GESTOR D'ARXIUS

Projecte de Programació

Quadrimestre Tardor 22/23

Identificador de l'equip: 42.1

Alós Cuadrado, Jaume
Cabero Armengol, Marc
El Mrabet Abous, Elias
Hispano Corbera, Dídac

Versió del lliurament: 3.1

3ª entrega PROP

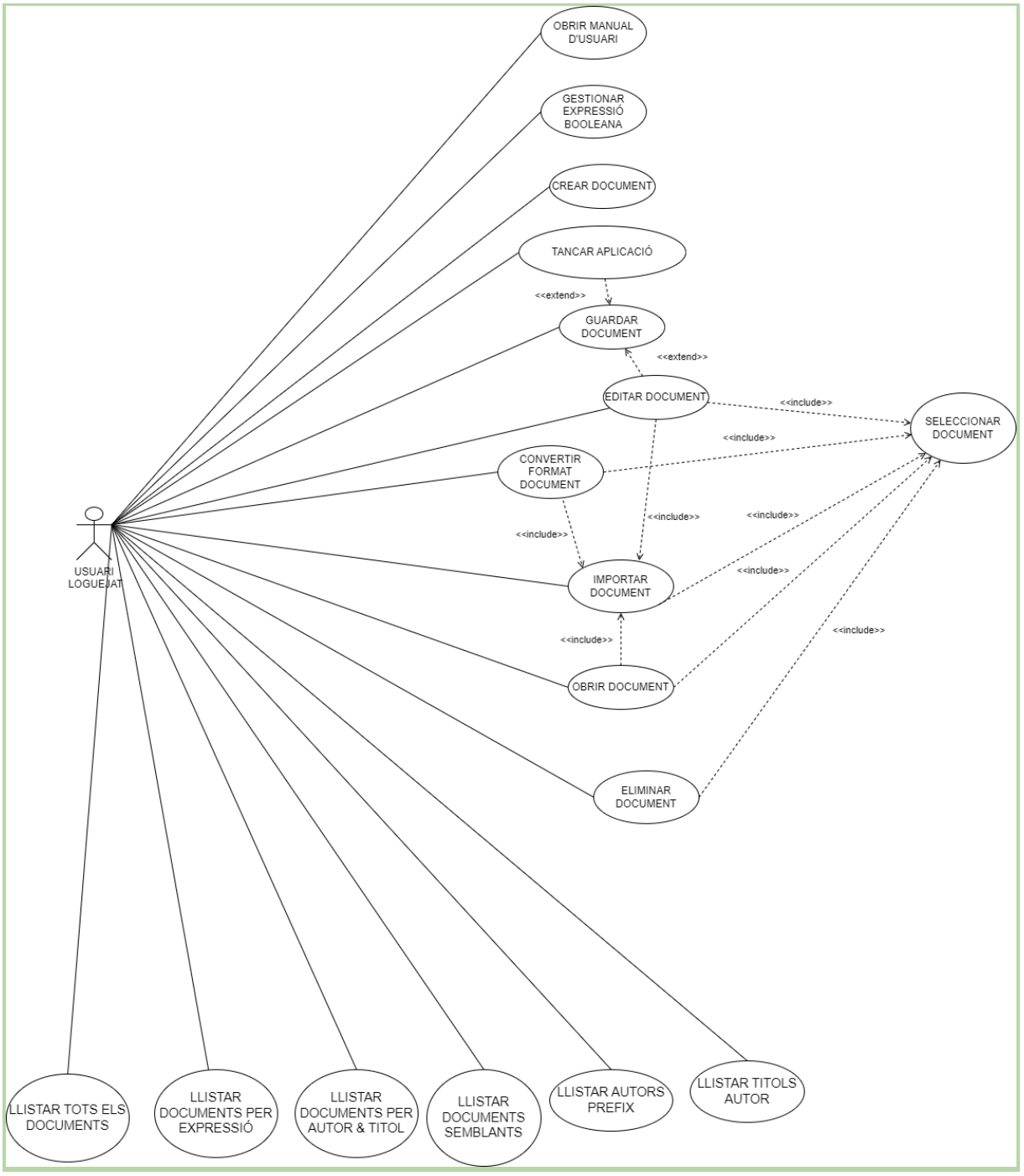
FIB

UPC

Índex

1.Diagrama de casos d'ús

A continuació adjuntem el diagrama de casos d'ús. Aquest, es pot trobar detalladament en el directori *DOCS*.



