UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

GESTOR DE DOCUMENTOS

PROP 22-23 Q1

Autores:

Rubén Dabrio Ramírez

Lorenzo Ricci Pujol

Haobin Guo

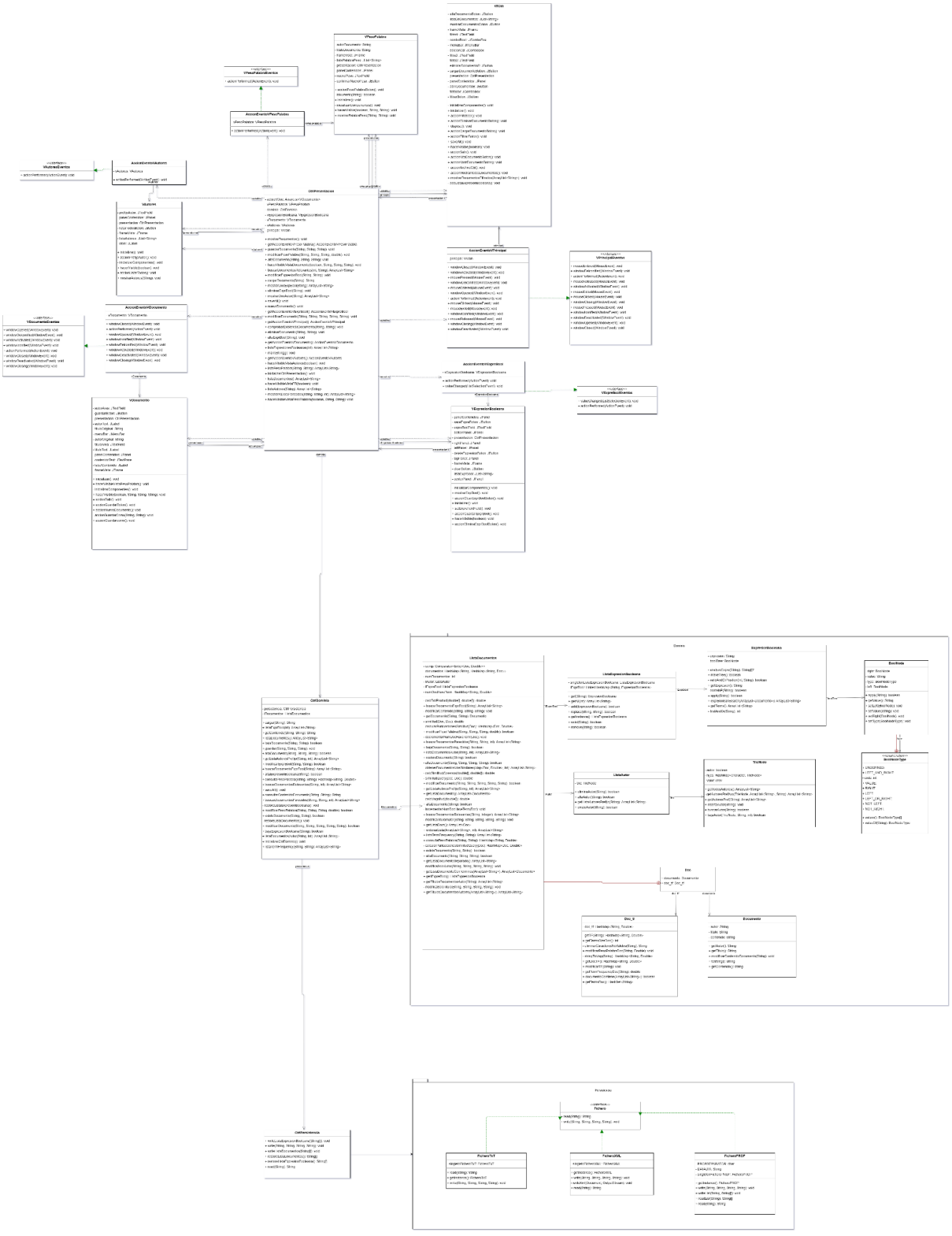
Sergi Campuzano Cabot

Tutor:

