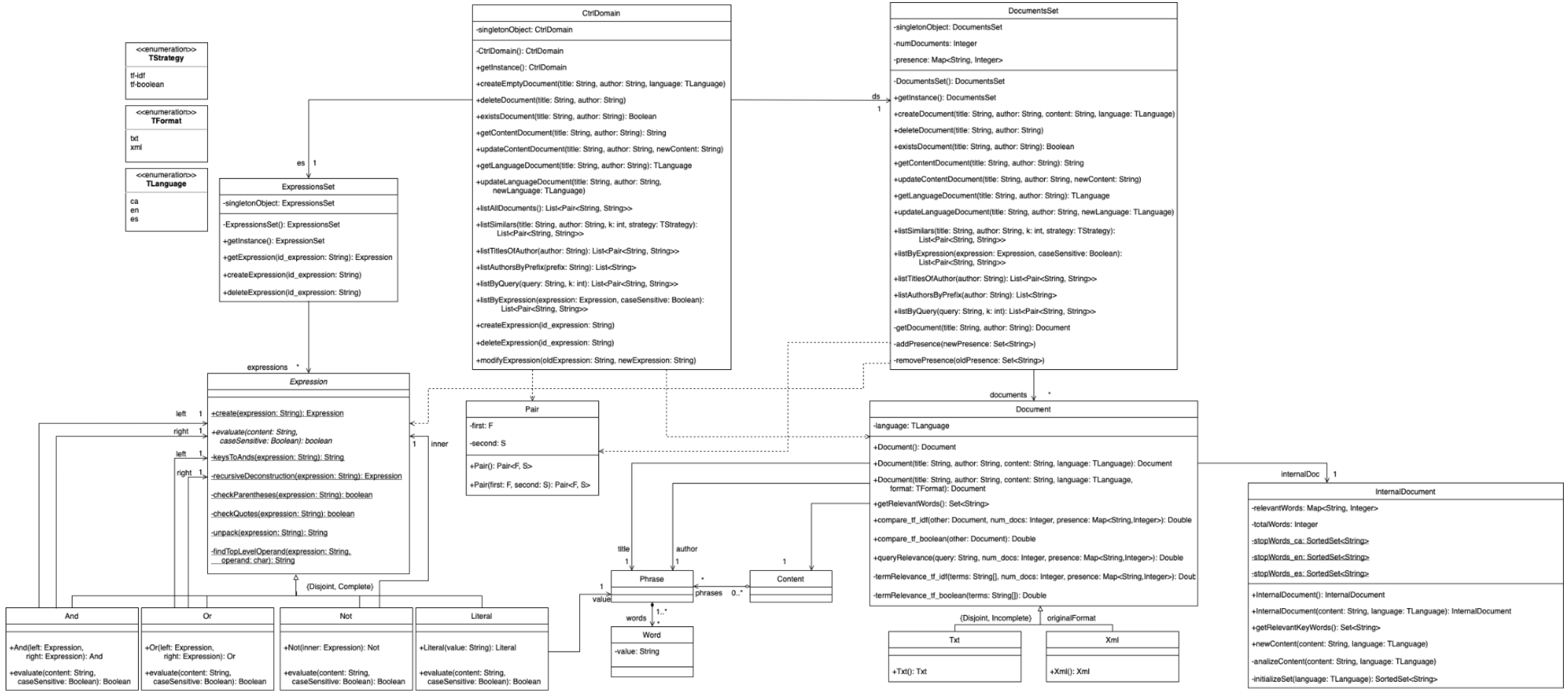
Actor	Usuari
Descripció	L'usuari modifica una expressió booleana existent que prèviament ha seleccionat
Precondicions	L'expressió booleana seleccionada és vàlida
Entrada	Cadena de text corresponent a la nova expressió
Sortida/Postcond.	L'expressió nova queda registrada al sistema i s'esborra l'antiga
Errors/Cur.Alt.	Si l'expressió és invàlida, es mostra un missatge d'error a l'usuari i no es modifica l'expressió. El sistema queda inalterat. Si l'expressió modificada coincideix amb una que ja existeix al sistema llavors no es modifica res i es mostra un avís a l'usuari.

1.2.20 Baixa expressió booleana

Actor	Usuari
Descripció	L'usuari esborra del sistema una expressió booleana que prèviament ha seleccionat
Precondicions	L'expressió booleana seleccionada és vàlida
Entrada	Expressió seleccionada
Sortida/Postcond.	L'expressió s'esborra de les dades del sistema
Errors/Cur.Alt.	

2. Model conceptual

2.1. Diagrama UML



Restriccions textuals del diagrama anterior:

1. Claus externes: (And, left+right), (Or, left+right), (Not, inner), (Literal, value), (Document, title+author), (InternalDocument, relevantWords), (Pair, first+second), (Content, phrases), (Phrase, words), (Word, value).

2. Les classes CtrlDomain, ExpressionsSet i DocumentsSet són singleton, és a dir, com a molt poden tenir una instància en el sistema.

3. L'atribut numDocuments, de DocumentsSet, ha de ser un valor enter major o igual a 0.

4. L'atribut totalWords, de InternalDocument, ha de ser un valor enter major o igual a 0.

5. Tots els "integer" de presence, atribut de DocumentsSet, han de ser valors enters majors estricte a 0.

2.2. Especificació del model conceptual

El nostre diagrama de classes de la capa de domini està compost d'un controlador (CtrlDomain) que s'encarrega de començar a processar i traslladar totes les consultes que es reben des de la capa de presentació a la resta de classes del domini. A més, també s'encarrega de la comunicació amb la capa de persistència.

Aquest controlador trasllada totes les crides, bàsicament, a dues altres classes, que són el conjunt d'expressions (ExpressionsSet) i el de documents (DocumentsSet). Aquestes dues classes s'encarreguen de la gestió pròpia de les expressions i documents, respectivament, que requereix el nostre sistema.

