Gestor de documents

Identificador de l'equip: 11.1

Pol Camprubí Prats: pol.camprubi.prats@estudiantat.upc.edu
Pol Mañé Roiger: pol.mane@estudiantat.upc.edu
Isaac Roma Granado: isaac.roma.granado@estudiantat.upc.edu
Juli Serra Balaguer: juli.serra@estudiantat.upc.edu

Versió de lliurament 2.1
2ª Entrega PROP
16/12/2022

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

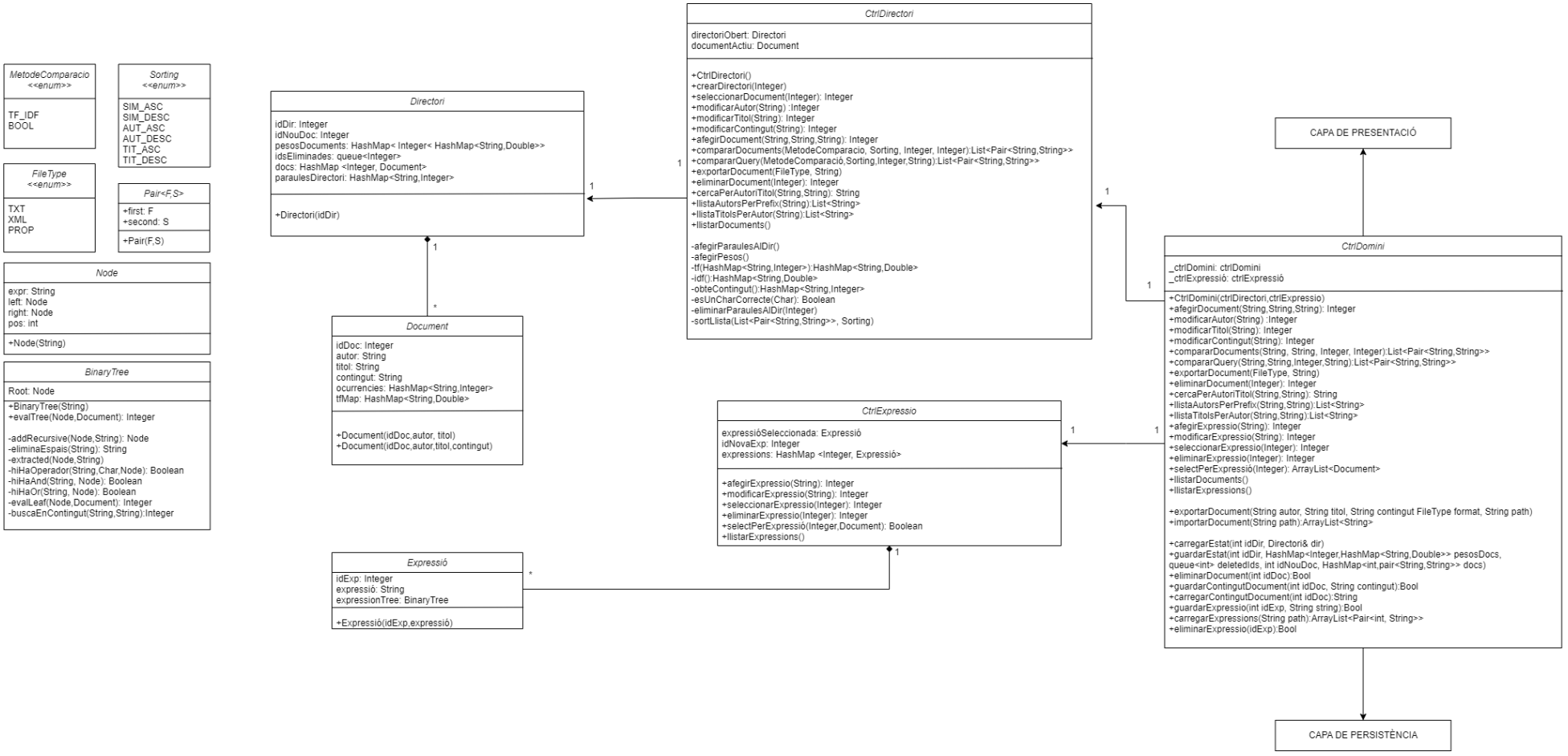
FIB

Índex

1. Diagrama del model conceptual

1.1 Disseny del diagrama de model conceptual

A continuació mostrem el diagrama del model conceptual. El diagrama es pot trobar detallat al directori de Google Drive i en la carpeta DOCS d'aquest repositori.



Restriccions textuals:

-Claus primàries:

(**Directori,** Integer: idDir)

(**Document,** Integer: idDoc)

(**Expressió**, Integer: idExp)

-Claus Candidates:

(**Document,** autor + títol)

1.2 Descripció de les classes

Directori: La classe Directori és aquella que conté l'agrupació de tots els Documents de l'usuari. S'encarrega principalment de tenir guardats i ubicats els documents i guardar la informació corresponent de cada un per tal de que el CtrlDirectori pugui realitzar les operacions sense problema.

Document: La classe Document representa un dels documents compresos dins el directori. Aquesta classe té un atribut privat i tres de públics. L'atribut privat és la id (PK de la classe) que serveix al sistema per identificar el document que l'usuari està intentant accedir en aquell moment. Els atributs públics són: autor, títol i contingut. Com hem dit anteriorment, autor i títol formen una clau candidata ja que no ens interessa tenir guardat un document on aquests dos atributs siguin el mateix.

Expressió: La classe Expressió guarda tota la informació necessària per tal de fer les operacions sobre ella. En aquest cas tenim una expressió que representa un String, un atribut id que fa la mateixa funció que en la classe Document i un expression Tree necessari per tal de calcular quins documents compleixen l'expressió.

Arribats a aquest punt, ja tenim descrites totes les classes del nostre sistema. Fixem-nos que cap classe té operacions importants, només setters, getters i creadores. A continuació, veurem els controladors del nostre sistema que són els encarregats d'executar els mètodes necessaris per fer les operacions corresponents:

CtrlDomini: El controlador de Domini s'encarrega d'instanciar la resta de controladors de la capa de domini i proporciona una interfície més pròxima a l'usuari dels mètodes de la resta de controladors com el CtrlDirectori o CtrlDocument.

CtrlDirectori: El controlador de directori s'encarrega de la gestió dels mètodes de les classes Directori i Document. Es comunica amb Directori per tal de realitzar totes les funcionalitats referents a aquests. Els atributs d'aquesta

classe són el directori obert que s'està utilitzant i el document que té obert en aquell moment l'usuari

CtrlExpressió: El controlador d'Expressions s'encarrega de gestionar els mètodes guardar, modificar, eliminar i comparar els documents del directori amb les expressions que l'usuari ha afegit anteriorment.

Finalment, descrivim les classes complementàries de l'UML:

Pair<F,S>: La classe Pair és una classe genèrica per tal de crear tuples de dos elements. L'ús del pair és per aquelles operacions que s'esperi el retorn de dos variables bàsiques (Strings. Integers, Booleans,...).

Node: La classe node és creada per tal de fer la classe BinaryTree que explicarem a continuació. Un node té els atributs següents: un string que representa la expressió, un integer que recorre l'expressió com ho faria un punter i dos nodes corresponents al fill esquerra i el fill dret.

BinaryTree: La classe BinaryTree és l'encarregada d'estudiar l'expressió que ha ofert l'usuari, convertir-la en un arbre binari segons unes instruccions donades i ser capaç d'avaluar la funció resultant de la manera més simple possible. Tota aquesta classe està implementada utilitzant recursivitat.

1.3 Descripció de atributs i mètodes de cada classe

Totes les operacions que poden tenir una excepció (aquelles que tenen return int) retornen un dels següents resultats:

- 10 si l'operació ha funcionat correctament.
- 11 s'executa correctament però avisa que l'objecte eliminat estava en ús per l'usuari
- 20 si l'operació té un error per objecte ja existent
- 30 si l'operació té un error per un valor null o paràmetre no vàlid.
- 31 si l'operació referencia a un objecte null

Classe Document

int idDoc
Representa el número identificador d'un document.

String autor
Representa l'autor d'un document.

String títol
Representa el títol d'un document.

String contingut
Representa el contingut d'un document.

HashMap<String, Integer> ocurrencies
Representa les vegades que apareix cada paraula en el contingut del document.

HashMap<String, Double> tfMap
Representa el càlcul de l'operació tf per aquell document

public Document(int idDoc, String autor, String titol)
Constructora de document.

public Document(int idDoc, String autor, String titol, String contingut)
Constructora de document.

Classe Directori

int iDir
Representa el número identificador d'un directori.

HashMap<Integer, HashMap<String, Double>> pesosDocs
Representa la taula de pesos del nostre sistema.

Queue<Integer> deletedIds
Representa els números identificadors dels documents eliminats.

int idNouDoc
Representa un contador per atribuir la id corresponent al nou document.