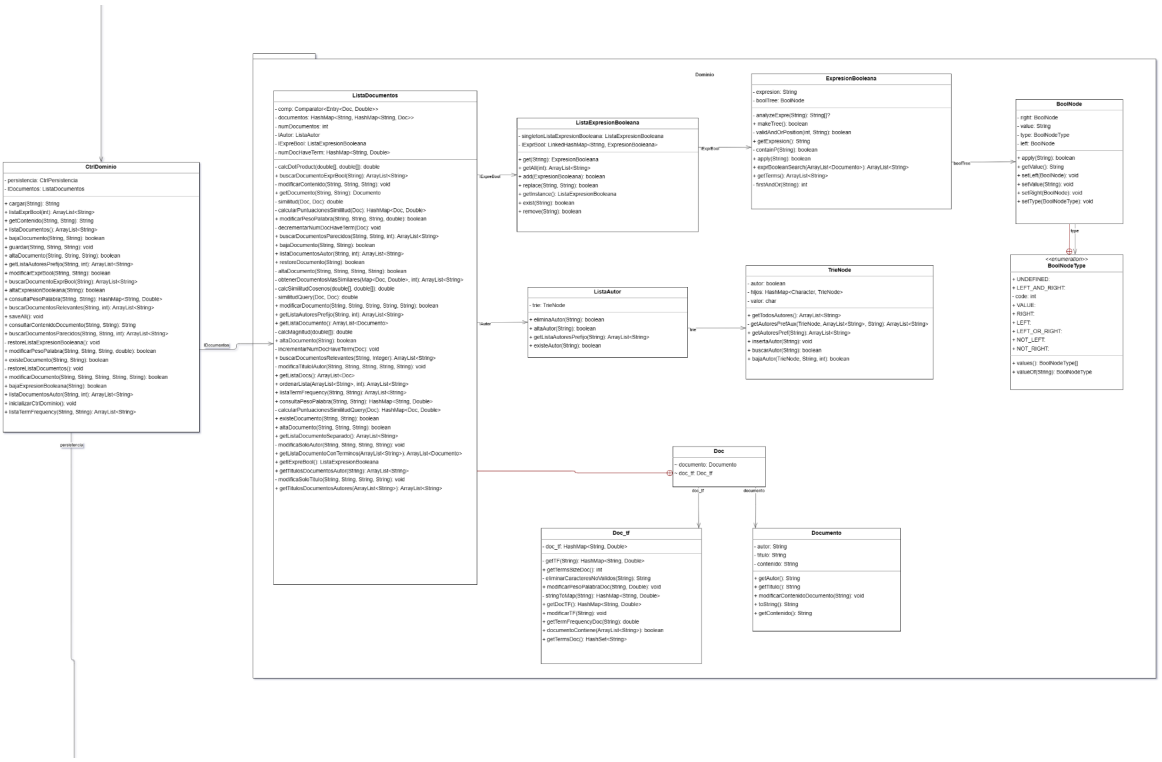
Sergio Álvarez

Index

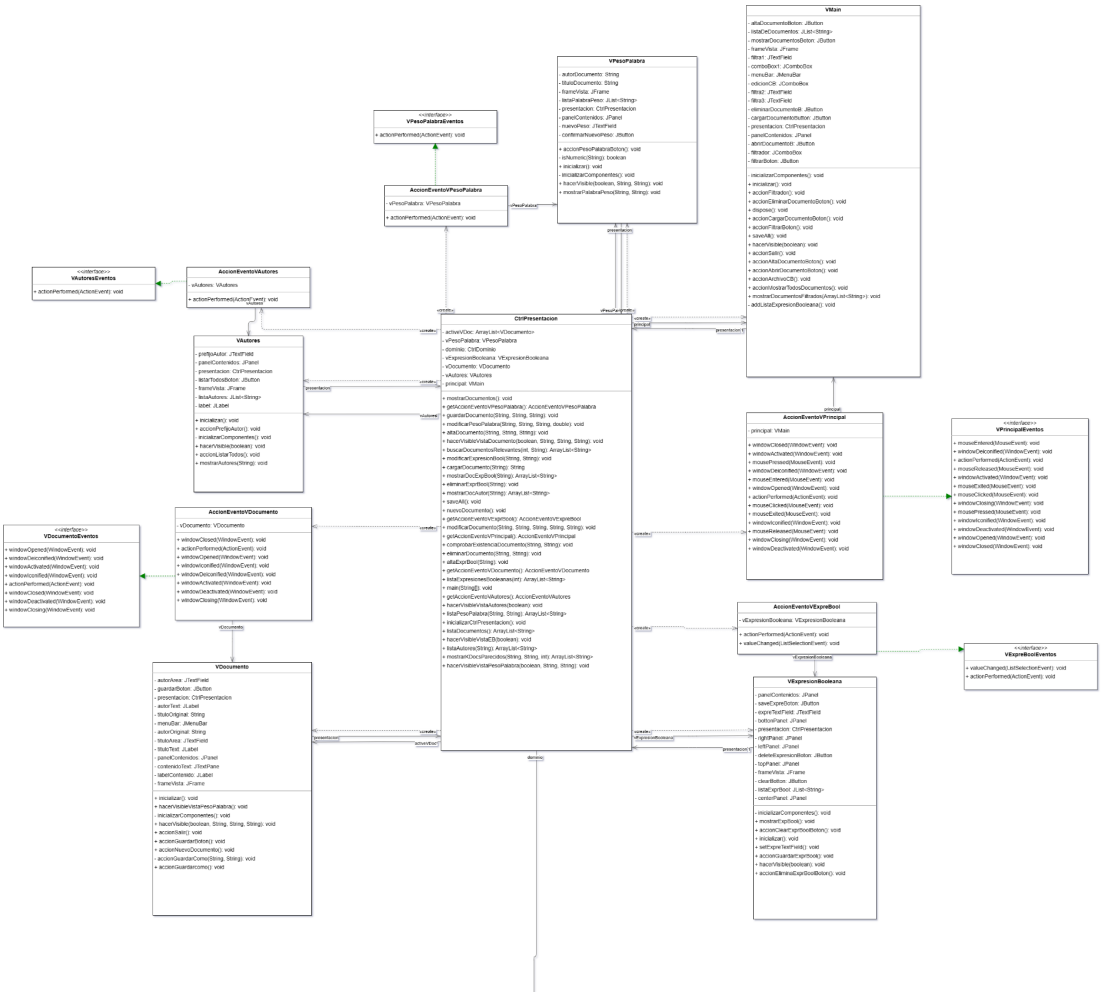
1. Diagramas y controladores



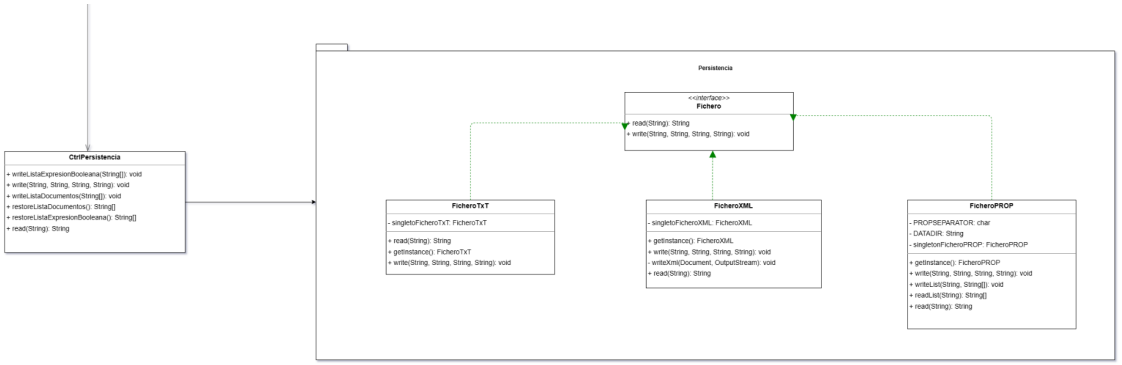
1.1.Dominio


Dominio
CtrlDominio
- persistencia: CtrlPersistencia
- tDocumentos: ListaDocumentos
+ cargar(String): String
+ listaExprBool(int): ArrayList<String>
+ getContenido(String, String): String
+ listaDocumentos(): ArrayList<String>
+ bajaDocumento(String, String): boolean
+ guardar(String, String, String): void
+ altaDocumento(String, String, String): boolean
+ getListaAutores(Prefijo(String, int): ArrayList<String>
+ modificarExprBool(String, String): boolean
+ buscarDocumentosExprBool(String): ArrayList<String>
+ altaExpresionBooleana(String): boolean
+ consultaPesoPalabra(String, String): HashMap<String, Double>
+ buscarDocumentosRelevantes(String, int): ArrayList<String>
+ saveAll(): void
+ consultarContenidoDocumento(String, String): String
+ buscarDocumentosFrancotiro(String, String, int): ArrayList<String>
- restoreListaExpresionBooleana(): void
+ modificarPesoPalabra(String, String, String, double): boolean
+ existeDocumento(String, String): boolean
- restoreListaDocumentos(): void
+ modificarDocumento(String, String, String, String, String): boolean
+ bajaExpresionBooleana(String): boolean
+ listaDocumentosAutor(String, int): ArrayList<String>
+ reiniciarCtrlDominio(): void
+ listaTermFrequency(String, String): ArrayList<String>
persistencia
tDocumentos
ListaDocumentos
- comp: Comparator<Entry<Doc, Double>>
- documentos: HashMap<String, HashMap<String, Doc>>
- numDocumentos: int
- tAutor: ListaAutor
- tExpreBool: ListaExpresionBooleana
- numDocsHaveTerm: HashMap<String, Double>
- calcDotProduct(double[], double[]): double
+ buscarDocumentosExprBool(String): ArrayList<String>
- modificarContenido(String, String, String): void
+ getDocumento(String, String): Documento
- similitud(Doc, Doc): double
- calcularPuntuacionesSimilitud(Doc): HashMap<Doc, Double>
+ modificarPesoPalabra(String, String, String, double): boolean
- decrementarNumDocsHaveTerm(Doc): void
+ buscarDocumentosFrancotiro(String, String, int): ArrayList<String>
+ bajaDocumento(String, String): boolean
+ listaDocumentosAutor(String, int): ArrayList<String>
+ restoreDocumentos(String): boolean
- altaDocumento(String, String, String, String): boolean
- obtenerDocumentosMasSimilares(Map<Doc, Double>, int): ArrayList<String>
- calcSimilitudCosenos(double[], double[]): double
- similitudQuery(Doc, Doc): double
+ modificarDocumento(String, String, String, String, String): boolean
+ getListaAutoresPrefijo(String, int): ArrayList<String>
+ getListaDocumento(): ArrayList<Documento>
- calcMagnitud(double[]): double
+ altaDocumento(String): boolean
- incrementarNumDocsHaveTerm(Doc): void
+ buscarDocumentosRelevantes(String, Integer): ArrayList<String>
- modificarTituloAutor(String, String, String, String, String): void
+ getListaDocs(): ArrayList<Doc>
+ ordenarListaArrayList<String>, int): ArrayList<String>
+ listaTermFrequency(String, String): ArrayList<String>
+ consultaPesoPalabra(String, String): HashMap<String, Double>
- calcularPuntuacionesSimilitudQuery(Doc): HashMap<Doc, Double>
+ existeDocumento(String, String): boolean
+ altaDocumento(String, String, String): boolean
+ getListaDocumentosSeparados(): ArrayList<String>
- modificarAutor(String, String, String, String): void
+ getListaDocumentosConTerminos(ArrayList<String>): ArrayList<Documento>
+ getExpresBool(): ListaExpresionBooleana
+ getTitulosDocumentosAutor(String): ArrayList<String>
- modificaSoloTitulo(String, String, String, String): void
+ getTitulosDocumentosAutores(ArrayList<String>): ArrayList<String>
ExpreBool
ListaExpresionBooleana
- singletonListaExpresionBooleana: ListaExpresionBooleana
- tExpr: LinkedHashMap<String, ExpresionBooleana>
+ get(String): ExpresionBooleana
+ getAll(int): ArrayList<String>
+ add(ExpresionBooleana): boolean
+ replace(String, String): boolean
- getInstance(): ListaExpresionBooleana
+ exist(String): boolean
+ remove(String): boolean
theRoot
ExpresionBooleana
- expresion: String
- boolTree: BoolNode
- analyzeType(String): String[]?
+ makeTree(): boolean
- validAndOrPosition(int, String): boolean
+ getExpresion(): String
- containT(String): boolean
+ apply(String): boolean
+ exprBooleanSearch(ArrayList<Documento>): ArrayList<String>
+ getTerms(): ArrayList<String>
- firstAndOr(String): int
theTree
BoolNode
- right: BoolNode
- value: String
- type: BoolNodeType
- left: BoolNode
+ apply(String): boolean
+ getValue(): String
+ setLeft(BoolNode): void
+ setValue(String): void
+ setRight(BoolNode): void
+ setType(BoolNodeType): void
type
<<enumeration>>
BoolNodeType
+ UNDEFINED
+ LEFT_AND_RIGHT:
- code: int
+ VALUE:
+ RIGHT:
+ LEFT:
+ LEFT_OR_RIGHT:
+ NOT_LEFT:
+ NOT_RIGHT:
+ values(): BoolNodeType[]
+ valueOf(String): BoolNodeType
ListaAutor
- tree: TrieNode
+ eliminaAutor(String): boolean
+ altaAutor(String): boolean
+ getListaAutores(Prefix(String): ArrayList<String>
+ existeAutor(String): boolean
tree
TrieNode
- autor: boolean
- hijos: HashMap<Character, TrieNode>
- valor: char
+ getTodosAutores(): ArrayList<String>
- getAutores(TrieNode, ArrayList<String>, String): ArrayList<String>
+ getAutores(Prefix(String): ArrayList<String>
+ insertaAutor(String): void
+ buscarAutor(String): boolean
+ bajaAutor(TrieNode, String, int): boolean
Doc
- documento: Documento
- doc_tf: Doc_tf
doc_tf
documento
Doc_tf
- doc_tf: HashMap<String, Double>
- getTF(String): HashMap<String, Double>
+ getTermsSizeDoc(): int
- eliminarCaracteresNoValidos(String): String
+ modificarPesoPalabraDoc(String, Double): void
- stringToMap(String): HashMap<String, Double>
+ getDocTF(): HashMap<String, Double>
+ modificarTF(String): void
+ getTermFrequency(Doc(String): double
+ documentoContiene(ArrayList<String>): boolean
+ getTermsDoc(): HashSet<String>
Documento
- autor: String
- titulo: String
- contenido: String
+ getAutor(): String
+ getTitulo(): String
+ modificarContenidoDocumento(String): void
+ toString(): String
+ getContenido(): String