Pel que fa a les expressions, usarem un seguit de classes que parteixen de l'abstracte Expression, que són And, Or, Not i Literal, aquesta última formada a partir de Phrase, que alhora empra Word.

Sobre els documents, definim la classe Document, que també serveix de base per les subclasses Txt i Xml, i que alhora requereix la classe InternalDocument i Content.
Paral·lelament a aquesta estructura, hem definit la classe Pair, que l'usen diferents classes del domini.

A continuació, detallem totes aquestes classes que hem esmentat. Tot i això, aquesta explicació de les classes no es troba en aquest document completa: l'especificació més detallada de cada funció de les classes (detalls, precondicions, paràmetres d'entrada i de retorn, postcondicions i excepcions possibles) es troba documentada en la documentació generada a partir de Doxygen.

2.2.1. CtrlDomain

El controlador de domini és una classe singleton, és a dir, només hi haurà una instància en l'aplicatiu. Per tal de fer efectiva aquesta restricció, disposa de l'atribut singletonObject, de tipus CtrlDomain, que serà inicialitzat a partir de la constructora, que és privada, ja que perquè la resta de classe aconsegueixin la instància usaran la funció getInstance(), que sí que és pública.

El controlador, com s'ha esmentat, requereix les classes ExpressionsSet i DocumentsSet. Tot i que podrien haver estat dependències, aquest acoblament l'hem materialitzat amb dos atributs privats, es i ds, respectivament, perquè el controlador està constantment cridant funcions d'aquestes dues classes, d'aquesta manera ens estalviarem haver d'anar demanant la instància a aquestes classes, que també són singleton.

Pel que fa a les funcions, a més de la constructora i el getInstance que ja s'han mencionat, disposem d'un seguit d'operacions públiques que estan disponibles per a la capa de presentació. Fins ara només hem dissenyat la capa de domini i un subconjunt de les operacions que en la versió final estaran disponibles, ja que manquen les que requereixen la capa de persistència, com les de càrrega i exportació de documents, entre d'altres.

En tot cas, les funcions definides permeten desenvolupar diferents casos d'ús dels definits i algunes funcionalitats addicionals:

- createEmptyDocument() s'usarà en el cas d'ús d'"Alta d'un document".
- deleteDocument() s'usarà en el cas d'ús "Baixa d'un document".
- existsDocument() no es relaciona amb cap cas d'ús en concret, però permet saber si existeix o no un document.
- getContentDocument() s'usarà en el cas d'ús "Modificar un document" i "Obtenir document per títol i autor".
- updateContentDocument() s'usarà en el cas d'ús "Guardar".
- getLanguageDocument() s'usarà en el cas d'ús "Modificar un document" i "Obtenir document per títol i autor". Cal esmentar que un dels paràmetres l'hem representat amb una enumeració, TLanguage, que admet tres valors. Tot i això, la implementació d'aquest paràmetre serà de tipus String i es comprovarà que el valor sigui un dels possibles de l'enumeració.
- updateLanguageDocument() s'usarà en el cas d'ús "Guardar".
- listAllDocuments() no es relaciona amb cap cas d'ús concret, però permet aconseguir un llistat de tots els identificadors de documents que hi ha donats d'alta al sistema.
- listSimilars() s'usarà en el cas d'ús "Obtenir k documents més semblants". D'aquesta funció, destacar que un dels paràmetres, que és "strategy" en el diagrama de classes l'hem representat amb una enumeració TStragey que admet dos valors. Tot i això, en la implementació d'aquest paràmetre serà de tipus String i es comprova que el valor sigui un dels possibles de l'enumeració.
- listTitlesOfAuthor() s'usarà en el cas d'ús "Llistar els títols d'un autor".

- listAuthorsByPrefix() s'usarà en el cas d'ús "Llistar els autors per prefix".
- listByQuery() s'usarà en el cas d'ús "Llistar els k documents més rellevants d'una query".
- listByExpression() s'usarà en el cas d'ús "Llistar per expressió booleana". D'aquesta funció surt la dependència que hem representat en el diagrama del controlador cap a Expression. Això és perquè aquesta funció requereix aconseguir l'expressió que es vol avaluar i, posteriorment, analitzar-la en el document corresponent. Ho hem fet d'aquesta manera per evitar l'acoblament entre DocumentsSet cap a ExpressionsSet.
- createExpression() s'usarà en el cas d'ús "Alta expressió booleana".
- deleteExpression() s'usarà en el cas d'ús "Baixa expressió booleana".
- modifyExpression() s'usarà en el cas d'ús "Modificar expressió booleana".

Sobre el controlador, també esmentar que té una altra dependència marcada en el diagrama cap a la classe Pair. Això és perquè en les funcions de llistar (exceptuant la de listAuthorsByPrefix) la manera de retornar l'identificador del document (títol i autor) és un parell d'strings amb aquesta informació, sent first el títol i second l'autor.

2.2.2. ExpressionsSet

La classe ExpressionsSet pretén representar el conjunt d'expressions booleanes que hi haurà donades d'alta a l'aplicatiu, a més de ser la classe encarregada de la gestió d'aquestes expressions. Es tracta d'una classe singleton que, doncs, té com atribut el singletonObject, la creadora per defecte privada i la funció pública getInstance(). A més, disposa de l'atribut expressions, que representa l'associació cap a Expression del diagrama, que emmagatzemarà totes les expressions donades d'alta al sistema i que aquesta classe s'encarrega de guardar.

Pel que fa a la resta de funcions, totes són públiques. A més de la getter i la setter de l'atribut expressions, disposa de:

- createExpression() que inicialitza una expressió i la registra al conjunt.
- deleteExpression() que simplement esborra una expressió del conjunt.