HashMap<Integer, Document> docs
Representa els documents que existeixen en el directori.

HashMap<String, Integer> paraulesDirectori
Representa les paraules que trobem en el directori seguit de les ocurrències.

public Directori(int idDir)
Constructora de directori.

Classe Expressió

int idExp
Representa el número identificador d'una expressió.

String expressio
Representa una expressió.

BinaryTree ExpressionTree
Representa l'arbre binari d'una expressió.

public Expressio(Integer idExp, String expressio)
Constructora d'expressió.

Classe CtrlDirectori

Directori directoriObert
Representa el directori que s'està utilitzant.

Document documentActiu
Representa el document que l'usuari està usant

public CtrlDirectori()
Constructora de controlador de directori.

public void crearDirectori(int idDir)

Operació per crear un directori en el nostre sistema.

public int seleccionarDocument(int idDoc)
Operació per obrir un document que ja teníem carregat dins el nostre sistema.
@param idDoc és l'identificador del document que volem obrir.

public int modificarAutor(String autor)
Operació per modificar l'autor d'un document.
@param autor és el nou nom d'autor que es vol utilitzar pel document.

public int modificarTitol(String titol)
Operació per modificar el títol d'un document.
@param títol és el nou nom del títol que es vol utilitzar pel document.

public int modificarContingut(String contingut)
Operació per modificar el contingut d'un document.
@param contingut és el nou contingut que es vol utilitzar pel document.
A part de modificar el contingut es realitzen les següents operacions:

- S'actualitza el atribut paraulesDirectori restant 1 a les paraules que contenia el document
- Inicialitzem mb el valor corresponent l'atribut ocurrencies del document modificat.
- Inicialitzem amb el valor corresponent l'atribut tfMap del document modificat.
- Afegim les paraules del nou contingut a l'atribut paraulesDirectori.
- Afegim els pesos corresponents a l'atribut pesosDocs.

public int afegirDocument (String autor, String titol, String contingut)
Operació per donar d'alta un document en el directori.
@param autor representa l'autor del document que es vol afegir.
@param títol representa el títol del document que es vol afegir.
@param contingut representa el contingut del nou document.
A part de crear el Document es realitzen les següents operacions:

- S'afegeix el document a l'atribut docs del directoriObert
- Inicialitzem amb el valor corresponent l'atribut ocurrencies de document creat
- Inicialitzem amb el valor corresponent l'atribut tfMap del document creat
- Afegim les paraules del document a l'atribut ParaulesDirectori
- Afegim els pesos corresponents a l'atribut pesosDocs

private void afegeixParaulesAlDir()
Operació per afegir noves parules a ParaulesDirectori cada cop que es crea un document nou o que es modifica el contingut d'un document ja existent

private void afegeixPesos()
Operació per calcular i afegir els pesos de cada paraula en el directori.

private HashMap<String, Double> tf(HashMap<String, Integer> paraules)
Operació per calcular el tf d'un document concret
@param paraules representa el HashMap on es guarden les ocurrències de cada paraula en el document.

private HashMap<String, Double> idf()
Operació per calcular l'idf en el nostre sistema.

private HashMap<String, Integer> obteContingut()
Operació que serveix per obtenir el contingut del document actiu.

private boolean esUnCharCorrecte(char c)
Operació que serveix per saber si un caràcter compleix els requisits per ser considerat vàlid
@param c representa el caràcter que volem evaluar.

public enum METODE_COMPARACIO
Enumeració dels diferents tipus de comparació que podem realitzar (TF_IDF i BOOL).

public enum SORTING
Enumeració de les possibles ordenacions al comparar documents (similaritat ascendent, similaritat descendent, autors ascendent, autors descendent, títol ascendent i títol descendent).

public List<Pair<String, String>> compararDocuments(METODE_COMPARACIO m, SORTING s, Integer k, Integer IdDoc)
Operació que serveix per comparar un document amb els que tenim en el directori i obtenir-ne els k més semblants. A part, el resultat el podem obtenir mitjançant dos mètodes diferents, el TF-IDF i el BOOLEÀ (a decidir per l'usuari) i podem ordenar aquest resultat segons els criteris que ens ofereix el SORTING (similaritat ascendent, similaritat descendent, autors ascendent, autors descendent, títol ascendent i títol descendent).
@param m representa els diferents tipus de comparació que podem realitzar.
@param s representa el criteri d'ordenació pel resultat.
@param k representa el nombre de documents semblants que volem obtenir.
@param IdDoc representa el número identificador d'un document.

private void sortLlista(List<Pair<String, String>> llistaSemblants, SORTING s)
Operació que serveix per ordenar els documents semblants segons els diferents criteris d'ordenació que ens ofereix el SORTING (similaritat ascendent, similaritat

descendent, autors ascendent, autors descendent, títol ascendent i títol descendent).
@param llistaSemblants representa la llista que es vol ordenar.
@param s representa el criteri d'ordenació pel resultat.

public List<Pair<String, String>> compararQuery(METODE_COMPARACIO m, SORTING s, Integer k, String paraules)
Operació que serveix per obtenir els documents més rellevants pel que fa a contingut segons la query de paraules passada com a paràmetre. Aquest mètode crea un nou document dins el sistema (que és eliminat un cop s'acaba l'operació) i té fa les mateixes funcionalitats que l'operació compararDocuments.
@param m representa els diferents tipus de comparació que podem realitzar.
@param s representa el criteri d'ordenació pel resultat.
@param k representa el nombre de documents rellevants que volem obtenir.
@param paraules representa la query que l'usuari entra com a variable per examinar.

public enum FILETYPE
Mètode que enumera els diferents tipus de formats en els que podem tenir un document (TXT i XML).

public void exportarDocument(FILETYPE format, String path)
Operació per exportar el document del directori a un path elegit.
@param format es correspon en quin format es desitja exportar el document.
@param path es correspon al camí desitjat per tal de guardar el document.

public int eliminarDocument(int idDoc)
Operació per eliminar un document del directori.
@param idDoc és l'identificador del document que es vol eliminar del directori.
Abans d'eliminar el document, el sistema:

- Resta les paraules del document a l'atribut paraulesDirectori
- Elimina el document de la taula de pesos (PesosDocs)
- Recalcula els pesos amb els documents que queden
- Elimina el document de l'atribut docs
- Afegeix la id del document borrat a la cua de ids

private void eliminarParaulesAlDir(int idDoc)
Operació per eliminar les paraules del document idDoc en el directori.
@param idDoc representa el número identificador del document que volem eliminar.

public String cercaPerAutoriTitol(String autor, String titol)
Operació per cercar un document del directori segons títol i autor.
@param autor representa l'autor del document que estem cercant.

@param títol representa el títol del document que estem cercant.

public List<String> llistaAutorsPerPrefix(String pre, SORTING s)
Operació que retorna una llista amb els documents que tenen un autor que comença amb el prefix passat com a paràmetre.
@param pre representa el prefix pel qual volem realitzar la cerca.
@param s representa el criteri d'ordenació pel resultat.
El resultat es pot ordenar segons les dues opcions de SORTING ofertes a l'usuari (ordenació per nom d'autors ascendent i descendent).

public List<String> llistaTitolsPerAutor(String autor, SORTING s)
Operació que retorna una llista amb els documents que tenen com a autor l'autor que es passa per paràmetre.
@param autor representa l'autor pel qual volem fer la cerca.
@param s representa el criteri d'ordenació pel resultat.
El resultat es pot ordenar segons les dues opcions de SORTING ofertes a l'usuari (ordenació per nom d'autors ascendent i descendent).

public ArrayList<String> llistarDocuments()
Operació que afegeix en un arraylist l'identificador, l'autor i el títol de tots els documents que hi donats d'alta en el moment donat al sistema. L'utilitzarem de cara a la comunicació entre capes.

Classe CtrlExpressio

Expressio expressioSeleccionada
Representa l'expressió sobre la que es treballa.

Integer IdNovaExp
Representa el número identificador de la nova expressió.