1.2.Presentación



1.3.Persistencia



2. Descripción clases

2.1. Dominio

2.1.1. CtrlDominio

Controlador de la parte de **Dominio** que se encarga de enlazar a los demás controladores que constituyen la arquitectura con las funciones del dominio, llamando a funciones de *ListaDocumentos*, *ListaAutor* y *ListaExpresionBooleana*.

Atributos:

- private CtrlPersistencia persistencia
    - Controlador de persistencia del sistema
- private ListaDocumentos lDocumentos
    - Lista de documentos del sistema

Metodos:

- public CtrlDominio()
    - Constructor por defecto
- public void inicializarCtrlDominio()
    - Inicializa el controlador de dominio, restableciendo la información de la sesión anterior.
- public void guardar(String path, String titulo, String autor)
    - Guarda un documento en el path especificado.
- public String cargar(String path)
    - Carga el documento especificado en el path .
- public void saveAll()
    - Guarda toda la información del conjunto de documentos y expresiones booleanas en un directorio por defecto del disco.
- private void restoreListaExpresionBooleana()
    - Restablece la información del conjunto de expresiones booleanas de la sesión anterior.
- private void restoreListaDocumentos()
    - Restablece la información del conjunto de documentos de la sesión anterior.
- public boolean altaDocumento(String titulo, String autor, String contenido)

    - Añade un documento al conjunto de documentos
- public boolean bajaDocumento(String titulo, String autor)
    - Elimina un documento del conjunto de documentos
- public boolean modificarDocumento(String tituloID, String autorID, String titulo, String autor, String contenido)
    - Modifica el contenido, autor, título… de un documento.
- public HashMap<String,Double> consultaPesoPalabra(String titulo, String autor)
    - Consulta el peso de las palabras de un documento
- public boolean modificarPesoPalabra(String titulo, String autor, String palabra, double peso)
    - Modifica el peso de una palabra de un documento
- public boolean altaExpresionBooleana(String expr)
    - Añade una expresión booleana al conjunto de las mismas
- public boolean bajaExpresionBooleana(String expr)
    - Elimina una expresión booleana del conjunto de las mismas
- public ArrayList<String> listaExprBool(int orden)
    - Obtiene la lista de expresiones booleanas, en orden especificado
- public boolean modificarExprBool(String oldExpr, String newExpr)
    - Modifica la expresion booleana oldExpr por newExpr
- public ArrayList<String> buscarDocumentoExprBool(String expr)
    - Obtiene los títulos de los documentos que cumplen una expresión
- public ArrayList<String>listaDocumentosAutor(String autor, int code)
    - Obtiene los títulos de los documentos de un autor ordenados de varias formas
- public ArrayList<String> getListaAutoresPrefijo(String prefijo, int code)
    - Obtiene los nombres de los autores que contienen cierto prefijo ordenados de varias formas
- public String getContenido(String titulo, String autor)
    - Obtiene el contenido del documento.
- public ArrayList<String> buscarDocumentosParecidos(String titulo, String autor, int k)
    - Obtiene los títulos de los documentos que más se parecen al indicado
- public ArrayList<String> buscarDocumentosRelevantes(String query, int k)
    - Obtiene los títulos de los documentos que son más relevantes
- public ArrayList<String> listaDocumentos()
    - Obtiene los títulos de todos los documentos.

- public ArrayList<String> listaTermFrequency(String titulo, String autor)
    - Obtiene la lista de frecuencia de términos de un documento.
- public boolean existeDocumento(String titulo, String autor)
    - Devuelve true si el documento existe.

2.1.2. Documento

Objeto documentos y gestión de contenido del mismo

Atributos:

- String Autor
    - Representa el autor del documento.
- String Título
    - Representa el título del documento.
- String Contenido
    - Representa el contenido del documento

. **Metodos:**

- public Documento()
    - Constructor por defecto, vacío.
- public Documento(String Titulo, String Autor, String Contenido)
    - Constructor por parametrós.
- public String getAutor()
    - Devuelve el autor del documento.
- public String getTiítulo()
    - Devuelve el título del documento
- public String getContenido()
    - Devuelve el contenido del documento.
- public void modificarContenidoDocumento(String nContenido)
    - Dado un nuevo contenido cambia el contenido antiguo por el especificado.
- public String toString()
    - Devuelve un String con el título y autor del documento.

2.1.3. Doc_tf

Atributos:

- HashMap<String,Double> doc_tf
    - Mapa de frecuencia de términos en el documento, la key es el término y el value es la frecuencia de este mismo dentro del documento (número de ocurrencias de la palabra en el documento / número de palabras totales del documento)

. **Metodos:**

- public Doc_tf (Documento doc)
    - Constructor de objeto Doc_tf dado el documento al que hace referencia.
- public Doc_tf (String termFreq)
    - Constructor de objeto Doc_tf a partir de String.
- public HashMap<String,Double> getDocTF()
    - Devuelve el mapa de frecuencia de términos del documento.
- public HashSet<String> getTermsDoc()
    - Devuelve un HashSet de String que contiene todos los términos del documento.
- public int getTermsSizeDoc()
    - Devuelve el número de términos en el documento
- public double getTermFrequencyDoc(String term)
    - Devuelve la frecuencia del término "term" si está en el documento, si no devuelve 0.0.
- public void modificarTF(String nContenido)
    - Dado un nuevo contenido recalcula el mapa de frecuencia de términos.
- public void modificarPesoPalabraDoc(String palabra, Double new_value)
    - Dados el término y el nuevo peso, cambia el peso del término.
- public boolean documentoContiene(ArrayList<String> terms)
    - Devuelve true si el documento contiene algún término del ArrayList.
- private HashMap<String, Double> stringToMap(String termFrequency)
    - Método privado para pasar de un string termFrequency al mapa de frecuencia de términos.
- private HashMap<String,Double> getTF(String contenido)
    - Método privado para obtener el mapa de frecuencia de términos del documento tratando su contenido.

- private static String eliminarCaracteresNoValidos(String s)
    - Método privado para eliminar caracteres no válidos para así poder tratar bien el contenido.

2.1.4. ListaDocumentos

Estructura para guardar y gestionar un conjunto de documentos.

Atributos:

- private class Doc
    - Clase privada que representa un documento y su frecuencia de términos.
- HashMap<String, HashMap<String, Doc>> documentos
    - Representa el conjunto de documentos.
- ListaAutor lAutor
    - Representa la lista de autores del sistema
- ListaExpresionBooleana lExpreBool
    - Representa la lista de expresiones booleanas del sistema
- int numDocumentos
    - Contador de todos los documentos
- HashMap<String, Double> numDocHaveTerm
    - Conjunto palabras con el número de documentos que contiene la palabra

Metodos:

- public ListaDocumentos()
    - Constructor por defecto.
- public ListaExpresionBooleana getlExpreBool()
    - Devuelve la lista de expresiones booleanas.
- public ArrayList<String> getListaDocumentoSeparado()
    - Devuelve la lista de documentos con los documentos en string con sus partes por separado, siendo la primera línea el título, la segunda el autor, el tercero la frecuencia de términos, y el resto el contenido.
- public boolean altaDocumento(String documento)
    - Añade el documento especificado al conjunto de documentos, devuelve true si la alta se ha llevado a cabo, el formato del

documento es, la primera línea el título, segunda el autor, y el resto el contenido.