2.2.3. Expression

Aquesta classe és la representació del concepte d'expressió booleana que hem definit en el nostre sistema. Es tracta d'una classe abstracta, que defineix una herència completa i disjunta. És a dir, Expression no tindrà cap instància que no sigui de cap de les subclasses que hereten d'ella (And, Or, Not o Literal) i una instància d'alguna d'aquestes subclasses és d'un i només d'un tipus, no pot haver-hi una instància que sigui de la classe And i Or alhora, per exemple. Per aquest motiu, aquesta classe no té una constructora, perquè no hi haurà cap instància.

La classe Expression com a tal, ofereix dues funcions fonamentals com a públiques:

- create(): Que permet rebre una expressió i en genera tota l'estructura, instanciant les subclasses que siguin necessàries per donar-li forma i comprova que l'expressió sigui vàlida (es concreta què entenem per expressió vàlida i invàlida en l'apartat d'estructures de dades i algorismes d'aquesta documentació). Cal destacar que aquesta funció és estàtica (per això està subratllada en el diagrama), perquè no es cridarà des de cap instància, ja que no hi ha instàncies de la classe, sinó que s'usarà com una funció de classe. El fet que sigui estàtica provoca que les funcions privades que farà servir també ho siguin.

- evaluate(): La segona funció imprescindible que disposa l'expressió és la d'avaluar si un contingut compleix o no amb l'expressió booleana, a més tenint en compte si es vol que l'avaluació sigui case sensitive o no. La funció és abstracta perquè el resultat de l'avaluació depèn completament del tipus d'expressió, així que la implementació d'aquest mètode està escrit en cadascuna de les subclasses que hereten d'Expression.

Com s'ha esmentat, la funció create() usa un seguit de funcions privades, aquestes són:

- keysToAnds(): Que tracta les claus de tipus { } de l'expressió rebuda.
- recursiveDeconstruction(): Per realitzar la pròpia traducció cap a instàncies d'alguna de les subclasses.
- checkParentheses(): Que comprova si els parèntesis, (), obren i tanquen adientment.
- checkQuotes(): Que comprova que les cometes, "", obrin i tanquin correctament.
- unpack(): Per eliminar espais i parèntesis i poder analitzar-ne l'interior.
- findTopLevelOperand(): Que busca l'operand del nivell més alt en tant que parèntesis.

2.2.4. And

Aquesta classe és filla d'Expression i representa una expressió booleana formada per dues expressions que per tal d'avaluar cert cal que es compleixin les dues expressions. Per aquest motiu, aquesta classe té dos atributs privats, que són left i right, representats amb les associacions cap a Expression en el diagrama. A més, com que la classe és filla d'Expression, com s'ha esmentat, n'hereta les funcions públiques (no hereta cap atribut perquè no n'hi ha a Expression), és a dir, create() i evaluate(), i afegeix una creadora pública:

- And(): És la creadora pública per defecte, que permet instanciar aquesta classe.
- create(): S'hereta d'Expression, però no apareix en el diagrama en la classe And perquè no se sobreescriu la implementació del pare, Expression.
- evaluate(): En aquest cas sí, s'implementa l'avaluació d'un contingut tenint en compte les dues expressions que la configuren, i fent una avaluació lazy.

2.2.5. Or

Aquesta classe és filla d'Expression i representa una expressió booleana formada per dues expressions que per tal d'avaluar cert cal que es compleixi alguna de les dues expressions. Per aquest motiu, aquesta classe té dos atributs privats, que són left i right, representats amb les associacions cap a Expression en el diagrama. A més, com que la classe és filla d'Expression, com s'ha esmentat, n'hereta les funcions públiques (no hereta cap atribut perquè no n'hi ha a Expression), és a dir, create() i evaluate(), i afegeix una creadora pública:

- Or(): És la creadora pública per defecte, que permet instanciar aquesta classe.
- create(): S'hereta d'Expression, però no apareix en el diagrama en la classe Or perquè no se sobreescriu la implementació del pare, Expression.
- evaluate(): En aquest cas sí, s'implementa l'avaluació d'un contingut tenint en compte les dues expressions que la configuren, i fent una avaluació lazy.

2.2.6. Not

Aquesta classe és filla d'Expression i representa una expressió booleana formada per una expressió que per tal d'avaluar cert cal que no es compleixi allò que indica l'expressió. Per aquest motiu, aquesta classe té un atribut privat, que és inner, representat amb l'associació cap a Expression en el diagrama. A més, com que la classe és filla d'Expression, com s'ha esmentat, n'hereta les funcions públiques (no hereta cap atribut perquè no n'hi ha a Expression), és a dir, create() i evaluate(), i afegeix una creadora pública:

- Not(): És la creadora pública per defecte, que permet instanciar aquesta classe.
- create(): S'hereta d'Expression, però no apareix en el diagrama en la classe Not perquè no se sobreescriu la implementació del pare, Expression.
- evaluate(): En aquest cas sí, s'implementa l'avaluació d'un contingut tenint en compte l'expressió que la configura.

2.2.7. Literal

Aquesta classe és filla d'Expression i representa una expressió booleana formada per una frase. Per aquest motiu, aquesta classe té un atribut privat, value, representat amb l'associació cap a Phrase en el diagrama. A més, com que la classe és filla d'Expression, com s'ha esmentat, n'hereta les funcions públiques (no hereta cap atribut perquè no n'hi ha a Expression), és a dir, create() i evaluate(), i afegeix una creadora pública:

- Literal(): És la creadora pública per defecte, que permet instanciar aquesta classe.
- create(): S'hereta d'Expression, però no apareix en el diagrama en la classe Literal perquè no se sobreescriu la implementació del pare, Expression.
- evaluate(): En aquest cas sí, s'implementa l'avaluació d'un contingut tenint en compte la frase que configura aquesta expressió.