HashMap<Integer,Expressio> expressions
Representa les expressions que hi han en el directori.

public CtrlExpressio()
Constructora de CtrlExpressio.

public int seleccionarExpressio (Integer idExp)
Operació que serveix per seleccionar l'expressió seleccionada dins el sistema
@param idExp representa el número identificador de l'expressió que volem seleccionar.

public int afegirExpressio(String expressio)

Operació que serveix per afegir una expressió al directori. L'operació actualitza l'atribut expressions de la classe CtrlExpressio afegint la nova expressió si no existia ja en el directori.
@param expressió representa l'string que l'usuari reconeix com l'expressió per fer la cerca.

public int modificarExpressio(String exp)
Operació que serveix per modificar l'expressió seleccionada amb un nou text corresponent. Es modifica l'expressió sempre que no existeixi ja una expressió en el directori amb el mateix valor que l'expressió passada com a paràmetre.
Si es modifica l'expressió, es crea un nou BinaryTree de la nova expressió.
@param exp representa la nova expressió.

public int eliminarExpressio(int idExp)
Operació que serveix per eliminar l'expressió amb idExp del sistema.
@param idExp representa la id de l'expressió que volem eliminar.

public boolean selectPerExpressio(Integer idExp, Document document)
Operació que serveix per buscar, de tots els documents que nosaltres tenim en el sistema, aquells que compleixen l'expressió passada com a paràmetre.
Retorna una llista amb els documents que compleixen l'expressió.
Es crida a l'operació evalTree de la classe BinaryTree per calcular el resultat.
@param idExp representa el número identificador de l'expressió que volem avaluar.
@param document representa el document possible a avaluar.

public ArrayList<String> llistarExpressions()
Operació que llista en un arraylist totes les expressions booleanes que hi ha en el sistema, és a dir, que no han estat borrades. L'utilitzarem de cara a la comunicació entre capes.

Classe BinaryTree

class Node:
Representa un node del BinaryTree.

Node root
Representa el node arrel.

public BinaryTree(String exp)
Creadora de la classe BinaryTree.
Crea un BinaryTree de l'expressió que li entra com a paràmetre.
@param exp representa l'expressió sobre la que es vol crear el BinaryTree.

private Node addRecursive(Node current, String exp)
Operació que serveix per crear l'arbre binari de manera recursiva. El codi té com a prioritats els parèntesis, les ands i, finalment, les ors.
@param current representa el node en que volem treballar.
@param exp representa l'expressió que estem utilitzant.

private String eliminaEspais(String exp)
Operació que serveix per eliminar els espais d'una expressió.
Per tal de eliminar els espai es crida a la funció isWhitespace per determinar si tenim un espai en blanc o no.
@param exp representa l'expressió de la qual es volen eliminar els espais.

private void extracted(Node current, String exp)
Operació que serveix per avaluar l'expressió dividint-la en dues parts tenint com a node actual el operador de l'expressió.
@param current representa el node en que volem treballar.
@param exp representa l'expressió que estem utilitzant.

private boolean hiHaOperador(String exp, char op, Node current)
Operació que serveix per comprovar si una expressió conté un operador.
@param exp representa l'expressió que estem utilitzant.
@param op representa l'operador que estem buscant.
@param current representa el node en que volem treballar.

private boolean hiHaAnd(String exp, Node current)
Operació que serveix per comprovar si hi ha l'operador AND en l'expressió passada com a paràmetre. Per tal de fer la comprovació es crida a la funció hiHaOperador.
@param exp representa l'expressió que estem utilitzant.
@param current representa el node en que volem treballar.

private boolean hiHaOr(String exp, Node current)
Operació que serveix per comprovar si hi ha l'operador OR en l'expressió passada com a paràmetre. Per tal de fer la comprovació es crida a la funció hiHaOperador.
@param exp representa l'expressió que estem utilitzant.
@param current representa el node en que volem treballar.

public static int evalTree(Node current, Document d)
Operació que ens serveix per avaluar el tree de manera recursiva. El current és el node en que estem treballant i el document representa l'objecte que nosaltres estem passant a la funció per veure si compleix l'expressió.
@param current representa el node en que volem treballar.
@param d representa el document que estem comparant.

public static int evalLeaf(Node current, Document d)

Quan la funció evalTree troba que el node és una fulla, crida al mètode evalLeaf que analitza i calcula cada fulla de l'arbre.
@param current representa el node en que volem treballar.
@param d representa el document que estem comparant.

private static int buscaEnContingut(String word, String contingut)
Operació que serveix per comprovar si una paraula apareix en el contingut d'un document.
@param word representa la paraula que volem cercar.
@param contingut representa el contingut del document en el que volem fer la cerca.

Classe Pair

F _first
Representa el primer atribut de la classe Pair.

S _second
Representa el segon atribut de la classe Pair.

public Pair(F first, S second)
Constructora de la classe Pair
@param first representa el primer atribut de la classe Pair.
@param second representa el segon atribut de la classe Pair.

public boolean equals(Object o)
Operació que sobrescriu la funció equals i serveix per comprovar si un pair és igual a un altre pair pel que fa al contingut.
@param o representa el atribut a comparar.

public String toString()
Operació que serveix per passar els objectes de la classe Pair a string.

Classe CtrlDomini

CtrlDirectori _ctrlDirectori
Representa el controlador de directori.

CtrlExpressio _ctrlExpressio
Representa el controlador d'expressió.

public CtrlDomini(CtrlDirectori _ctrlDirectori, CtrlExpressio _ctrlExpressio)
Creadora del controlador de domini

@param _ctrlDirectori representa el controlador de directori.
@param _ctrlExpressió representa el controlador d'expressió.

public List<Pair<String, String>> compararDocuments(String metodeComp, String sorting, Integer k, Integer IdDoc)
Operació explicada en el controlador directori que realitza la funcionalitat de documents semblants. En aquesta crida, però, tenim paràmetres adequats per comunicar-se entre capes, com indicar el metode de comparació i el "sorting" desitjat.
@param metodeComp representa els diferents tipus de comparació que podem realitzar.
@param sorting representa el criteri d'ordenació pel resultat.
@param k representa el nombre de documents semblants que volem obtenir.
@param IdDoc representa el número identificador d'un document.

public List<Pair<String, String>> compararQuery(String metodeComp, String sorting, Integer k, String paraules)
Operació del controlador de directori que fa la funció de documents rellevants amb una query donada, però amb paràmetres adequats per comunicar-se entre capes.
@param metodeComp representa els diferents tipus de comparació que podem realitzar.
@param sorting representa el criteri d'ordenació pel resultat.
@param k representa el nombre de documents rellevants que volem obtenir.
@param paraules representa la query que l'usuari entra com a variable per examinar.

public List<String> llistaAutorsPerPrefix(String pre, String sorting)
Operació que retorna una llista amb els documents que tenen un autor que comença amb el prefix passat com a paràmetre, però amb paràmetres correctes per comunicar-se entre capes.
@param pre representa el prefix pel qual volem realitzar la cerca.
@param sorting representa el criteri d'ordenació pel resultat.

public List<String> llistaTitolsPerAutor(String autor, String sorting)
Operació anàloga a la de controlador directori, però amb paràmetres adequats per a la comunicació entre capes.
@param autor representa l'autor pel qual volem fer la cerca.
@param sorting representa el criteri d'ordenació pel resultat

public ArrayList<Document> selectPerExpressio(Integer idExp)
Operació que serveix per obtenir els document que compleixin l'expressió passada com a paràmetre. El controlador consegueix l'atribut docs de controlador de directori. Un cop té el valor de docs, itera sobre tots els documents del directori

obert i crida a la funció selectPerExpressio de la classe CtrlExpressio per comprovar si compleix o no l'expressió passada com a paràmetre.
@param idExp representa el número identificador de l'expressió la qual es vol comprovar.

A continuació hi han totes les funcions que la capa de domini té per tal de comunicar-se amb persistència, cada un d'aquests mètodes estan explicats de forma més detallada en l'apartat 3:

+exportarDocument(String autor, String titol, String contingut FileType format, String path)

+importarDocument(String path):ArrayList<String>

+carregarEstat(int idDir, Directori& dir)

+guardarEstat(int idDir, HashMap<Integer,HashMap<String,Double>> pesosDocs, queue<int> deletedIds, int idNouDoc, HashMap<int,pair<String,String>> docs)