- public boolean restoreDocumento(String documento)
    - Restablece un documento a partir del documento, devuelve true si la alta se ha llevado a cabo, el formato del documento es, la primera línea el título, la segunda el autor, el tercero la frecuencia de términos, y el resto el contenido.
- private boolean altaDocumento(String titulo, String autor, String termFrequency, String cont)
    - Método privado para dar de alta un documento a partir de sus partes como Strings separados, con termFrequency, devuelve true si la alta se ha llevado a cabo.
- public boolean altaDocumento(String titulo, String autor, String cont)
    - Añade un documento al conjunto de documentos devuelve true si la operación se ha efectuado correctamente y falso si ya existe el documento.
- public boolean bajaDocumento(String titulo, String autor)
    - Elimina el documento especificado por título y autor, del conjunto de documentos. Devuelve false si falta modificar el autor, sino retorna true.
- private void decrementarNumDocHaveTerm(Doc d)
    - Decrementa los pesos de las palabras del contenido del documento registradas en numDocHaveTerm.
- private void incrementarNumDocHaveTerm(Doc d)
    - Incrementa los pesos de las palabras del contenido de documento en numDocHaveTerm.
- public boolean modificarDocumento(String tituloID, String autorID, String titulo, String autor, String contenido)
    - Modifica un documento. Devuelve falso si queda por modificar aún el autor de lista autores, sino devuelve true
- private void modificarContenido(String tituloID, String autorID, String contenido)
    - Modifica el contenido del documento identificado con tituloID y autorID.
- private void modificaTituloIAutor(String tituloID, String autorID, String titulo, String autor, String contenido)
    - Modifica el título y el autor del documento

- private void modificaSoloTitulo(String tituloID, String autorID, String titulo, String contenido)
    - Modifica solo el título del documento
- private void modificaSoloAutor(String tituloID, String autorID, String autor, String contenido)
    - Modifica solo el autor del documento
- public Documento getDocumento(String titulo, String autor)
    - Si existe un documento con titulo y autor especificados, devuelve el documento. Sino devuelve null.
- public ArrayList<String> ordenarLista(ArrayList<String> docs, int code)
    - Devuelve ordenada una lista de documentos en formato String según el code.
- public ArrayList<String> getTitulosDocumentosAutor(String autor)
    - Consulta la lista de los documentos del autor, que existe.
- public ArrayList<String> getTitulosDocumentosAutores (ArrayList<String> autores)
    - Consulta la lista de los documentos de un conjunto de autores
- public ArrayList<String> listaDocumentosAutor(String prefijo, int code)
    - Consulta la lista de los documentos de los autores que tengan el prefijo ‘prefijo’ y se ordena segun el ‘code’
- public ArrayList<String> getListaAutoresPrefijo(String prefijo, int code)
    - Consulta la lista de los autores que tengan el prefijo ‘prefijo’ y se ordena segun el ‘code’
- public ArrayList<Documento> getListaDocumento()
    - Obtiene la lista de todos los documentos de ListaDocumentos.
- public ArrayList<Doc> getListaDocs()
    - Obtiene la lista de Doc de ListaDocumentos.
- public ArrayList<Documento> getListaDocumentoConTerminos (ArrayList<String> terms)
    - Obtiene la lista de todos los documentos de ListaDocumentos que contengan algún término de la lista.
- public ArrayList<String> buscarDocumentosParecidos(String titulo, String autor, int k)
    - Devuelve una lista de los "k" documentos más parecidos al documento "d".
- private HashMap<Doc,Double> calcularPuntuacionesSimilitud(Doc d)

    - Devuelve un HashMap de la puntuación de similitud del Doc d con los demás documentos de ListaDocumentos.
- private double similitud(Doc d1, Doc d2)
    - Retorna el grado de similitud entre el documento modelo y otro documento.
- public ArrayList<String> buscarDocumentosRelevantes(String query, Integer k)
    - Devuelve el conjunto de los "k" documentos más relevantes para la "query".
- private HashMap<Doc,Double> calcularPuntuacionesSimilitudQuery(Doc query)
    - Devuelve un HashMap de la puntuación de similitud de la query con los documentos de ListaDocumentos.
- private double similitudQuery(Doc queryD, Doc toCompareD)
    - Retorna el grado de similitud entre el Doc query y otro Doc.
- private double calcSimilitudCosenos(double [] vectorA, double [] vectorB)
    - Devuelve el cálculo de la similitud de Cosenos entre el vectorA y el vectorB.
- private double calcDotProduct(double[] vectorA, double[] vectorB)
    - Devuelve el cálculo del producto escalar de los dos vectores.
- private double calcMagnitud(double[] vector)
    - Devuelve el cálculo de la magnitud del vector.
- private final Comparator<Map.Entry<Doc, Double>> comp = (o1, o2) ->
    - Comparador privado usado para ordenar los documentos relevantes y parecidos.
- private ArrayList<String> obtenerDocumentosMasSimilares(Map<Doc, Double> puntuacionesSimilitud, int k)
    - Devuelve una lista de documentos más similares.
- public ArrayList<String> listaTermFrequency(String titulo, String autor)
    - Devuelve una lista con todos los terminos del documento y su peso
- public boolean existeDocumento(String titulo, String autor)
    - Devuelve true si el documento existe en ListaDocumentos.
- public ArrayList<String> buscarDocumentoExprBool(String expr)
    - Busca el conjunto de documentos que tengan una frase que cumpla la expresin booleana, mantiene la lista de expresion booleana ordenada por uso.

2.1.5. ListaAutor

Estructura para guardar y gestionar un conjunto de autores, representados por su nombre.

Atributos:

- TrieNode trie
    - Es un árbol de prefijos cuyo nodo raíz contiene los demas

Métodos:

- public ListaAutor()
    - Constructor por defecto
- public boolean altaAutor(String autor)
    - Devuelve true si la operación se efectúa correctamente, false en cualquier otro caso. Lanza una excepción si el autor ya existe.
- public boolean existeAutor(String autor)
    - Comprueba si existe el autor pasado como parámetro. Devuelve true si existe y false en caso contrario
- public boolean eliminaAutor(String autor)
    - Devuelve true si la operación se efectúa correctamente, false en cualquier otro caso. Lanza una excepcion si el autor no existe
- public ArrayList<String> getListaAutoresPrefijo(String pref)
    - Devuelve la lista de los nombres de los autores que tienen como prefijo pref.

2.1.6. TrieNode

Estructura que representa un nodo de un árbol de prefijos capaz de expandirse como árbol.

Atributos:

- boolean autor
    - Indica si el nodo actual representa el final de un nombre de autor o no
- Hashmap<Character, TrieNode> hijos
    - Conjunto de los nodos hijo del nodo actual
- char valor
    - Valor del nodo actual

Métodos:

- public TrieNode(char valor)
    - Constructor por defecto que asigna un caracter al atributo valor
- public TrieNode()
    - Constructor por defecto que asigna un caracter vacio al atributo valor
- public void insertaAutor(String autor)
    - Inserta un autor que no existe en el arbol
- public boolean buscarAutor(String autor)
    - Comprueba si existe un autor en el arbol
- public boolean bajaAutor(TrieNode nodo, String autor, int indice)
    - Elimina un autor que existe del arbol
- public ArrayList<String> getAutoresPref(String pref)
    - Devuelve la lista de los autores que empiezan por un prefijo
- private ArrayList<String> getAutoresPrefAux(TrieNode nodo, ArrayList<String> autores, String nombre)
    - Funcion auxiliar recursiva que devuelve la lista de los autores que empiezan por un prefijo
- public ArrayList<String> getTodosAutores()
    - Devuelve la lista de todos los autores

2.1.7. ListaExpresionBooleana

Estructura para guardar y gestionar un conjunto de ExpresionBooleana. Se aplica el patrón singleton, ya que solo es necesario una lista para guardar todas las expresiones booleanas del aplicativo.

Atributos:

- ListaExpresionBooleana singletonListaExpresionBooleana
    - Representa la única instancia del conjunto de expresiones booleanas.
- LinkedHashMap <String,ExpresionBooleana> lExprBool
    - Representa el conjunto de expresiones booleanas.

Metodos:

- private ListaExpresionBooleana()
    - Constructor por defecto
- public static ListaExpresionBooleana getInstance()
    - Consultora de la instancia del conjunto de expresiones booleanas.
- public ExpresionBooleana get(String expr)

    - Buscar una expresión booleana “expr” en el conjunto de expresiones booleanas.
- public ArrayList<String> getAll(int orden)
    - Retorna el conjunto de expresion booleana como String en orden especificado
- public boolean exist(String expr)
    - Comprueba la existencia de la expresión booleana “expr” en el conjunto de expresiones booleanas.
- public boolean add(ExpresionBooleana exprBool)
    - Añade una expresión booleana “exprBool” al conjunto de expresiones booleanas.
- public boolean remove(String expr)
    - Elimina la expresión booleana “expr” del conjunto de expresiones booleanas.
- public boolean replace(String oldExpre, String newExpre)
    - Reemplaza una expresión booleana “oldExpre” por otra expresión boolean “newExpre”.