2.2.8. Phrase

La classe Phrase permet representar en el nostre diagrama el concepte de frase, entesa com una seqüència de paraules. D'aquesta manera, aquesta classe té com atribut privat una seqüència de paraules, representades en el diagrama amb l'associació cap a la classe Word. Aquesta associació és de composició, perquè en el nostre sistema no té sentit una paraula que no sigui de cap frase, ja que sempre qualsevol paraula és d'una frase. La classe Phrase no disposa de cap atribut més ni li hem assignat cap funció, ni tan sols la creadora. Això és perquè en la codificació, aquesta classe no serà implementada per nosaltres, sinó que usarem estructures de dades ja implementades en el llenguatge de programació que usem. Tot i això, l'hem volgut incloure en el diagrama perquè ens aporta el significat comentat.

2.2.9. Word

Aquesta classe representa el concepte de paraula, és a dir, de seqüència de caràcters, en el nostre esquema conceptual. Disposa d'un atribut privat, value, que emmagatzema la pròpia paraula, tot i que de la mateixa manera que la classe anterior, en la implementació aquesta classe no apareixerà com a tal, sinó que usarem estructures de dades ja existents per a representar aquest concepte. D'igual manera que amb Phrase, però, hem volgut incloure aquesta classe al diagrama perquè conceptualment la considerem necessària.

2.2.10. DocumentsSet

La classe DocumentsSet pretén representar el conjunt de documents que hi haurà donades d'alta a l'aplicatiu i cedir-li la responsabilitat de la gestió d'aquests documents. Es tracta d'una classe singleton que, doncs, té com atribut el singletonObject, la creadora per defecte privada i la funció pública getInstance().

A més, disposa d'altres atributs privats, com el numDocuments, que és un enter major o igual a 0 que indica el nombre de documents que té registrats. També té un conjunt anomenat presence, que serveix per emmagatzemar totes les paraules que formen part d'algun contingut d'algun document del sistema, sempre que no sigui una stop word (és a dir, una paraula sense aportació semàntica), i el nombre d'aparicions en diferents documents de cada paraula, sempre garantint la quarta restricció textual del diagrama. L'últim atribut de la classe és el representat amb la relació cap a la classe Document, documents, que consisteix en un conjunt que emmagatzema tots els documents que estan donats d'alta al sistema.

Pel que fa a les operacions, totes les públiques tenen una relació directa i, de fet la majoria amb el mateix nom, amb les funcions esmentades en la classe del CtrlDomain, que també generen una dependència cap a les classes Pair i Expression, tal com li succeïa al controlador. Com hem esmentat, per més detall sobre cada funció es pot consultar en la documentació generada amb Doxygen. En aquest punt, ens agradaria comentar que, a més de les funcions públiques, aquesta classe disposarà de tres funcions privades:

- getDocument(): Que serà usada per diferents funcions de les públiques per aconseguir un document del conjunt de documents.
- addPresence(): Servirà per actualitzar el conjunt de presence, registrant noves aparicions de paraules.
- removePresence(): Servirà per actualitzar el conjunt de presence, esborrant aparicions de paraules.

2.2.11. Document

La classe Document representa el concepte d'un document en el nostre sistema. Per definició, un document s'identifica per títol i autor, així que els primers atributs a esmentar són aquests dos, que per definició també, són dues frases. Per aquest motiu, Document disposa de dos atributs privats, title i author, que es veuen representats en el diagrama com dues associacions cap a la classe Phrase. A més, el document té un contingut, que es representa com un atribut privat sortit de la relació cap a la classe Content. El nom de l'atribut és content, perquè en el diagrama no hem especificat el nom de rol, així que en absència d'aquest, s'usa el nom de la classe en minúscules. De Document apareix una altra associació, en aquest cas cap a InternalDocument, donant peu a l'atribut privat internalDoc, que emmagatzemarà la representació interna del document en el sistema. Finalment, falta esmentar l'atribut language, que és de tipus TLanguage (enumeració ja esmentada en el CtrlDomain), que registra en quin idioma està el document, tant pel que fa a contingut com a títol.

Com podem veure en el diagrama, Document és la classe pare d'una herència disjunta i incompleta a dues subclasses, Txt i Xml. Aquesta herència permet representar quin era el format original del document quan es va carregar al sistema en cas que el document hagués estat carregat des d'un document ja existent en el dispositiu de l'usuari, que només acceptem que pugui ser de tipus .txt o .xml i, evidentment, no és possible que un document sigui dels dos tipus alhora (per això l'herència és disjunta). Ara bé, si el document s'ha donat d'alta des de l'aplicatiu, no tindrà cap instància en cap de les dues subclasses, perquè no té sentit el concepte de format original (per això l'herència és incomplete, pot ser que un document no sigui de cap de les dues subclasses).

Pel que fa a les funcions, trobem:

- Document(): Hi ha un seguit de creadores que accepten diferents paràmetres. Com hem esmentat a l'inici d'aquest apartat, per una explicació més detallada de cada funció i dels seus paràmetres, es pot consultar la documentació generada a partir de Doxygen. Només destacar que en algunes de les constructores un dels paràmetres és el format, que l'hem especificat de tipus TFormat com a enumeració, admetent els dos formats que representem en el nostre diagrama, txt o xml.
- getRelevantWords(): Permet consultar quines són les paraules rellevants del document (sense tenir en compte les stop words).
- compare_tf_idf(): Dona la possibilitat de comparar el document amb un altre usant l'estratègia tf-idf.

- compare_tf_boolean(): Dona la possibilitat de comparar el document amb un altre usant l'estratègia tf-boolean.
- queryRelevance(): Avalua la rellevància d'una query (seguit de paraules) sobre el document.