+eliminarDocument(int idDoc):Bool

+guardarContingutDocument(int idDoc, String contingut):Bool

+carregarContingutDocument(int idDoc):String

+guardarExpressio(int idExp, String string):Bool

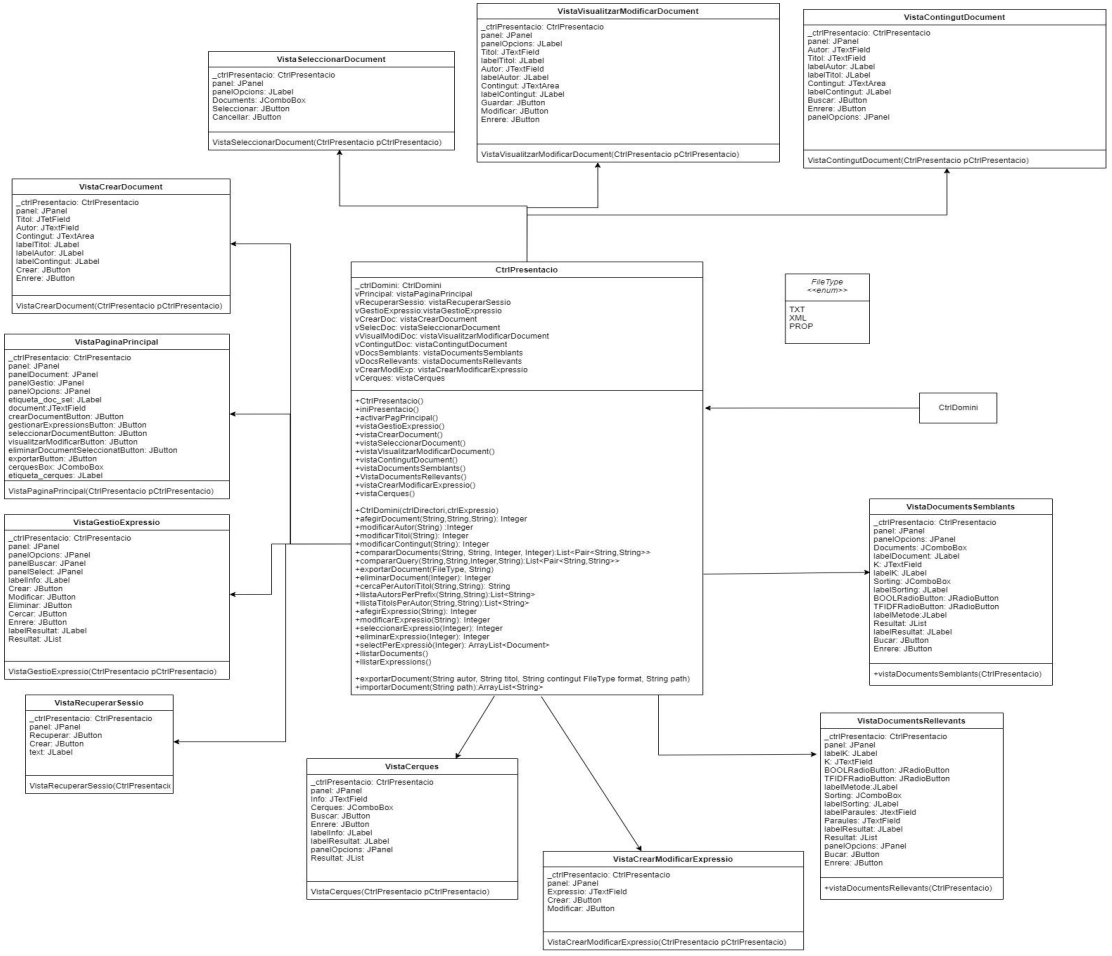
+carregarExpressions(String path):ArrayList<Pair<int, String>>

+eliminarExpressio(idExp):Bool

Els altres mètodes d'aquesta classe no varien de la seva descripció en la classe corresponent.

2. Diagrama de classes de la capa de presentació

2.1 Disseny del diagrama de classes



2.2 Descripció de les vistes

Vista recuperar sessió (vistaRecuperarSessió)
Aquesta vista és l'encarregada de mostrar la primera pantalla que apareix a l'executar el nostre programa i és la vista encarregada de crear el directori sobre el que treballarem, o per contra recuperar-lo.
La vista consta d'un "label" que conté la pregunta sobre quina és la decisió que ha de prendre l'usuari entre crear un nou directori o recuperar la sessió anterior.

Vista de la pàgina principal (vistaPaginaPrincipal)
Aquesta vista és l'encarregada d'oferir les diferents opcions que pot fer l'usuari sobre el directori, com serien afegir una expressió, crear un nou document o importar-ne un de nou, entre d'altres.
La vista consta de nombroses accions per accedir a les altres vistes, entre botons i llistes desplegables. Aquesta serà la vista principal i des d'on realitzarem totes les funcionalitats, o pràcticament totes.
Ens mostra el document seleccionat actualment, i ens permet visualitzar-lo i modificar-lo. També podem eliminar aquest mateix o fins i tot exportar-lo en diversos formats.
Podem seleccionar quines cerques realitzar amb el nostre conjunt de documents amb una llista desplegable.
Comentat anteriorment, podem crear un nou document, canviar de document seleccionat, importar un document des del computador o gestionar les expressions booleanes.

Vista per gestionar una expressió (vistaGestioExpressio)
Aquesta vista permet a l'usuari gestionar tot el que te a veure amb les expressions. Permet a l'usuari afegir, eliminar o modificar una expressió i també permet fer una cerca dels documents que compleixen una certa expressió.
La vista consta de 5 botons per realitzar les diferents opcions que es poden fer sobre una expressió, d'una àrea de text on es mostrarà el resultat i d'un seleccionable que permet seleccionar una de les expressions creades en el directori.

Vista per crear un document (vistaCrearDocument)
Aquesta vista permet a l'usuari crear un document.
La vista consta de camps de text, per tal que l'usuari introdueixi l'autor, el títol i el contingut del document. També hi ha un 2 botons, un que crea el document i un altre que redirigeix a l'usuari a la vista de la pàgina principal.

Vista per seleccionar un document (vistaSeleccionarDocument)
Aquesta vista permet a l'usuari seleccionar un document dels que estan donats d'alta en aquell moment en el sistema.

La vista consta d'un desplegable per seleccionar el document que vulgui el usuari i un botó per tal de confirmar la selecció.

Vista per visualitzar i modificar un document (vistaVisualitzarModificarDocument)
Aquesta vista permet a l'usuari visualitzar o modificar un document seleccionat.
La vista consta de 3 camps de text, un per l'autor, un pel títol i un pel contingut. També hi ha 3 botons, un que permet a l'usuari guardar els canvis fets al document, un que fa editables els 3 camps de text per tal de poder modificar el document i un que porta a l'usuari a la vistaPaginaPrincipal.

Vista del contingut d'un document (vistaContingutDocument)
Aquesta vista permet a l'usuari buscar el contingut d'un document donat el títol i l'autor del document.
La vista consta de 3 camps de text, dos on l'usuari escriu el títol i l'autor del document i un tercer on apareixerà el contingut del document amb l'autor i títol introduÏts per l'usuari. També hi ha 2 botons, un que serveix per realitzar la cerca i que farà aparèixer el contingut al camp de text, i l'altre que redirigeix a l'usuari a la pàgina principal.

Vista per cercar els documents semblants (vistaDocumentsSemblants)
Aquesta vista permet a l'usuari fer una cerca dels k documents més semblants al document escollit per l'usuari.
La vista consta de 2 camps de text un per introduir el nombre de documents semblants que vol obtenir, i un altre on apareixerà el resultat de la cerca.. També conté 2 desplegables, un per seleccionar un document i un altre per escollir com es vol ordenar el resultat. Trobem 2 botons, un per buscar i un per anar la pagina principal. També trobem 2 radio buttons per seleccionar el mètode de comparació.

Vista de documents rellevants donada una query (vistaDocumentsRellevants)
La vista esmentada ens proporciona la funcionalitat de trobar els documents més rellevants donat un conjunt de paraules (query). Tenim components semblants a la vista de documents semblants perque els seu funcionament és pràcticament idèntic.
En aquesta vista tindrem a una casella per inserir les paraules de la query i les mateixes opcions per, escollir el nombre de documents que hi ha d'haver en la solució, el ordre a seguir per la solució i el mètode de comparació utilitzat.
Òbviament compta amb una casella pel resultat.

Vista per crear o modificar una expressió (vistaCrearModificarExpressio)
Aquesta vista permet a l'usuari crear o modificar una expressió.

La vita consta d'un camp de text per introduir o modificar l'expressió i 2 botons, un per crear l'expressió i un per modificar-la.

Vista de cerques per autor o títol (vistaCerques)
La darrera vista ens permet aplicar la funcionalitat de cerca d'autors amb un prefix donat i a més a més, la cerca de títols donat un autor. Serà una vista "mutant" ja que depenent de l'opció escollida en la llista desplegable realitzarem una o l'altra.
Consta de un camp on introduir el prefix, o l'autor, depenent del cas, i un altre camp on es mostra la resposta. Com sempre, tenim els botons que activen la cerca o que ens permet tornar a la vista principal.