2.1.8. ExpresionBooleana

Estructura que guarda una expresión booleana.

Atributos:

- String expresion
    - Representa una expresión booleana en formato texto.
- BoolNode boolTree
    - Representa una expresión booleana en forma de BoolNode.

Metodos:

- public ExpresionBooleana()
    - Constructor por defecto.
- public ExpresionBooleana(String expr)
    - Constructor por parámetro, con inicialización de expresión.
- public String getExpresion()
    - Retorna la expresion actual
- public ArrayList<String> getTerms()
    - Retorna la lista de términos que aparece en la expresión booleana.
- public boolean makeTree()

    - Crear la estructura del árbol para el posterior análisis.
- public boolean apply(String frase)
    - Comprueba que la frase cumpla la expresión booleana.
- public ArrayList<String> exprBooleanSearch(ArrayList<Documento> docs)
    - Busca el conjunto de documentos que tengan una frase que cumpla la expresión booleana sobre el conjunto de documentos “docs”.
- private String[] analyzeExpre(String)
    - Analiza la expresión, y devuelve un conjunto de partes de la expresión valido, null si la expresión no es válida.
- private static int firstAndOr(String)
    - Busca la posición del primer operador lógico a comprobar.
- private static boolean validAndOrPosition(int,String)
    - Valida la posición del operador lógico.
- private static boolean containP(String)
    - Busca (), {}, ! o espacio en la expresión, ignorado los que están entre comillas ".

2.1.9. BoolNode

Estructura para representar expresiones booleanas, expresiones que tengan operadores lógicos (ej, &, |, !).
La estructura se basa en la idea de *Binary Tree*, tiene un valor, un tipo, un nodo izquierdo y un nodo derecho.
El valor puede ser un operador lógico aplicado a un nodo o dos nodos, o un valor, dependiendo del tipo.

Atributos:

- String value
    - Valor del BoolNode
- BoolNodeType type
    - Tipo de nodo: UNDEFINED, VALUE, LEFT_OR_RIGHT, LEFT_AND_RIGHT, NOT_LEFT, NOT_RIGHT, LEFT, RIGHT.
- BoolNode left
    - Nodo izquierdo del BoolNode
- BoolNode right
    - Nodo derecho del BoolNode

Metodos:

- public BoolNode()
    - Constructor por defecto, con inicialización del valor a vacío, tipo a UNDEFINED, nodos a null
- public BoolNode(String)
    - Constructor por parámetro , con inicialización del valor a expresión, tipo a UNDEFINED, nodos a null
- public String getValue()
    - Retorna el valor del nodo actual
- public void setValue(String)
    - Modifica el valor del nodo actual
- public void setLeft(BoolNode)
    - Modifica el nodo izquierdo
- public void setRight(BoolNode)
    - Modifica el nodo derecho
- public void setType(int type)
    - Modifica el tipo de nodo
- public boolean apply(String)
    - Comprueba que la frase cumpla la expresión booleana partiendo de este nodo.

2.2.Presentación

2.2.1. CtrlPresentacion

Controlador de la parte de **Presentación** que se encarga de enlazar los eventos que recibe de cada vista de la aplicación con las funciones a aplicar por el controlador de dominio.

Atributos:

- CtrlDominio dominio
    - Instancia del controlador de dominio para enlazar controladores
- VMain principal
    - Instancia de la vista VPrincipal
- VExpresionBooleana vExpresionBooleana
    - Instancia de la vista VExpresionBooleana
- VAutores vAutores
    - Instancia de la vista VAutores
- VDocumento vDocumento

    - Instancia de la vista VDocumento
- VPesoPalabra vPesoPalabra
    - Instancia de la vista VPesoPalabra
- ArrayList<VDocumento> activeVDoc
    - Lista de todas las vistas VDocumento que se han creado al crear un nuevo documento

Metodos:

- public CtrlPresentacion()
    - Crea todas las vistas y tiene una instancia de controlador de dominio para acceder a dominio
- public void inicializarCtrlPresentacion()
    - Inicializa todas las vistas y a controlador de dominio
- public AccionEventoVPrincipal getAccionEventoVPrincipal()
    - Detecta un evento de VPrincipal y dirige la acción a tomar
- public AccionEventoVDocumento getAccionEventoVDocumento()
    - Detecta un evento de VDocumento y dirige la acción a tomar
- public AccionEventoVAutores getAccionEventoVAutores()
    - Detecta un evento de VAutores y dirige la acción a tomar
- public AccionEventoVExpreBool getAccionEventoVExprBool()
    - Detecta un evento de VExpresionBooleana y dirige la acción a tomar
- public AccionEventoVPesoPalabra getAccionEventoVPesoPalabra()
    - Detecta un evento de VPesoPalabra y dirige la acción a tomar
- public void hacerVisibleVistaEB(boolean b)
    - Hace visible o no la vista VExpresionBooleana
- public void hacerVisibleVistaAutores(boolean b)
    - Hace visible o no la vista VAutores
- public void hacerVisibleVistaDocumento(boolean b, titulo, autor, contenido)
    - Hace visible o no la vista VDocumento
- public void guardarDocumento(path, titulo, autor)
    - Guarda el documento en el disco
- public void hacerVisibleVistaPesoPalabra(boolean b, titulo, autor)
    - Hace visible o no la vista VPesoPalabra
- public ArrayList<String> listaDocumentos()
    - Devuelve la lista de **titulo + " \0" +autor** de todos los documentos
- public ArrayList<String> listaExpresionesBooleanas(int ord)
    - Devuelve la lista de todas las expresiones booleanas
- public ArrayList<String> listaAutores(prefijo)
    - Devuelve la lista de los nombres de todos los autores
- public ArrayList<String> listaPesoPalabra(titulo, autor)
    - Devuelve la lista de las palabras de un documento junto a su peso
- public ArrayList<String> mostrarKDocsParecidos(titulo, autor, int k)
    - Devuelve la lista de **titulo + " \0" +autor** de los k documentos más parecidos a un documento
- public ArrayList<String> buscarDocumentosRelevantes(int k, query)
    - Devuelve la lista de **titulo + " \0" +autor** de los k documentos más relevantes en una query de palabras
- public ArrayList<String> mostrarDocExpBool(expr)

    - Devuelve la lista de **titulo + " \0" +autor** de los documentos que cumplan con la expresion booleana expr
- public ArrayList<String> mostrarDocAutor(autor)
    - Devuelve la lista de **titulo + " \0" +autor** de los documentos que tengan como autor al parámetro
- public void modificarDocumento(tituloO, autorO, titulo, autor, contenido)
    - Modifica el documento titulo0 autorO con los parametros titulo, autor y contenido
- public void saveAll()
    - Guarda todas las listas al disco
- public String cargarDocumento(path)
    - Carga un documento y lo da de alta
- public void eliminarDocumento(titulo, autor)
    - Elimina el documento identificado con los parámetros
- public void mostrarDocumentos()
    - Dirige la acción de los documentos a filtrar dependiendo del tipo de filtrado
- public void altaDocumento(titulo, autor, contenido)
    - Crea un documento con los parámetros
- public void eliminarExprBool(expr)
    - Elimina la expresion booleana expr
- public void altaExprBool(expr)
    - Crea la expresion booleana expr
- public void modificarExpresionBool(oldExpr, newExpr)
    - Modifica la expresion booleana oldExpr por newExpr
- public void modificarPesoPalabra(titulo, autor, palabra, peso)
    - Modifica el peso (dado como parámetro) de la palabra(dada como parámetro) del documento identificado con titulo y autor
- public void comprobarExistenciaDocumento(titulo, autor)
    - Comprueba si existe el documento identificado con titulo y autor
- public void nuevoDocumento()
    - Abre la vista VDocumento para crear un nuevo documento
- public static void main(String[] args)
    - Función main que inicializa CtrlPresentacion y detecta errores

2.2.2. VMain

Vista principal donde se listan todos los documentos y desde la cual se pueden acceder a las demás vistas. Contiene diferentes opciones para poder filtrar los documentos listados. Estos documentos se pueden abrir y modificar o eliminar, también se pueden cargar o crear nuevos.