A més, també trobem dues funcions privades:

- termRelevance_tf_idf(): Usada per comparar el document amb l'estratègia tf-idf des de la funció compare_tf_idf.
- termRelevance_tf_boolean(): Usada per comparar el document amb l'estratègia tf-boolean des de la funció compare_tf_boolean.

2.2.12. Txt

La classe Txt representa un document que s'ha carregat al sistema en un format de text pla, és a dir, a partir d'una ruta local amb l'extensió .txt. Aquesta classe encara no està implementada en la nostra entrega, perquè encara no hem treballat la capa de persistència i de gestió de fitxers, però sí que ens permet representar un document que sabem que, en el seu format original, disposava d'una primera línia amb el títol, d'una segona línia amb l'autor i que a partir de la tercera hi havia tot el contingut. Ens serà útil tenir aquesta informació per tal de saber com exportar el document, per exemple.

2.2.13. Xml

La classe Xml representa un document que s'ha carregat al sistema en un format de llenguatge d'etiquetes xml, és a dir, a partir d'una ruta local amb l'extensió .xml. Aquesta classe encara no està implementada en la nostra entrega, perquè encara no hem treballat la capa de persistència i de gestió de fitxers, però sí que ens permet representar un document que sabem que, en el seu format original, disposava de les etiquetes o tags pel títol, l'autor i el contingut. Ens serà útil tenir aquesta informació per tal de saber com exportar el document, per exemple.

2.2.14. InternalDocument

La classe d'InternalDocument ens permet representar com entenem internament un document en el nostre sistema, és a dir, com el representem en format propietari. La classe disposa de dos atributs de classe, relevantWords, que emmagatzema les paraules rellevants del contingut del document, sense tenir en compte les stop words, i totalWords, que recull el nombre total de paraules del contingut del document.

Per altra banda, disposem de tres atributs privats de classe, és a dir, que són estàtics i per això estan subratllats en el diagrama. Aquests atributs, que són stopWords_ca, stopWords_en i stopWords_es, representen el conjunt d'stop words que hem definit pels tres idiomes que suporta el nostre aplicatiu, català, anglès i castellà, respectivament. Són atributs de classe perquè no depenen de la instància concreta d'internalDocument, sinó que són comuns per totes les representacions internes de tots els documents.

Sobre les funcions públiques, trobem:

- InternalDocument(): Disposem de diferents creadores per instanciar una representació interna, com s'ha mencionat, per més informació sobre els paràmetres i les diferències entre les funcions es pot consultar la documentació generada amb Doxygen.
- getRelevantKeyWords(): Que permet aconseguir les paraules rellevants emmagatzemades en la representació interna.
- newContent(): Dona la possibilitat d'actualitzar la informació interna en cas que es modifiqui el contingut del document representat.

A més, hem definit una funció privada:

- analizeContent(): Que servirà tant en la creació com en l'actualització de contingut per a poder definir adientment l'estructura interna.

2.2.15. Content

La classe Content en el diagrama representa el contingut d'un document. Per definició, el contingut d'un document és una seqüència de paraules, així que aquesta classe té una associació cap a la classe Phrase, que configura el contingut. Aquesta relació és una agregació, perquè un contingut s'entén estrictament com un conjunt de frases, tot i que hi ha frases que no formen part de cap contingut de documents, com per exemple les que configuren el value de la classe Literal. Esmentar que en la implementació, aquesta classe no

apareixerà com a tal, sinó que usarem estructures de dades ja existents per tal de representar aquest concepte. Tot i això, hem volgut representar aquesta classe en el diagrama perquè conceptualment ens aporta el matís semàntic esmentat.

2.2.16. Pair

La classe Pair representa un parell de valors, de tipus qualsevol. No aporta cap contingut semàntic al nostre diagrama, però és usada, com es pot veure, per diferents classes del model per tal de poder passar paràmetres i retornar valors de manera més còmoda. Només consta dels atributs del first i del second, que poden ser de qualsevol tipus, en funció del que necessitin les classes que l'usen. No té cap funció extra a part de les constructores, els getters i els setters.

3. Estructura de dades i algorismes

En aquest apartat descrivim les estructures de dades i els algorismes que hem usat per implementar les classes del domini que hem descrit, excepte les que ja hem esmentat i justificat que no hem codificat.

3.1. CtrlDomain

Els atributs del controlador usen els tipus de dades evidents: el singletonObject és de topus CtrlDomain, ds és de tipus DocumentsSet i es de tipus ExpressionsSet. Sí que volem justificar en aquest apartat la definició dels atributs ds i es, és a dir, podríem haver realitzat la implementació sense aquests dos atributs, però com que el controlador està constantment cridant funcions d'aquestes dues classes, d'aquesta manera ens estalviem haver d'anar demanant la instància a aquestes classes, que també són singleton.

Algorísmicament aquesta classe és senzilla, en tant que fa simplement d'intermediària entre les consultes o peticions que rep i aquestes dues altres classes. Les funcions createEmptyDocument(), deleteDocument(), existsDocument(), getContentDocument(), updateContentDocument(), getLanguageDocument(), updateLanguageDocument(), listAllDocuments(), listSimilars(), listTitlesOfAuthor(), listAuthorsByPrefix() i listByQuery() criden directament a funcions amb el mateix nom i paràmetres, pràcticament, de funcions de DocumentsSet. Per altra banda, createExpression(), deleteExpression() i modifyExpression() usen funcions de ExpressionsSet. Només destacar que la funció listByExpression() és la que requereix la interacció amb el DocumentsSet i l'ExpressionsSet, perquè primer ha de demanar l'expressió a ExpressionsSet i posteriorment aconseguir els documents que la compleixen, a partir de DocumentsSet. Ho hem decidit d'aquesta manera per evitar l'acoblament entre els dos conjunts.