2.3 Descripció d'atributs i mètodes de cada classe

VistaRecuperarSessio

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JButton Recuperar:
Botó per recuperar el directori de la sessió anterior.

private JButton Crear:
Botó per crear un nou directori.

private JLabel text:
Text que conté la pregunta.

private JPanel panel:
Panell que conté tots els components de la vista.

public vistaRecuperarSessio(CtrlPresentacio _ctrlPresentacio):
Constructora de vistaRecuperarSessio. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaPaginaPrincipal

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell que conté tots els components de la vista.

private JButton crearDocumentButton:
Botó per crear un document nou.

private JButton importarButton:
Botó per importar un fitxer. Obre el JFileChooser per així triar un fitxer del computador.

private JButton gestionarExpressionsButton:
Botó per accedir a la vista de gestió i ús de les expressions booleanes.

private JTextField document:
Text que conté l'autor i el títol del document seleccionat.

private JLabel etiqueta_doc_sel:
Etiqueta per indicar l'atribut document.

private JButton visualitzarModificarButton:
Botó per accedir a la vista de visualització i modificació de document.

private JButton eliminarDocumentSeleccionatButton:
Botó per eliminar el document seleccionat.

private JButton seleccionarDocumentButton:
Botó per seleccionar un altre document donat d'alta en el sistema, és a dir, per canviar de document seleccionat.

private JButton exportarButton:
Botó per exportar el document seleccionat. Obre el JFileChooser per indicar a quina carpeta exportar-lo del sistema.

private JComboBox cerquesBox:
Llista desplegable amb les diferents opcions de cerca: Cerca per autor i títol, cerca dels títols d'un autor, cerca d'autors per prefix, cerca de documents semblants, cerca d'expressió booleana i cerqua de similitud amb una "query". Acte seguit obre la vista corresponent.

private JLabel etiqueta_cerques:
Etiqueta per indicar el "combo box" amb les diferents cerques

private JPanel panelDocument:
Panell que conté els components relacionats amb la informació del document seleccionat: etiqueta_doc_sel i document.

private JPanel panelGestio:

Panell que conté els components de gestió de programa: crearDocumentButton, importarButton, gestionarExpressionsButton i seleccionarDocumentButton.

private JPanel panelOpcions:
Panell que conté els components per realitzar una acció amb el document seleccionat: visualitzarModificarButton, exportarButton i eliminarDocumentSeleccionatButton.

private JFileChooser file_chooser:
Atribut on instanciarem el component "JFileChooser" per obtenir els "paths" d'importació i exportació.

public vistaPaginaPrincipal(CtrlPresentacio pCtrlPresentacio):
Constructora de la vistaPaginaPrincipal. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaGestioExpressio

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell on s'inclouen tots els elements de la finestra.

private JPanel panelOpcions:
Panell on s'inclouen els botons de crear, modificar o eliminar una expressió.

private JPanel panelBuscar:
Panell on s'inclouen els botons de cercar i enrere.

private JPanel panelSelect:
Panell on s'inclou el desplegable per seleccionar una expressió.

private JLabel labelInfo:
Etiqueta que indica el que ha de fer l'usuari

private JButton Crear:
Botó que redirigeix a l'usuari a la vistaCrearModificarExpressio.

private JButton Modificar:
Botó que redirigeix a l'usuari a la vistaCrearModificarExpressio.

private JButton Eliminar:

Botó que elimina l'expressió seleccionada a la ComboBox.

private JButton Cercar:
Botó que realitza la cerca.

private JButton Enrere:
Botó que porta a l'usuari a la vistaPaginaPrincipal.

private JLabel labelResultat:
Etiqueta per indicar el resultat.

private JList Resultat;
Area de text que mostra el resultat de la cerca.

public VistaGestioExpressio(CtrlPresentacio pCtrlPresentacio):
Constructora de la VistaGestioExpressio. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaCrearDocument

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell on s'inclouen tots els elements de la finestra.

private JTextField Titol:
Camp de text per introduir el títol del document.

private JLabel labelTitol:
Etiqueta per indicar el títol del document.

private JTextField Autor:
Camp de text per introduir l'autor del document.

private JLabel labelAutor:
Etiqueta per indicar l'autor del document.

private JTextArea Contingut:
Camp de text per introduir el contingut del document.

private JLabel labelContingut:
Etiqueta per indicar el contingut del document.

private JButton Crear:
Botó que crea un document i porta a l'usuari a la vistaPaginaPrincipal.

private JButton Enrere:
Botó que porta a l'usuari a la vistaPaginaPrincipal.

public VistaCrearDocument(CtrlPresentacio _ctrlPresentacio):
Constructora de la VistaCrearDocument. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaSeleccionarDocument

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell on s'inclouen tots els elements de la finestra.

private JPanel panelOpcions:
Panell on s'inclouen el botó de seleccionar i cancel·lar.

private JComboBox Documents:
Desplegable que permet a l'usuari seleccionar un document.

private JLabel labelDocuments:
Etiqueta per indicar que fa el desplegable.

private JButton Seleccionar:
Botó que confirma la selecció d'una expressió i porta a l'usuari a la vistaPaginaPrincipal.

private JButton Cancellar:
Botó que cancel·la la selecció d'un document i porta a l'usuari a la vistaPaginaPrincipal.

public VistaSeleccionarDocument(CtrlPresentacio pCtrlPresentacio):
Constructora de la VistaSeleccionarDocument. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaVisualitzarModificarDocument

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell on s'inclouen tots els elements de la finestra.

private JPanel panelOpcions:
Panell on s'inclouen els botons modificar, guardar i enrere.

private JTextField Titol:
Camp de text per introduir el títol del document.

private JLabel labelTitol:
Etiqueta per indicar el títol del document.

private JTextField Autor:
Camp de text per introduir l'autor del document..

private JLabel labelAutor:
Etiqueta per indicar l'autor del document.

private JTextArea Contingut:
Camp de text per introduir el contingut del document.

private JLabel labelContingutl:
Etiqueta per indicar el contingut del document.

private JButton Guardar:
Botó que guarda els canvis fets al document.

private JButton Modificar:
Botó que serveix per fa editables els 3 camps de text i permet modificar el document.

private JButton Enrere:
Botó que porta a l'usuari a la vistaPaginaPrincipal.

public VistaVisualitzarModificarDocument(CtrlPresentacio pCtrlPresentacio):
Constructora de la VistaVisualitzarModificarDocument. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaContingutDocument

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell que conté els components de la vista.

private JTextField Autor;
Text que contindrà el nom de l'autor del document a buscar.

private JTextField Titol:
Text que contindrà el títol del document a buscar.

private JLabel labelAutor:
Etiqueta que indica l'autor.

private JLabel labelTitol:
Etiqueta que indica el títol.

private JTextArea Contingut;
Text on hi ha el resultat de la cerca.

private JLabel labelContingut:
Etiqueta que indica el contingut.

private JButton Buscar:
Botó per realitzar la cerca.

private JButton Enrere:
Botó per tornar a la vista de la pàgina principal.

private JPanel panelOpcions:
Panell que conté els components: Buscar, enrere

public vistaContingutDocument(CtrlPresentacio pCtrlPresentacio)
Constructora de la vistaContingutDocument. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaDocumentsSemblants

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell on s'inclouen tots els elements de la finestra.

private JPanel panelOpcions:
Panell on s'inclouen els botons de buscar i enrere.

private JComboBox Documents:
Desplegable que permet a l'usuari seleccionar un document.

private JLabel labelDocument:
Etiqueta que indica a l'usuari per que serveix el desplegable.

private JTextField k:
Camp de text per introduir el nombre de documents semblants que volem obtenir.

private JLabel labelK:
Etiqueta que indica a l'usuari que introduir al camp de text.

private JComboBox Sorting:
Desplegable que permet a l'usuari seleccionar com vol ordenar el resultat de la cerca.

private JLabel labelSorting:
Etiqueta que indica a l'usuari per que eserveix el desplegable.

private JRadioButton BOOLRadioButton:
Radio Button que permet seleccionar el mètode bool de comparació.

private JRadioButton TF-IDFRadioButton:
Radio Button que permet seleccionar el mètode tf-idf de comparació.

private JLabel labelMetode:
Etiqueta que indica el mètode de comparació.

private JList Resultat:
Area de text on apareixerà el resultat de la cerca.

private JLabel labelResultat:
Etiqueta que indica el resultat de la cerca.

private JButton Buscar:
Botó que mostra el resultat de la cerca.

private JButton Enrere:
Botó que porta a l'usuari a la vistaPaginaPrincipal.

public VistaDocumentsSemblants(CtrlPresentacio pCtrlPresentacio):
Constructora de la VistaDocumentsSemblants. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaDocumentsRellevants

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell que conté les components de la vista.

private JLabel labelK:
Etiqueta que indica a l'usuari què introduir al camp de text "k".

private JTextField k:
Text que indica la quantitat de documents que volem obtenir de la cerca.

private JRadioButton BOOLRadioButton:
RadioButton per indicar que utilitzarem la comparació booleana entre documents.

private JRadioButton TFIDFRadioButton:
RadioButton per indicar que utilitzarem la comparació TFIDF entre documents.

private JComboBox sorting:
Permet triar un tipus d'ordenació.

private JLabel labelSorting:
Etiqueta que indica a l'usuari per que serveix el desplegable.

private JButton Buscar:
Botó per iniciar la cerca.

private JButton Enrere:
Botó per tornar a la vista de pàgina principal.

private JPanel panelOpcions:
Panell que conté les opcions: Buscar i Enrere.

private JLabel labelResultat:
Etiqueta que indica el resultat.

private JLabel labelMetode:
Etiqueta que indica el mètode de comparació.

private JTextField Paraules:
Text on inserirem les paraules de la query.

private JLabel labelParaules

private JList Resultat;
Llista on hi haurà el resultat de la cerca.

public vistaDocumentsRellevants(CtrlPresentacio pCtrlPresentacio)
Constructora de la VistaDocumentsRellevants. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaCrearModificarExpressio

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell que conté les components de la vista.

private JTextField Expressio:
Àrea de text que serveix per introduir una expressió.

private JButton Crear:
Botó que crea una expressió i porta a l'usuari a la vistaPaginaPrincipal.

private JButton Modificar:
Botó que modifica una expressió i porta a l'usuari a la vistaPaginaPrincipal.

public vistaCrearModificarExpressio(CtrlPresentacio pCtrlPresentacio)
Constructora de la VistaCrearModificarExpressio. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l'atribut de la classe.