Atributos:

- CtrlPresentacion presentacion
    - Instancia del controlador de presentacion para enlazar operaciones
- JFrame frameVista
    - Frame de la vista actual

- JPanel panelContenidos
    - Panel donde esta el contenido de la vista
- JComboBox edicionCB
    - Desplegable para poder cambiar de vista
- JButton altaDocumentoBoton
    - Boton usado para dar de alta un documento
- JList<String> listaDeDocumentos
    - Lista donde aparecen los documentos
- JButton abrirDocumentoB
    - Boton usado para abrir un documento
- JButton eliminarDocumentoB
    - Boton usado para eliminar un documento
- JComboBox filtrador
    - Desplegable para poder cambiar de tipo de filtrado
- JButton cargarDocumentoButton
    - Boton usado para cargar un documento
- JTextField filtra1
    - Area de texto para filtrar documentos
- JTextField filtra2
    - Area de texto para filtrar documentos
- JTextField filtra3
    - Area de texto para filtrar documentos
- JButton filtrarBoton
    - Boton usado para filtrar documentos
- JButton mostrarDocumentosBoton
    - Boton usado para mostrar todos los documentos
- JComboBox comboBox1
    - Área de texto para filtrar documentos por expresión booleana, permite al usuario ver la lista de expresión booleana guardados.
- private JMenuBar menuBar
    - Barra de menú de la vistaPrincipal

Metodos:

- public void inicializar()
    - Inicializa las características de la vista
- private void inicializarComponentes()
    - Inicializa los componentes de la vista
- public void dispose()
    - Cierra la vista
- public void saveAll()
    - Guarda todas las listas al disco
- public void hacerVisible(boolean b)
    - Hace visible o no la propia vista segun b
- public void accionArchivoCB()
    - Procede a cambiar de vista
- public void accionEliminarDocumentoBoton()
    - Elimina un documento
- public void accionAltaDocumentoBoton()

    - Da de alta un documento
- public void accionCargarDocumentoBoton()
    - Carga un documento
- public void accionSalir()
    - Sale de la vista y de la aplicacion preguntando antes si queremos salir
- public void accionAbrirDocumentoBoton()
    - Abre un documento
- public void accionMostrarTodosDocumentos()
    - Muestra todos los documentos
- public void accionFiltrador()
    - Procede a cambiar el tipo de filtrado
- private void addListaExpresionBooleana()
    - Añade al comboBox1 la lista expresión booleana, ordenada por uso, es decir la primera expresión es la ultima expresion usada
- public void accionFiltrarBoton()
    - Filtra los documentos segun el tipo de filtrado que haya
- public void mostrarDocumentosFiltrados(ArrayList<String>documentos)
    - Muestra los documentos que recibe como parámetro

2.2.3. VDocumento

Vista que permite la visualización de un documento existente o a crear y la edición de su título, autor y contenido mediante áreas de texto y un botón de guardado. Además existe un desplegable en el que podemos guardar el documento como un .txt o un .xml.

Atributos:

- String autorOriginal
    - Guarda el autor anterior a la última modificación de autor, el original si no ha habido modificación o está vacío en caso de ser un nuevo documento.
- String tituloOriginal
    - Guarda el título anterior a la última modificación de título, el original si no ha habido modificación o está vacío en caso de ser un nuevo documento.
- CtrlPresentacion presentacion
    - Instancia del controlador de presentacion para enlazar operaciones
- JFrame frameVista
    - Frame de la vista actual
- JPanel panelContenidos
    - Panel donde esta el contenido de la vista
- JButton guardarBoton
    - Boton usado para guardar un documento
- JTextPane contenidoText
    - Area de texto del contenido del documento
- JLabel tituloText
    - Etiqueta con valor fijo ‘título’

- JTextField tituloArea
    - Area de texto del titulo del documento
- JLabel autorText
    - Etiqueta con valor fijo ‘autor’
- JTextField autorArea
    - Area de texto del autor del documento
- JLabel labelContenido
    - Etiqueta con valor fijo ‘contenido’
- JMenuBar menuBar

Metodos:

- public void inicializar()
    - Inicializa las características de la vista
- public void inicializarComponentes()
    - Inicializa los componentes de la vista
- public void hacerVisible(estado, titulo, autor, contenido)
    - Hace visible o no la propia vista segun el estado y actualiza el valor de titulo, autor y contenido para poder operar sobre el documento
- public void accionGuardarBoton()
    - Guarda los cambios del documento
- private void accionGuardarComo(titulo, autor)
    - Funcion auxiliar que abre un cuadro de opciones donde podremos escoger el nombre, el formato y la ubicacion del guardado de nuestro documento
- public void accionSalir()
    - Sale de la vista preguntando antes si queremos guardar el documento
- public void accionNuevoDocumento()
    - Abre un documento vacio para darlo de alta
- public void accionGuardarcomo()
    - Abre un cuadro de opciones donde podremos escoger el nombre, el formato y la ubicacion del guardado de nuestro documento
- public void hacerVisibleVistaPesoPalabra()
    - Hace visible la vista de peso palabra

2.2.4. VExpresionBooleana

Vista que muestra listadas todas las expresiones booleanas y que te permite hacer altas, bajas y modificaciones de expresiones booleanas.

Atributos:

- CtrlPresentacion presentacion;
    - Instancia del controlador de presentacion para enlazar operaciones
- JFrame frameVista
    - Frame de la vista actual
- JPanel panelContenidos;
    - Panel donde esta el contenido de la vista

- JButton saveExpreBoton
    - Boton para guardar una expresion booleana
- JList<String> listaExprBool
    - Lista de expresiones guardadas
- JTextField expreTextField
    - Area de texto para una expresion booleana
- JButton deleteExpresionBoton
    - Boton para eliminar una expresión booleana
- JPanel centerPanel
    - Panel contenedor de elementos principales de la vista
- JPanel rightPanel
    - Panel contenedor de spacer para delimitar
- JPanel leftPanel
    - Panel contenedor de spacer para delimitar
- JPanel topPanel
    - Panel contenedor de spacer para delimitar
- JPanel bottomPanel
    - Panel contenedor de spacer para delimitar
- JButton clearBotton
    - Boton para borrar el contenido de la área de texto “expreTextField”

Metodos:

- public void inicializar()
    - Inicializa las características de la vista
- private void inicializarComponentes()
    - Inicializa los componentes de la vista
- public void setExpreTextField()
    - Copia la expresion seleccionada en la área de texto “expreTextField”
- public void accionClearExprBoolBoton()
    - Borra el contenido de la área de texto “expreTextField” y deja la lista listaExprBool si elementos seleccionados.
- public void hacerVisible(boolean b)
    - Hace visible o no la propia vista según b
- public void accionGuardarExprBool()
    - Guarda la expresion booleana que hay en el área de texto
- public void accionEliminaExprBoolBoton()
    - Borra la expresion booleana seleccionada en listaExprBool
- public void mostrarExpBool()
    - Muestra todas las expresiones booleanas

2.2.5. VAutores

Vista que muestra listados los autores que tienen algún documento. Inicialmente se listan todos los autores pero también hay la opción de que se muestren solo los autores que empiezen por un prefijo determinado.

Atributos:

- CtrlPresentacion presentacion;
    - Instancia del controlador de presentacion para enlazar operaciones
- JFrame frameVista
    - Frame de la vista actual
- JPanel panelContenidos;
    - Panel donde esta el contenido de la vista
- JList<String> listaAutores
    - Lista donde aparecen los autores
- JTextField prefijoAutor
    - Área de texto para filtrar autores por prefijo
- JLabel label
    - Etiqueta con el contenido “prefijo autor”
- JButton listarTodosBoton
    - Botón para que se listen todos los autores

Metodos:

- public void inicializar()
    - Inicializa las características de la vista
- private void inicializarComponentes()
    - Inicializa los componentes de la vista
- public void hacerVisible(boolean b)
    - Hace visible o no la propia vista según b
- public void accionPrefijoAutor()
    - Lista los autores con el prefijo seleccionado en JTextField
- public void accionListarTodos()
    - Lista todos los autores
- public void mostrarAutores(prefijo)
    - Muestra todos los autores que empiezan por el prefijo dado como parámetro

2.2.6. VPesoPalabra

Vista que muestra listadas todas las palabras que aparecen en el contenido de un documento junto a su peso respecto a este documento. El peso de cada palabra se puede modificar en esta vista.

Atributos:

- CtrlPresentacion presentacion
    - Instancia del controlador de presentación para enlazar operaciones
- JFrame frameVista
    - Frame de la vista actual
- JPanel panelContenidos
    - Panel donde esta el contenido de la vista
- JList<String> listaPalabraPeso
    - Lista donde aparecen los palabras junto a su pesoç
- JTextField nuevoPeso

    - Valor del nuevo peso a una palabra
- JButton confirmarNuevoPeso
    - Botón de confirmación de un nuevo peso a una palabra
- String tituloDocumento
    - Guarda el título del documento
- String autorDocumento
    - Guarda el autor del documento

Metodos:

- public void inicializar()
    - Inicializa las características de la vista
- private void inicializarComponentes()
    - Inicializa los componentes de la vista
- public void hacerVisible(boolean b, titulo, autor)
    - Hace visible o no la propia vista según b
- public void mostrarPalabraPeso(titulo, autor)
    - Muestra todos las palabras junto a su peso del documento identificado por título y autor
- public void accionPesoPalabraBoton()
    - Guarda los cambios del peso de la palabra del documento
- private static boolean isNumeric(String peso)
    - Retorna si el parámetro string es numérico o no

2.2.7. VPrincipalEventos

Interfaz hija de ActionListener, WindowListener y MouseListener que permite la gestión de estos tres tipos de eventos en la vista principal (VMain).