3.2. ExpressionsSet

En la implementació d'aquesta classe hem hagut de decidir com materialitzar els dos atributs privats de la classe. Pel que fa a singletonObject, evidentment el tipus triat és ExpressionsSet. Per altra banda, sobre l'atribut expressions, hem hagut de plantejar quina estructura de dades s'adequava més a les necessitats d'aquesta classe. Primerament vam plantejar una estructura de Set<Expression>, vector o llista que simplement emmagatzemés

totes les expressions, però això impedia accedir ràpidament a una expressió a partir d'un identificador (un string), ja que hagués calgut recórrer linealment l'estructura preguntant a l'expressió el seu identificador. Per aquest motiu finalment hem usat la classe Map de Java, definit com Map<String, Expressions>, on la clau és el String que identifica l'expressió (el que ha introduït l'usuari per donar-la d'alta) i com a valor es correspon l'expressió que s'ha creat a partir d'aquell String. D'aquesta manera, veure si existeix una expressió, trobar-les i afegir-ne de noves comprovant que no es repeteixin són més eficients i fàcils de programar.

La següent decisió a prendre era el tipus de map. En Java Map és una interfície que usa diferents tipus d'estructures clau-valor, després d'analitzar-ne unes quantes, vam decidir emprar HashMap. Aquest tipus de dades ens garanteix un temps constant en l'accés i a l'hora d'afegir parells de clau-valor. L'únic possible inconvenient en aquesta estructura és que no garanteix cap ordre, però això no ens suposa cap problema, perquè en cap moment requerim d'una ordenació d'expressions.

Un cop triats els tipus dels paràmetres, la implementació de les funcions és força senzilla algorísmicament parlant. L'estructura de map, com comentàvem, facilita aconseguir les expressions (getExpression()), donar-ne d'alta noves (createExpression()) i esborrar-ne (deleteExpression()) que són les funcionalitats bàsiques d'aquesta classe. A més, alhora de llençar excepcions també ens facilita la feina, perquè hem pogut aprofitar que el mètode put retorna el valor previ que tenia la clau donada (per tant a l'hora de crear, si no es retorna null hem de llençar l'excepció) i que el mètode remove de map retorna el valor previ que tenia la clau donada (així que per esborrar, si el valor retornat és null, vol dir que no hi havia cap expressió amb aquell identificador).

3.3. Expression

Abans d'explicar com funciona aquesta classe, definirem quines coses considerem que ha de complir una String per ser candidata a ser donada d'alta com a expressió booleana al sistema. En resum, són les següents:

1. Si conté parèntesis, aleshores han de tancar bé, n'hi ha d'haver tants d'obrir com de tancar i estan ben enniuats.
2. Si conté claus, aleshores han de tancar bé, n'hi ha d'haver tantes d'obrir com de tancar, i no són multinivell.

3. El nombre de cometes és parell, i el que contenen es considerarà frase sigui el que sigui. Al ser les cometes d'obrir i tancar exactament el mateix caràcter, no seran multinivell i una frase no podrà contenir mai cometes.
4. En cap cas uns parèntesis, unes claus, o un operador |, & o ! es consideraran paraula o part d'una paraula. S'admet que apareguin entre cometes, però, cas en el que es consideraran com a part de la frase i no com operadors lògics.
5. Han de consistir únicament en una paraula o una frase, o bé s'han de poder descomposar recursivament en un operand i una o dues subexpressions associades a aquest operand segons el tipus.
    - Noteu que per aquesta definició no és possible trobar consecutivament dues paraules, dues frases, o una frase i una paraula.
6. La String buida no és una expressió vàlida.
    - Noteu que això implica que a un operador mai li podran faltar operands.

La implementació d'aquesta classe l'hem fet a partir d'una herència abstracta, basant-nos en el fet que una expressió consisteix en una de les següents opcions:

1. Un Literal, una paraula o frase
2. Not d'una expressió
3. And de dues expressions
4. Or d'una expressió

Totes aquestes classes les trobareu explicades més avall en aquest mateix document. En particular totes elles implementen un mètode abstracte booleà que serveix per avaluar una Expressió implícita ja creada donada una String de contingut i un booleà que indica si cal diferenciar majúscules i minúscules.

A banda d'aquest mètode abstracte, la superclasse Expressió ofereix també un mètode estàtic per validar Strings i crear-ne l'instància d'Expression corresponent. Si la String passada per paràmetre verifica les propietats descrites abans, es retorna una instància d'Expression, i en cas contrari llança una excepció indicant que el format no és el correcte.

La implementació d'això segueix aquest ordre:

1. Es verifica que els parèntesis, les claus i les cometes que tanquen
2. Es substitueixen totes les claus que no estan dins de cometes a un conjunt de & dins de la String
3. Es desempaqueta la String, s'eliminen espais i parèntesis al principi i al final

4. S'intenta descomposar recursivament la String, **en aquest ordre**:
    - Or de dues expressions: si es troba una | en el nivell més extern dels parèntesis no dins de cometes, es parteix la String en dos trossos i es calcula l'expressió a que corresponen recursivament tornant al pas 3 per cada una. Amb aquestes Expression es crida la constructora de Or i es retorna l'Expression
    - And de dues expressions: anàloga a l'anterior
    - Not d'una expressió: només si el primer caràcter de la String és !, es resol recursivament per la resta de la String, es crida a la constructora de Not amb aquest resultat i es retorna l'Expression
    - Literal: es retornarà excepció si la String conté ! o espais que no estiguin dins de cometes, o si conté dues frases seguides. Si no és així, s'agafarà la String i amb ella es construeix i retorna un Literal

Aquest procediment garanteix que s'estan verificant les propietats descrites al principi, i a més crea correctament l'Expression associada a la String, tal i com es veu en els tests.