VistaCerques

private CtrlPresentacio _ctrlPresentacio:
Atribut per referenciar la instància del controlador de presentació.

private JPanel panel:
Panell que conté tots els components de la vista.

private JTextField Info:
Text que serveix per introduir el prefix o l'autor.

private JComboBox Cerques:

“Combo box” per escollir entre les dues cerques possibles.

private JButton Buscar:
Botó per realitzar la cerca.

private JButton Enrere:
Botó per tornar a la vista de pàgina principal.

private JLabel labelInfo:
Etiqueta per indicar el component “info”.

private JLabel labelResultat:
Etiqueta per indicar el component resultat.

private JPanel panelOpcions:
Panell que conté els botons: Buscar i Enrere.

private JList Resultat:
Llista on obtindrem els resultat de la cerca

public vistaCerques(CtrlPresentacio pCtrlPresentacio)
Constructora de vistaCerques. Té com a paràmetre el controlador de presentació amb el qual inicialitzarem l’atribut de la classe.

CtrlPresentació

El controlador de presentació crea les vistes descrites anteriorment i crida a les funcions de domini per tal de poder-les fer servir cada cop que es realitzi una certa acció en la interfície gràfica. Les funcions de domini estan explicades en la secció 1.3 del document.

private static CtrlDomini _ctrlDomini
Instància del controlador de domini

Atributs per la instància de cada una de les vistes:

private vistaPaginaPrincipal vPrincipal;
private vistaRecuperarSessio vRecuperarSessio;
private vistaGestioExpressio vGestioExpressio;
private vistaCrearDocument vCrearDoc;
private vistaSeleccionarDocument vSelecDoc;
private vistaVisualitzarModificarDocument vVisualModiDoc;
private vistaContingutDocument vContingutDoc;
private vistaDocumentsSemblants vDocsSemblants;

private vistaDocumentsRellevants vDocsRellevants;
private vistaCrearModificarExpressio vCrearModiExp;
private vistaCerques vCerques;

public CtrlPresentacio()
Constructora del controlador de presentació i inicialitza el controlador de domini.

public void iniPresentacio()
Creem la instància de la vista de recuperació de sessió (o no) i la sincronitzem amb la vista principal.

public void activarPagPrincipal()
Activa la vista de la pàgina principal, permet interactuar amb ella.

public void vistaGestioExpressio()
Inicialitzem la instància de la vista per gestionar les expressions booleanes al sistema i la sincronitzem amb la vista principal.

public void vistaCrearDocument()
Creem la instància de la vista de creació d'un document i la sincronitzem amb la vista principal.

public void vistaSeleccionarDocument()
Creem la instància de la vista de selecció de documents i la sincronitzem amb la vista principal.

public void vistaVisualitzarModificarDocument()
Creem la instància de la vista per modificar o visualitzar el document seleccionat i la sincronitzem amb la vista principal.

public void vistaContingutDocument()
Creem la instància de la vista per buscar el contingut d'un document i la sincronitzem amb la vista principal.

public void vistaDocumentsSemblants()
Instanciem la vista per realitzar la cerca de documents semblants i la sincronitzem amb la vista principal.

public void vistaDocumentsRellevants()
Instanciem la vista per realitzar la cerca de documents rellevants donat un conjunt de paraules i la sincronitzem amb la vista principal.

public void vistaCrearModificarExpressio()

Creem la instància de la vista per modificar o crear una expressió booleana i la sincronitzem amb la vista principal.

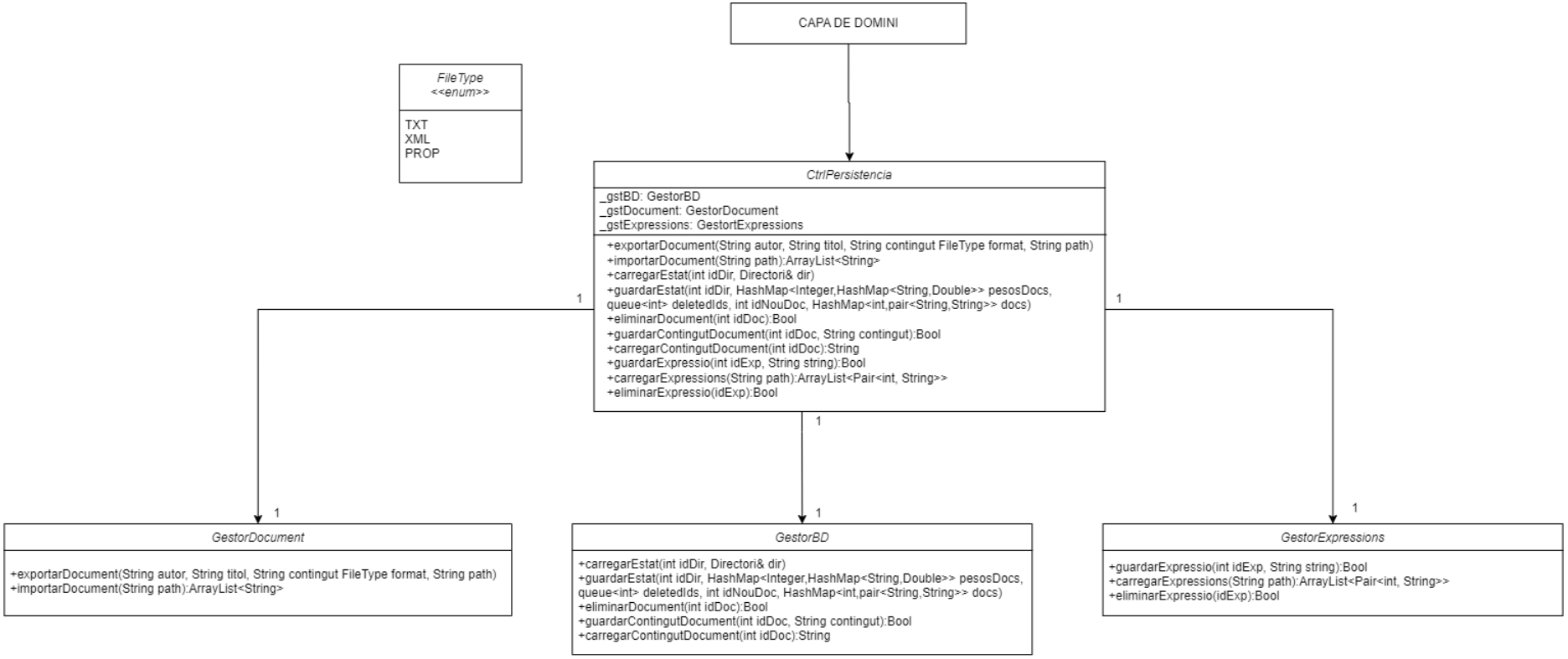
public void vistaCerques()
Creem la instància de la vista per realitzar cerques d'autors per prefix o de títols d'un autor i la sincronitzem amb la vista principal.

Com hem dit, tenim les crides des del controlador de presentació a les funcions del domini mitjançant el controlador domini. Les funcions de domini ja explicades les utilitzem en aquest controlador de la forma:

_ctrlDomini.funcio(param)
(instancia de controlador de domini).funcioDelDomini(paràmetres)

3.Diagrama de classes de la capa de persistència

3.1 Disseny del diagrama de classes



El document de la pàgina anterior representa el nostre diagrama UML pel que fa a la capa de persistència. Com podem observar, tenim un controlador anomenat CtrlPersistencia que s'utilitza per realitzar les comunicacions necessàries amb la capa de domini. A continuació hi ha una explicació amb més detall de que fa cada classe dins el diagrama.

3.2 Descripció de les classes

3.2.1 GestorDocument

Aquesta classe és l'encarregada d'exportar i importar documents entre el nostre programa i el sistema de fitxers.
Les exportacions es fan en el format que tria l'usuari (.txt, .xml o .prop). La capa de domini haurà de passar els camps que formen un document i el gestor de documents crearà un fitxer que representa aquest document, en el format seleccionat, a l'adreça proporcionada per l'usuari.
Per a les importacions es detecta el format del fitxer segons el seu nom (els mateixos formats que per a les exportacions) i es retornen al domini els camps que formen el document que es vol importar.