Metodos:

- public void actionPerformed(evento)
- public void mouseClicked(evento)
- public void mousePressed(evento)
- public void mouseReleased(evento)
- public void mouseEntered(evento)
- public void mouseExited(evento)
- public void windowOpened(evento)
- public void windowClosing(evento)
- public void windowClosed(evento)
- public void windowIconified(evento)
- public void windowDeiconified(evento)
- public void windowActivated(evento)
- public void windowDeactivated(evento)

2.2.8. VDocumentoEventos

Interfaz hija de ActionListener y WindowListener que permite la gestión de estos dos tipos de eventos en la vista de documento (VDocumento).

Metodos:

- public void actionPerformed(evento)
- public void windowOpened(evento)
- public void windowClosing(evento)
- public void windowClosed(evento)
- public void windowIconified(evento)
- public void windowDeiconified(evento)
- public void windowActivated(evento)
- public void windowDeactivated(evento)

2.2.9. VAutoresEventos

Interfaz hija de ActionListener que permite la gestión de este tipo de eventos en la vista de autores (VAutores).

Metodos:

- public void actionPerformed(evento)

2.2.10. VExpreBoolEventos

Interfaz hija de ActionListener y ListSelectionListener que permite la gestión de estos dos tipos de eventos en la vista de expresiones booleanas (VExpresionBooleana).

Metodos:

- public void actionPerformed(evento)
- public void valueChanged(evento)

2.2.11. VPesoPalabraEventos

Interfaz hija de ActionListener que permite la gestión de este tipo de eventos en la vista de peso palabra (VPesoPalabra).

Metodos:

- public void actionPerformed(evento)

2.2.12. AccionEventoVPrincipal

Implementa la interfaz VPrincipalEventos y representa la accion de los eventos sobre la vista principal (VMain).

Atributos:

- VMain principal
    - Vista principal sobre la cual se hacen las acciones de los eventos recibidos

Metodos:

- public AccionEventoVPrincipal(vista)
    - Constructora por defecto que asigna la vista como atributo
- public void actionPerformed(evento)
- public void mouseClicked(evento)
- public void mousePressed(evento)
- public void mouseReleased(evento)
- public void mouseEntered(evento)
- public void mouseExited(evento)
- public void windowOpened(evento)
- public void windowClosing(evento)
- public void windowClosed(evento)
- public void windowIconified(evento)
- public void windowDeiconified(evento)
- public void windowActivated(evento)
- public void windowDeactivated(evento)

2.2.13. AccionEventoVDocumento

Implementa la interfaz VDocumentoEventos y representa la accion de los eventos sobre la vista de documento (VDocumento).

Atributos:

- VDocumento vDocumento
    - Vista documento sobre la cual se hacen las acciones de los eventos recibidos

Metodos:

- public AccionEventoVDocumento(vista)
    - Constructora por defecto que asigna la vista como atributo
- public void actionPerformed(evento)
- public void windowOpened(evento)
- public void windowClosing(evento)
- public void windowClosed(evento)
- public void windowIconified(evento)
- public void windowDeiconified(evento)

- public void windowActivated(evento)
- public void windowDeactivated(evento)

2.2.14. AccionEventoVAutores

Implementa la interfaz VAutoresEventos y representa la accion de los eventos sobre la vista de autores (VAutores).

Atributos:

- VAutores vAutores
  - Vista de autores sobre la cual se hacen las acciones de los eventos recibidos

Metodos:

- public AccionEventoVAutores(vista)
  - Constructora por defecto que asigna la vista como atributo
- public void actionPerformed(evento)

2.2.15. AccionEventoVExpreBool

Implementa la interfaz VExpreBoolEventos y representa la accion de los eventos sobre la vista expresion booleana (VExpresionBooleana).

Atributos:

- VExpresionBooleana vExpresionBooleana
  - Vista de expresion booleana sobre la cual se hacen las acciones de los eventos recibidos

Metodos:

- public AccionEventoVExpreBool(vista)
  - Constructora por defecto que asigna la vista como atributo
- public void actionPerformed(evento)
- public void valueChanged(evento)

2.2.16. AccionEventoVPesoPalabra

Implementa la interfaz VPesoPalabraEventos y representa la accion de los eventos sobre la vista peso palabra (VPesoPalabra).

Atributos:

- VPesoPalabra vPesoPalabra

- Vista de peso palabra sobre la cual se hacen las acciones de los eventos recibidos

Metodos:

- public AccionEventoVPesoPalabra(vista)
  - Constructora por defecto que asigna la vista como atributo
- public void actionPerformed(evento)

2.3.Persistencia

La capa de persistencia se encarga de todo lo relacionado con la comunicación del programa y el disco.

2.3.1. CtrlPersistencia

Controlador de la parte de **Persistencia** que se encarga de enlazar los demás controladores que constituyen la arquitectura, con las funciones de la persistencia.

Metodos:

- public String read(String path)
  - Lee el fichero que está en el “path” especificado.
- public String write(String path,String titulo, String autor,String contenido)
  - Escribe el título, autor, contenido en el fichero especificado en el path.
- public String[] restoreListaExpresionBooleana()
  - Obtiene una lista de expresión booleana en formato String, del directorio por defecto de los ficheros propietarios.
- public String[] restoreListaDocumentos()
  - Obtiene una lista de documentos, del directorio por defecto de los ficheros propietarios.
- public void writeListaExpresionBooleana(String[] list)
  - Escribe la lista de expresiones booleanas “list”, en un fichero propietario del directorio por defecto de los ficheros propietarios.
- public void writeListaDocumentos(String[] list)
  - Escribe la lista de documentos “list”, en un fichero propietario del directorio por defecto de los ficheros propietarios.

2.3.2. Fichero

Es una interfaz para aplicar el patrón “Strategy” en la capa de persistencia, que representa un fichero en un sistema operativo.

Metodos:

- public String read(String path)
  - Lee el fichero que está en el “path” especificado.
- public void write(String path, String titulo, String autor, String contenido)
  - Escribe el título, autor, contenido en el fichero especificado en el path.

2.3.3. FicheroProp

Es una clase que implementa la interfaz Fichero. Representa un fichero propietario con extensión .prop. Se usa el patrón singleton ya que no se necesita más de una instancia de esta clase.

Atributos:

- private final String DATADIR
  - Dirección del directorio por defecto de los archivos propietarios.
- private final char PROPSEPARATOR
  - Signo para separar diferentes elementos de una lista en formato propietario.
- private static FicheroPROP singletonFicheroPROP;
  - Representa la única instancia de un fichero propietario.

Metodos:

- private FicheroPROP()
  - Constructor por defecto.
- public static FicheroPROP getInstance()
  - Consultora de la instancia del fichero propietario.
- public String read(String path)
  - Lee el fichero que está en el “path” especificado.
- public void write(String path, String titulo, String autor, String contenido)
  - Escribe el título, autor, contenido en el fichero especificado en el path.
- public String [] readList(String name)
  - Lee el fichero propietario especificado “name”, del directorio por defecto.
- public void writeList(String name, String[] list)
  - Escribe el array de String en fichero especificado, en el directorio por defecto.

2.3.4. FicheroTxT

Es una clase que implementa la interfaz Fichero. Representa un fichero de texto plano con extensión .txt. Se usa el patrón singleton ya que no se necesita más de una instancia de esta clase.

Atributos:

- private static FicheroTxT singletonFicheroTxt;
    - Representa la única instancia de un fichero txt.

Metodos:

- private FicheroTxT()
    - Constructor por defecto.
- public static FicheroTxT getInstance()
    - Consultora de la instancia del fichero txt.
- public String read(String path)
    - Lee el fichero que está en el “path” especificado.
- public void write(String path, String titulo, String autor, String contenido)
    - Escribe el título, autor, contenido en el fichero especificado en el path.

2.3.5. FicheroXML

Es una clase que implementa la interfaz Fichero. Representa un fichero XML (Extensible Markup Language) con extensión .xml. Se usa el patrón singleton ya que no se necesita más de una instancia de esta clase.

Atributos:

- private static FicheroXML singletonFicheroXMLt;
    - Representa la única instancia de un fichero xml.