3.4. And

Cada instància d'aquesta classe representa la conjunció de dues subexpressions booleanes. Els dos atributs de la classe, per tant, són de tipus Expression. La classe ofereix una creadora per valor, que inicialitza una instància amb les seves dues subexpressions, i també implementa el mètode abstracte per avaluar la And implícita. Aquesta avaluació és lazy i s'implementa recursivament cridant al mètode per avaluar que ofereixen les dues subexpressions i amb l'operador de Java &&.

3.5. Or

És totalment anàloga a l'anterior amb la diferència que representa la disjunció de dues subexpressions booleanes. També ofereix una creadora per valor amb idèntic funcionament al de And. Similarment, en el mètode que presenta per avaluar, es fa servir avaluació lazy, i s'implementa recursivament i amb l'operador de Java ||.

3.6. Not

Com els altres operadors, la classe Not conté un atribut Expression (només un) que representa la subexpressió que es nega. La constructora per valor, doncs, rep una instància de Expression que inicialitza aquest atribut d'una instància nova de Not. La implementació que es fa del mètode per avaluar expressions booleanes consisteix en cridar recursivament aquest mateix mètode per la subexpressió i simplement retornant el resultat negat.

3.7. Literal

Com no podia ser d'una altra manera, fem servir un String per representar les paraules o frases que formen part d'una expressió booleana. A més, aquesta classe ofereix també com les anteriors una creadora per valor, i un mètode per avaluar una instància de Literal donat un contingut i el booleà que indica si s'ha de diferenciar entre minúscules i majúscules. Es tracta d'un mètode que busca la paraula o frase representada per aquest Literal dins del contingut passat. Aquest segon mètode s'ha implementat de manera que no es retorni cert quan es troba el que es buscava dins del contingut però només com a prefix, infix o sufix d'una altra seqüència de caràcters més llarga. La implementació es fa buscant la frase o paraula emmagatzemada en el Literal implícit però precedida i seguida de caràcters de text especials. Aquests caràcters són caràcters de text que no són alfanumèrics, entre ells l'espai i la coma, per exemple. En el cas que no es vulgui diferenciar minúscules de majúscules, abans d'això senzillament es passen a minúscula tant el contingut com la paraula o frase que s'ha de buscar. La classe fa servir un mètode estàtic privat que donat un caràcter retorna cert si i només si es un dels caràcters que hem considerat de text especials.

3.8. DocumentsSet

En la implementació d'aquesta classe hem hagut de decidir com materialitzar els quatre atributs privats de la classe. Pel que fa a singletonObject, òbviament tipus triat és DocumentSet. El següent atribut és documents, per aquest atribut vam haver de pensar quina estructura de dades s'adequa més a les necessitats i l'ordre dels atributs dins la classe. Per temes d'eficiència vam considerar que la millor forma era utilitzar un Map<String, Map<String, Document>> on el String inicial és el nom de l'autor i el segon el títol del document, aquesta decisió va ser presa per les funcions que hem d'implementar on les funcions de buscar per prefix o per autor fan que necessitem que el Map extern pugui buscar pel nom de l'autor. Per altra banda, en el cas d'utilitzar un Map i que sigui de tipus

HashMap, això és pel fet que l'altra opció plantejada va ser un TreeMap (que es troba ordenat), però en l'àmbit d'eficiència l'única funció que ho aprofita és buscar autor per prefix, fet que ha fet que considerem que no contraresta el cost que suposa ordenar cada cop el Map.

Per altra banda, l'atribut numDocuments, un enter que indica el nombre de documents, vam afegir-ho per no haver de recórrer cada cop el mapa de documents per treure el nombre de documents i així guanyar en eficiència, ja que és un atribut que cada cop que fem una comparació per buscar similars o amb un query usem.

Finalment, presence és un Map<String,Integer> que guarda en quins documents apareix un string determinat, aquest atribut serveix per als algoritmes de l'espai vectorial que necessita aquesta informació i vam materialitzar en atribut per no calcular-ho cada cop que cridem a alguna d'aquestes funcions.

Després de decidir els paràmetres i el tipus d'aquests, la majoria de funcions va ser senzilla algorísmicament, l'únic que va portar una certa dificultat ve de les funcions listbyQuery i listSimilar que exigeixen fer una ordenació segons el pes obtingut, fent que vam implementar una ordenació del mapa segons el pes, sense eliminar en cas que el valor del pes estigui repetit. Això ho vam implementar a través de la classe Collections que permet fer un sort del mapa per sense eliminar

3.9. Document

La classe document és la representació d'un document en el nostre sistema. Per això, consta de cinc atribut senzills de tipus String: author, title, originalFormat i language. Tot i així, les dades que s'extreuen del contingut d'aquest document, hem decidit analitzar-les i guardar-les en una classe a part (InternalDocument), que en gestioni el tractament i ens retorni només allò que se'n consideri informació rellevant. Per aquest motiu, cada document constarà d'un únic atribut internalDoc de tipus InternalDocument, responsable de gestionar la informació associada a aquell document, seguint el model d'espai vectorial.

Els algorismes que implementa aquesta classe son els algorismes de comparació entre documents (seguint dos estratègies de pesos diferents) i l'algorisme de rellevància d'una query de p paraules. L'algorisme tf (term frequency) és un algorisme de comparació de documents que cerca coincidències entre les paraules del contingut de dos documents i les

pondera en base a uns pesos per a obtenir un índex que representa el grau de semblança entre dos documents. Cal destacar que hem considerat que aquells documents que estiguin en idiomes diferents els correspon un grau de semblança null.