3.2.1.1 Descripció del format propietari

El nostre format propietari (.prop) es basa amb etiquetes i text pla en una sola línia, de manera semblant a com ho fa xml.

Com sabem, els documents al nostre sistema tenen 3 camps: autor, títol i contingut. Aquestes etiquetes seran representades entre parèntesis i en qualsevol ordre. Seguidament de l'etiqueta hi haurà el camp corresponent, delimitat per unes fletxetes: "->" i "<-".

Un exemple de document en el nostre format propietari és el següent:

> (autor)->Charles Darwin<-(titol)->L'origen de les espècies<-(contingut)->contingut del document<-

Com hem dit, l'ordre de les etiquetes no és rellevant, per tant, el següent fitxer en format propietari representa un document idèntic a l'anterior:

> (titol)->L'origen de les espècies<-(contingut)->contingut del document<-(autor)->Charles Darwin<-

Sabem que el camp de contingut dels documents pot ser buit, aquest quedaria representat de la següent manera: (contingut)-><-

3.2.2 GestorBD

Com que en aquest projecte no tenim un servidor destinat a ser la base de dades, utilitzem una part del disc del propi usuari com a BD improvisada. Aquesta classe té tots els mètodes necessaris per tal de mantenir actualitzada i emmagatzemada les dades del programa i oferir a l'usuari la persistència requerida.

3.2.3 GestorExpressions

Aquesta classe té la funció de gestionar totes les expressions que l'usuari utilitza en el programa. Es pot considerar una part aïllada del GestorBD però d'ús exclusiu per les expressions, s'ha consensuat la partició entre GestorBD i GestorExpressions ja que pensem que les expressions son una part molt gran i important del nostre sistema com per tenir una classe pròpia i diferenciada dels documents i directoris. Té les funcions i mètodes necessaris per poder guardar i mantenir actualitzada la informació de cada expressió per tal que el sistema ofereixi de nou les expressions quan l'execució es reprèn.

3.3 Descripció de atributs i mètodes de cada classe

Les classes de la capa de persistencia, a part del controlador de persistència, no contenen atributs.

3.3.1 GestorDocument

public void exportarDocument(String autor, String titol, String contingut, FILETYPE format, String path)
Operació que serveix per crear un nou fitxer a disc que representi un document. El fitxer es crea amb el nom *autor_titol.format*
@param autor representa l'autor del document que volem exportar.
@param titol representa el títol del document que volem exportar.
@param contingut representa el contingut del document que volem exportar.
@param format representa el format en el qual volem exportar el document.
@param path representa l'adreça al directori on volem exportar el document.

public ArrayList<String> importarDocument(String path)
Operació que serveix per llegir un document de disc i retornar els 3 camps que formen la representació d'un document en forma d'ArrayList<String> (3).
@param path representa l'adreça del fitxer a importar.

private ArrayList<String> parseTXT(String path)

Operació que serveix per llegir un document de disc i, interpretant-lo en format .txt, retornar els 3 camps que formen la representació d'un document en forma d'ArrayList<String> (3).
@param path representa l'adreça del fitxer a importar.

private ArrayList<String> parseXML(String path)
Operació que serveix per llegir un document de disc i, interpretant-lo en format .xml, retornar els 3 camps que formen la representació d'un document en forma d'ArrayList<String> (3).
@param path representa l'adreça del fitxer a importar.

private ArrayList<String> parsePROP(String path)
Operació que serveix per llegir un document de disc i, interpretant-lo en format .prop, retornar els 3 camps que formen la representació d'un document en forma d'ArrayList<String> (3).
@param path representa l'adreça del fitxer a importar.

3.3.2 GestorBD

public carregarEstat(int idDir, Directori& dir)
CarregarEstat permet recuperar l'estat anterior d'un directori quan es va tancar l'aplicació. És a dir, es recuperen de disc la cua d'ids de documents borrades, així com l'identificador pel següent document que es vulgui crear, tots els documents del directori (id, autor i titol) i la taula de pesos dels continguts dels documents.
@param idDir representa la id del directori que es vol recuperar.
@param dir és el paràmetre de sortida on es carrega l'estat

public bool guardarEstat(int idDir, HashMap<Integer,HashMap<String,Double>>pesosDocs, queue<int> deletedIds, int idNouDoc, HashMap<int,pair<String,String>> docs)
GuardarEstat és l'operació que s'executarà únicament quan es tanqui l'aplicació per tal de guardar a disc tota la informació en el seu últim estat. Les variables que es guarden són: la id del Directori, la taula de pesos dins el nostre sistema, la cua d'ids borrades, la id corresponent al seguent document que es vol crear i un HashMap on hi ha la int del document com a clau i un pair d'autor i títol com a valor. D'aquesta manera, podem guardar la última versió dels documents de manera eficient. El contingut del document es guarda en un directori a part, ja que no ens interessa que es carregui a memòria quan es recupera tota la informació.
Retorna cert si l'operació s'ha efectuat correctament.

public bool eliminarDocument(int idDoc)
Eliminar document borra el fitxer corresponen al document amb idDoc.

Retorna cert si l'operació s'ha efectuat correctament.
@param idDoc representa la id del document que es vol eliminar de disc.

public bool guardarContingutDocument(int idDoc, String contingut)
Aquest mètode crea un nou fitxer amb nom idDoc on s'hi escriu el contingut, la comunicació entre autor-títol i contingut es fa mitjançant la seva id ja que tots els continguts es troben en fitxers diferents on el seu nom representa la id del document. D'aquesta manera, com que autor i títol ja estaran carregats a MP, només s'haurà de consultar els continguts necessaris en moments puntuals.
@param idDoc representa la id del document que es vol guardar a disc.
Retorna cert si l'operació s'ha efectuat correctament.
@param contingut representa el contingut del document amb idDoc

public String carregarContingutDocument(int idDoc)
Aquest mètode és l'encarregat de carregar a MP el contingut del document idDoc. El seu sistema busca el fitxer amb nom idDoc i en selecciona el contingut per tal d'enviar-lo a domini.
@param idDoc representa la id del document que es vol eliminar de disc.

3.3.3 GestorExpressions

public bool guardarExpressio(int idExp, String expressio)
Aquest mètode crea un nou fitxer amb nom idExp on s'hi escriu l'expressió corresponent.
Retorna cert si l'operació s'ha efectuat correctament.
@param idExp representa la id de l'expressió que es vol guardar de disc.
@param expressio representa l'expressió amb idExp

public ArrayList<pair<int, String> carregarExpressions(String path)
Aquest mètode llegeix tots els fitxers d'expressions de l'adreça path i els retorna en un ArrayList amb el seu identificador (idExp) i el seu contingut.
@param path representa l'adreça de la carpeta on s'han guardat els fitxers expressions

public bool eliminarExpressio(idExp)
Eliminar Expressió borra el fitxer corresponen a l'expressió amb idExp.
Retorna cert si l'operació s'ha efectuat correctament
@param idExp representa la id de l'expressió que es vol eliminar de disc.

4. Estructura de dades i algorismes utilitzats

4.1 Estructures de dades

4.1.1 ArrayList

Una classe ArrayList és una matriu redimensionable, que està present a Java. Tot i que les matrius integrades tenen una mida fixa, les ArrayLists poden canviar la seva mida de forma dinàmica. Els elements es poden afegir i eliminar d'una ArrayList sempre que calgui, ajudant l'usuari amb la gestió de la memòria.

L'avantatge que té respecte a els vectors és que quan augmenta la dimensió de l'estructura de dades, el vector dobla la seva capacitat mentre que l'arrayList només demana el 50%.

Els temps d'inserció i eliminació és constant. Per trobar un element, com que no està ordenat de cap manera, el temps és **O(n)**.

L'ús de les ArrayList en aquest projecte ha estat per donar resultats i com a utilització per tal de guardar informació en mètodes específics.

4.1.2 Diccionaris

Per tal de representar una taula de pesos dins el nostre sistema hem triat per l'opció dels diccionaris. L'altra opció que hi havia sobre la taula era una matriu de vectors, era una possibilitat molt viable a l'hora de programar, ja que l'espai en memòria que s'usava era mínim, tot i això, es va acabar descartant donat els alts costos temporals que comportaven.

Aquí és on entren els diccionaris, unes estructures de dades que ocupen més espai en memòria, però que ens permeten fer les operacions més ràpidament.

4.1.2.1 HashMap

Un HashMap és una estructura de dades que utilitza una funció hash per assignar valors d'identificació, coneguts com a claus, amb els seus valors associats. Conté parells "clau-valor" i permet recuperar el valor a partir de la seva clau.

El principal avantatge respecte a les matrius és que només es guarda aquella informació que hem afegit explícitament, ergo l'espai ocupat serà **Θ(x)**. L'altre avantatge és que la cerca en un diccionari és de cost lineal **Θ(x).**

En el nostre sistema podem veure sis diccionaris, tres a la classe Directori, dos a Document i un a CtrlExpressió.