Metodos:

- private FicheroXML()
    - Constructor por defecto.
- public static FicheroXML getInstance()
    - Consultora de la instancia del fichero txt.
- public String read(String path)
    - Lee el fichero que está en el “path” especificado.
- public void write(String path, String titulo, String autor, String contenido)
    - Escribe el título, autor, contenido en el fichero especificado en el path.
- private static void writeXml(org.w3c.dom.Document doc, OutputStream output)
    - Escribe el documento “doc” en el fichero especificado.

3. Estructura de datos y algoritmos

3.1. Documento

Un Documento está compuesto por su autor, título y contenido. En la clase solo hay constructores, getters, setters y un método toString para convertir Documento en String.

3.2. Doc_tf

En Doc_tf tenemos como solo atributo un HashMap<String,Double> doc_tf que representa el mapa de frecuencia de términos. Para obtener el mapa de frecuencia de términos usamos el método getTF. La función getTF toma como entrada una cadena de texto contenido y devuelve un HashMap que asigna a cada palabra de contenido su frecuencia en el texto. La frecuencia de una palabra se calcula dividiendo el número de veces que aparece esa palabra en el texto entre el número total de palabras en el texto. La función comienza eliminando los caracteres no válidos del contenido utilizando la función eliminarCaracteresNoValidos. Luego, se divide el contenido en un array de palabras utilizando el patrón de expresión regular "\\s+", que coincide con cualquier secuencia de uno o más espacios en blanco. El HashMap FreqAux se inicializa con un nuevo objeto vacío y se recorre el array de palabras. Si la palabra actual no está en el HashMap, se añade al HashMap y se le asigna la frecuencia 1/número de palabras. Si la palabra ya está en el HashMap, se actualiza su frecuencia multiplicando su frecuencia anterior por el número de palabras y sumando 1, y luego dividiendo el resultado por el número de palabras.

3.3. ListaDocumentos

En *listaDocumentos* se usa un HashMap<String, HashMap<String, Documento>>, guardando como key a cada uno de los autores que ha hecho algún documento y como valor a un HashMap<String, Doc> que guarda todos los nombres de título junto a Doc que es una clase privada de listaDocumentos que representa el documento (Documento) y su frecuencia de términos (Doc_tf). Se ha usado esta estructura para tener guardado por cada autor todos sus documentos y que nunca haya dos documentos con el mismo autor y título.
También se usa un HashMap<String, Double> para tener el conjunto de palabras de todos los documentos con el número de documentos que contienen la palabra. En casos de inserción, eliminación y modificación de un documento hay que modificar también este HashMap que convendrá tenerlo actualizado para poder utilizar correctamente las funciones **buscarDocumentosParecidos** y **buscarDocumentosRelevantes**.

La parte más costosa de *listaDocumentos* es cuando hay que actualizar **numDocHaveTerm** en las funciones **decrementanumDocHaveTerm**(despues de hacer una **bajaDocumento**) y **incrementanumDocHaveTerm**(despues de hacer una **altaDocumento**) ya que hay que recorrer las “m” palabras que tiene el documento a borrar y/o las “n” palabras que tiene el documento a insertar, es decir, un coste O(n) en caso de **altaDocumento**, O(m) en caso de **bajaDocumento** y O(m+n) en caso de **modificarDocumento**. Las otras partes del código de altaDocumento(inserción) y bajaDocumento(eliminación) son como máximo de O(1)+O(1) cada una cuando hay que insertar/borrar primero el autor y después el documento por lo tanto seguirán teniendo sus respectivos costes O(n) y O(m).

En caso de consulta de un documento siempre será coste O(1)+O(1) al tener que buscar en los 2 HashMap, por lo tanto es prácticamente trivial.

Respecto a los algoritmos utilizados, en el método **ordenarLista** se usa el método sort de la clase *Collections* y *List*, y en el método **getTitulosDocumentosAutores** se recorre completamente el HashSet<String> en busca de los documentos de los autores que conforman el conjunto, por tanto el coste dependerá de la cantidad de autores.

Para buscar documentos parecidos y buscar documentos relevantes sobre un conjunto de palabras, utilizaremos el modelo de espacio vectorial para obtener la similitud entre documentos.

Primero, calcularemos las puntuaciones de similitud respecto a los demás documentos con las funciones calcularPuntuacionesSimilitud y calcularPuntuacionesSimilitudQuery, respectivamente. Estas funciones utilizan una estructura de datos de tipo HashMap<Doc, Double> para almacenar la puntuación de similitud de cada documento. Para calcular las puntuaciones de similitud, iteraremos sobre la lista de todos los documentos (en calcularPuntuacionesSimilitud quitamos de esta lista el documento dado, mientras que en calcularPuntuacionesSimilitudQuery convertimos la consulta a un documento auxiliar para poder compararlo) y utilizaremos las funciones similitud o similitudQuery, respectivamente, para calcular la similitud entre cada documento y el documento dado o la consulta dada.

La función similitud guarda todos los términos de ambos documentos en una estructura de datos de tipo HashSet<String>, mientras que similitudQuery solo guarda los términos de la consulta ya que son los únicos que nos interesan. Ambas declaran dos arreglos de double llamados vector, donde almacenaremos el vector de peso para cada término. Una vez obtenidos los vectores de peso, los pasamos a la función calcSimilitudCosenos para obtener la puntuación de similitud.

La similitud coseno se utiliza para medir la similitud entre dos vectores y se calcula como el producto interno entre ellos, dividido por el producto de las magnitudes de los vectores. Si alguno de los vectores es nulo, se devuelve 0. Para calcular el producto interno, utilizamos la función calcDotProduct, la cual utiliza una estructura de datos de tipo IntStream para

iterar sobre los elementos de los vectores y calcular el producto interno. Para calcular las magnitudes, utilizamos la función calcMagnitud, la cual calcula la magnitud como la raíz cuadrada de la suma de los cuadrados de cada elemento del vector. Si el vector es nulo, se devuelve 0. Esta función utiliza una estructura de datos de tipo DoubleStream para iterar sobre los elementos del vector y calcular la magnitud. Finalmente usamos obtenerDocumentosMasSimilares para devolver en forma de ArrayList<String> los documentos más similares. Para obtener los k documentos más similares, se utiliza una PriorityQueue que mantiene una ordenación de los documentos según sus puntuaciones de similitud. La cola se inicializa con todas las entradas del mapa puntuacionesSimilitud y luego se extraen los k elementos más altos (los más similares) de la cola. Ambas funciones tienen coste O(n log n) ya que primero hemos de visitar todos los documentos de listaDocumentos y luego ordenarlos en función de su similitud.

3.4. ListaAutor

Respecto a la entrega anterior, hemos mejorado significativamente la estructura de datos y algoritmos de *ListaAutor*. Ahora usamos un árbol de prefijos o Trie, por tanto para consultar si un autor existe o para insertar o eliminar un autor el coste es O(m) siendo m la longitud del nombre de autor, la cantidad de carácteres que conforman el nombre.
Finalmente para la consulta de autores por prefijo, el coste en el mejor de los casos es de m siendo m la longitud del prefijo, mientras que en el peor de los casos es x*n, siendo x el número de autores que contienen el prefijo y n la altura del árbol, que se resume en O(n).

3.5. ListaExpresionBooleana

En *ListaExpresionBooleana* se usa un LinkedHashMap<String,ExpresionBooleana>, para no tener expresiones repetidas y que esté ordenado por orden de inserción, para que el usuario a la hora de consultar documentos por expresión booleana tenga una lista de expresiones recién usadas.. Además permite también que el coste de consulta e inserción de una expresión booleana es de O(1).

3.6. ExpresionBooleana y BoolNode

Para representar una expresión booleana se usa un String para almacenar la expresión introducida, y una nueva clase *BoolNode*, para facilitar el tratamiento de la expresión booleana.

La nueva clase *BoolNode*, se basa en la idea de un *Binary tree*, en que cada nodo tiene un valor, y está enlazado con otros dos nodos, izquierdo y derecho. De esta forma permite dividir la expresión en sub expresiones.

3.7. Buscar documentos por Expresion Booleana

La búsqueda de documentos por expresión booleanas se hace de la siguiente manera:
Primero seleccionamos la expresión que queremos usar, cogemos todos los documentos que contenga términos de la expresión, dividimos cada documento en conjuntos de frases, para cada frase hacemos la comprobación con la expresión booleanas, si la frase cumple la expresión booleana, añadimos este documento al conjunto de resultados, y saltamos al siguiente conjunto de frases de documento.
Los costes son O(D), siendo D el número de documentos total, más el coste de dividir los documentos en conjunto de frases O(D), para cada frase aplicar la comprobación O(F*C), F total de frases, C coste de comprobación, la C consiste en O(L*T), L longitud de la frase y T el número de términos de la expresión.