El tf-booleà, és aquella versió de l'algorisme en el que els pesos assignats a cada paraula són o bé 0 o bé 1. És a dir, al comparar dos documents, per cada paraula clau d'un d'ells, esbrinem si aquesta paraula també apareix en l'altre, cas en el que sumem u a l'índex de semblança.

En una versió més complexa d'aquest algorisme, anomenada tf-idf (term frequency - inverse term frequency), els pesos assignats no només dependran de si la paraula es troba o no dins del document, sino que s'assignarà més pes a aquells paraules que apareguin múltiples cops en el document (term frequency) i a més a més, es perjudicarà aquelles paraules que apareguin repetidament en tots els documents del sistema, que es consideraran paraules no tan rellevant a nivell de significat. Per a avaluar la rellevància d'una query, hem decidit replicar l'algorisme de tf-idf, però en comptes d'usar el contingut d'un altre document, usant les paraules donades per la query.

3.10. InternalDocument

La classe InternalDocument representa el format propietari dels nostres documents. Implementa el concepte d'espai vectorial que emmagatzema totes les paraules claus d'un contingut donat. Per a fer aquesta estructura hem esclit un Map<String, Integer>, que ens mapejarà el nombre d'ocurrències d'una paraula clau concreta en el contingut del document, de manera que tindrem entrades <paraula clau, aparicions>. Hem escollit representar-ho així no guardant els valors de tf-idf directament per diverses raons. Primer de tot, guardant només les aparicions i mantenint informació global del documentSet ens permet carregar i descarregar documents més ràpidament. Guardant els valors de ti-idf, aquest haurien de ser recalculats cada cop que s'afegís/s'esborrés un document en el sistema. A part, ens aporta flexibilitat a 'hora d'implementar nous algorismes d'assignació de pesos (com el tf-idf i el tf-booleà), ja que si en guardesim només el tf-idf, aquesta flexibilitat d'elecció d'algorismes es veuria perjudicada. A canvi, hem de pagar un petit cost per operacions numèriques extres (suma i multiplicació) al fer cerces i comparacions, però hem considerat que les avantatges superaven aquest desavantatge.

L'únic algorisme del que disposa aquesta classe és l'anàlisi del contingut d'un document. Aquest té una implementació simple. Donat un contingut en forma de String, en separa les

paraules utilitzant el mètode split, i comprova si és o no una stopword de l'idioma en el que està el document. En cas que no ho sigui, l'afegeix al mapa d'aparicions, creant una nova entrada si és el primer cop que apareix o afegint 1 al nombre d'aparicions actuals d'aquella paraula clau.

3.11. Pair

La implementació d'aquesta classe intenta representar només el concepte bàsic d'un parell de valors per tal de poder ser usada com a utilitat en altres classes. Per fer-la genèrica i no haver d'implementar pairs per diferents tipus de variable d'aquests parells de valors, els camps del first i del second del parell accepten qualsevol tipus de dades. Així doncs, el camp first serà de tipus F i el camp second de tipus S, on F i S serveixen com a template per qualsevol estructura de dades.

Pel que fa a les operacions, només hem implementat les més bàsiques que necessitem en el nostre sistema: una constructora amb i sense inicialització, i els getters i setters dels dos atributs per separat, que consisteixen en el retorn o l'assignació, respectivament, d'algun dels dos atributs.

4. Classes implementades per cada membre

4.1 Classes del domini implementades per cada membre

- Document — ariadna.cortes.danes
- InternalDocument — ariadna.cortes.danes
- DocumentsSet — pau.duran.manzano, marc.lopez.domenech
- Expression — pau.duran.manzano, marc.valls.camps
- ExpressionsSet — pau.duran.manzano
- CtrlDomain — pau.duran.manzano
- And — marc.valls.camps
- Or — marc.valls.camps
- Not — marc.valls.camps
- Literal — marc.valls.camps
- Pair — pau.duran.manzano

4.2 Testos implementats per cada membre

- Document: — ariadna.cortes.danes
- InternalDocument: — ariadna.cortes.danes
- DocumentsSet: — pau.duran.manzano, marc.lopez.domench
- Expression — marc.valls.camps
- ExpressionsSet — pau.duran.manzano
- CtrlDomain (Driver) — ariadna.cortes.danes, pau.duran.manzano
- And — marc.valls.camps
- Or — marc.valls.camps
- Not — marc.valls.camps
- Literal — marc.valls.camps
- Pair — pau.duran.manzano

4.3 Stubs implementats per cada membre

- InternalDocument: — ariadna.cortes.danes
- Document — pau.duran.manzano
- Expression — pau.duran.manzano
- True — pau.duran.manzano
- False — pau.duran.manzano

5. Llibreries externes utilitzades

En aquesta primera entrega hem usat dues llibreries externes. Els seus corresponents .jar es troben en el directori lib del nostre projecte. Les dues són necessàries per a realitzar els tests unitaris, que hem realitzat de totes les classes del domini implementades, perquè hem emprat junit, tal com s'ha explicat en la teoria de l'assignatura. Les dues llibreries són:

5.1. hamcrest-core

En la versió 1.3. L'hem usat en alguns tests unitaris per tal de poder afegir predicats i restriccions a valors especificats ens els asserts de junit, per especificar declarativament regles que s'han de complir.

5.2. junit

En la versió 4.12. És l'eina fonamental que hem emprat per tal de realitzar els tests unitaris sobre les classes del domini implementades, excepte el controlador. Consisteix en un framework que permet executar les classes de manera controlada i examinar el comportament d'aquestes classes. A més d'usar la llibreria per general els diferents tests suites, també n'hem fet servir el seu test runner per defecte per tal d'executar els tests realitzats. Per últim, només comentar que tot i que ja existeixin versions posteriors (com la 5), hem usat la 4.12 seguint les instruccions de les classes de teoria de l'assignatura.