Els mapes de la classe Directori es declaren de la següent manera:

- pesosDocuments -> Hashmap< Integer, Hashmap< String, Double>> on la key del primer map es correspon a la ID del document i el valor és un altre map. En el segon map la key és una paraula (la qual trobem en el contingut d'algun document dins el nostre sistema) i el value és el pes de la paraula en aquell document.
- docs -> HashMap<Integer, Document> la key d'aquest map representa la id del document que té com a valor. Ens serveix per a guardar tots els documents que tenim en el directori.
- paraulesDirectori -> HashMap<String, Integer> en aquest últim mapa s'hi guarden les paraules que trobem en el directori, on cada paraula representa una entrada en el mapa i el seu valor és el nombre d'ocurrències en el sistema.

Pel que fa a la classe Document tenim els següents mapes:

- ocurrencies -> HashMap <String, Integer> la clau d'aquest mapa són les paraules que tenim dins el contingut del document i el valor és el nombre de vegades que aquella paraula hi apareix. (és com el mapa de paraulesDirectori, però en aquest cas només hi guardem les paraules d'aquell document en especial)
- tfMap -> HashMap<String,Double> aquest mapa es guarda per un motiu d'eficiència, representa el pes de cada paraula del document dins el nostre sistema (la clau és la paraula i el valor el pes).

Finalment, el mapa que trobem a la classe CtrlExpressió és el següent:

- expressions: HashMap <Integer, Expressio> la key d'aquest mapa representa la id de l'expressió que té com a valor. Representen les expressions que nosaltres tenim en el sistema.

El cost per inserció en el map és constant en cas mitjà i lineal. En cas pitjor **O(n)** i el temps d'eliminació té els mateixos costs que els d'inserció així doncs estem davant una de les millors estructura de dades per utilitzar en aquest projecte.

4.1.3 List

La tercera estructura de dades que utilitzem en el projecte són les llistes. De la mateixa manera que les ArrayList, el seu ús en el codi ha estat per donar resultats i com a utilització per tal de guardar informació en mètodes específics.

Com que es tracta de la interfície de la que neix la classe ArrayList, els seus temps d'inserció, eliminació i cerca són els mateixos.

4.1.4 Cua

Finalment, l'última estructura que utilitzem són les cues. Quan eliminem un document dins el nostre directori la seva id quedarà invalidada i no s'usarà més. Per tal de poder "reciclar" les ids implementem aquesta estructura de dades. Gràcies al concepte de FIFO ens permetrà inserir les primeres ids eliminades i que siguin aquestes les primeres a sortir de la cua quan es crea un nou document. Ens podem fixar que una estructura, com per exemple una pila, també ens podria servir.

El cost temporal d'aquesta estructura per inserció i eliminació és immillorable, en ambdós casos és **O(1)**

4.1.4.1 Cua de prioritat

Aquest és l'única estructura de dades que canviem respecte la primera entrega ja que no vam tenir cap crítica per millorar el codi, tot i així, fem aquest petit canvi per millorar el programa.

Es tracta d'una cua de prioritat, la seva utilitat és que la tindrem ordenada per número de tal manera que quan a la cua hi hagin nombres més petits, seran aquests els primers els que s'usaran a l'hora de crear un nou document.

Perquè s'entengui millor fiquem un exemple:

- Imaginem que tenim una cua amb ids 54,64,75,3. Si tinguéssim una cua normal, el primer element que s'usaria seria el 54 però amb una cua de prioritat s'agafa primer el 3.

4.2 Algorismes

4.2.1 Model booleà

El model booleà és un del models més utilitzats per a la recuperació d'informació. Aquest model està basat en la lògica booleana i la teoria de conjunts. Es tracta d'un model binari ja que considera dues possibilitats en relació a un terme en el document: valor 0(f"alse") si aquest és absent, o valor 1("true") si hi és present. No tenim en compte la freqüència dels termes en els documents.

Algunes de les avantatges d'aquest model serien:

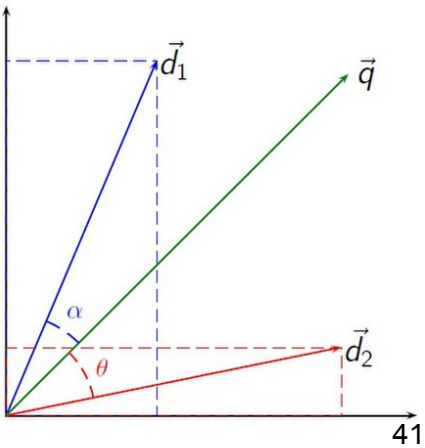
1. És un model formal
2. Fàcil d'implementar
3. Model bastant intuïtiu
4. És fàcil identificar els resultats

Però degut a la seva simplesa, tot i el seu gran ús no és dels models més eficaços i presenta diverses desavantatges :

1. Degut a la cerca pot retornar massa o massa pocs documents
2. Tots els termes presents pesen el mateix
3. Com a conseqüència de l'anterior punt, no podem oferir un ordre entre documents que compleixin la cerca

4.2.2 Model vectorial

El model d'espai vectorial és un model algebraic per a representar documents de text com a vectors d'identificadors. Cada document té el seu vector amb el seu pes, si apareix un terme (paraula) en el document el seu valor en el vector és el nombre d'ocurrències en aquell document. S'han desenvolupat diverses formes diferents de calcular aquests valors, també conegudes com a ponderacions (més endavant explicarem la ponderació tf-idf utilitzada en aquest projecte).



La semblança de documents en una cerca de paraules clau es poden calcular, utilitzant els supòsits de la teoria de similituds de documents, comparant la desviació d'angles entre cada vector de document i el vector de consulta original

podem obtenir la similitud entre documents. En la imatge de la dreta podem veure un exemple visual per entendre-ho més clarament, d1 i d2 fan referència als documents avaluats i q representa la consulta. Els angles α i θ representen la semblança.
El model d'espai vectorial té els següents avantatges sobre el model booleà estàndard:

1. Model simple basat en àlgebra lineal
2. Ponderacions de terme no binàries
3. Permet calcular un grau continu de similitud entre consultes i documents.
4. Permet classificar els documents segons la possible rellevància
5. Permet la coincidència parcial

El model d'espai vectorial té les limitacions següents:

1. Els documents llargs estan mal representats perquè tenen valors de similitud deficients (un producte escalar petit i una dimensionalitat gran)
2. Les paraules clau de cerca han de coincidir exactament amb els termes del document; les subcadenes de paraules poden donar com a resultat una "coincidència falsa positiva"
3. Sensibilitat semàntica; els documents amb un context similar, però un vocabulari de termes diferent no s'associaran, cosa que resultarà en una "coincidència falsa negativa".
4. L'ordre en què apareixen els termes al document es perd en la representació de l'espai vectorial.
5. Se suposa teòricament que els termes són estadísticament independents.
6. La ponderació és intuïtiva però no gaire formal.

4.2.3 TF-IDF

En la recuperació d'informació, tf-idf, abreviatura de freqüència de termini-freqüència inversa de documents, és una estadística numèrica que pretén reflectir la importància d'una paraula per a un document. El valor tf-idf augmenta proporcionalment a la quantitat de vegades que una paraula apareix al document i es compensa amb la quantitat de documents al directori que contenen la paraula, la qual cosa ajuda a ajustar el fet que algunes paraules apareixen amb més freqüència en general. Dit amb altres paraules, el pes de cada paraula en el document segueix la fórmula:

$$tfidf(t, d, D) = tf(t, d) * idf(t, D)$$

on

tf(t,d) = en nombre d'ocurrències de t dins el document d
idf(t,D) = el pes de la paraula dins el directori (explicat a continuació)

Seguint la definició anterior, paraules com "el" o "la", que són articles i s'utilitzen molts cops en els documents, tindran un pes inferior que els substantius.

Per saber el pes de la paraula en el directori s'usa la freqüència inversa del document. La freqüència inversa del document és una mesura de quanta informació proporciona la paraula, és a dir, si és comú o rara a tots els documents. És la fracció inversa escalada logarítmicament dels documents que contenen la paraula (obtinguda dividint el nombre total de documents pel nombre de documents que contenen el terme, i després prenent el logaritme d'aquest quocient):

$$idf(t, D) = log(\frac{N}{|d \in D : t \in d|})$$

on
N: nombre total de documents al directori N = {|D|}
$|d \in D : t \in d|$: nombre de documents d on el terme t apareix dins el directori D.

4.2.3 Informació d'entrega

Com que en el feedback rebut de la primera entrega no especifica cap canvi a fer per la part de millora d'algorismes i estructures de dades no s'ha tocat res en aquestes seccions (a part del canvi de cua a cua de prioritat).

Per altra banda, els canvis rebuts en el feedback (millora de llegibilitat del codi, millora dels comentaris i partició dels test per tal de no ser tan llargs) seran aplicats per tal de realitzar la tercera entrega.