1.1. Descripció de Casos d'ús

1.1.1. Gestionar documents

En aquest apartat englobarem els casos d'ús que guardin relació amb la gestió dels documents.

Cas d'ús 1	
Nom	Crear document
Actor	Usuari
Comportament	1.L'usuari executa la funcionalitat "Crear document" 2.El sistema mostra un formulari en una interfície per tal que l'usuari introdueixi les dades *[autor, títol, contingut i format]* 3.L'usuari introdueix la informació *[autor, títol, contingut i format]* per la creació del document. 4.El sistema crea el nou document i torna a la interfície inicial.
Errors possibles i cursos alternatius	**[Error 1.1]** Existeix un document amb aquest títol i autor. **[Alternatiu 1.1]** El sistema mostra un missatge d'error *[Ja existeix un document amb mateix títol i autor]* **[Error 1.2]** Un o més camps *[autor, títol i contingut]* és buit. **[Alternatiu 1.2]** El sistema mostra un missatge d'error *[Les dades introduïdes son nul·les]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari cap al Menu Principal.

Cas d'ús 2	
Nom	Esborrar document definitivament
Actor	Usuari
Comportament	1. El sistema mostra en una interfície un formulari per a que l'usuari introdueixi les dades *[autor, títol]* o bé *[path]* per cercar el document que es desitja eliminar. 2. L'usuari introdueix les dades *[autor, títol]* o bé *[path]* per a la cerca del document. 3. El sistema cerca el document i mostra la seva informació. *"Cas d'Ús: Buscar Document"* 4 .L'usuari executa la funcionalitat "Esborrar". 5. El sistema esborra el document
Errors possibles i cursos alternatius	**[Error 2.1]** El document no està introduït al sistema **[Alternatiu 2.1]** El sistema mostra un missatge d'error *[No tens aquest document introduït al sistema]* **[Error 2.2]** Algun dels camps *[autor, títol]* o bé *[path]* és buit. **[Alternatiu 2.2]** El sistema mostra un missatge d'error *[No s'han introduït els paràmetres necessaris]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari cap a la interfície d'inici.

Cas d'ús 3	
Nom	Guardar document
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Guardar document". 2. El sistema mostra una interfície juntament amb un formulari per indicar el format en que es vol enregistrar el document *[xml o txt]* 3. L'usuari introdueix el format desitjat *[xml o txt]* 4. El sistema emmagatzema la informació i desa el document en el format seleccionat.
Errors possibles i cursos alternatius	**[Error 3.1]** Extensió pel document no identificada. **[Alternatiu 3.1]** El sistema mostra un missatge d'error *"El format indicat no està suportat"* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari cap a la interfície d'inici.

Cas d'ús 4	
Nom	Tancar document
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Tancar Document" estant ja en un document en ús. 2. El sistema tanca el document i redirigeix a l'usuari cap la pantalla d'inici.
Errors possibles i cursos alternatius	**[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari a la interfície d'inici.

Cas d'ús 5	
Nom	Editar Document
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Editar" 2.El sistema mostra un formulari en una interfície per tal que l'usuari introdueixi les dades *[autor, títol]* o bé *[path]* del document que es vol editar. 3.L'usuari introdueix la informació *[autor, títol]* o bé *[path]* del document que desitja editar. 4.El sistema cerca el document desitjat. Inicia *"Cas d'Ús: Buscar Document"*, 5.El sistema carrega el document que ha estat seleccionat usant *"Cas d'Ús: Carregar Document".* 6. El sistema permet l'elecció del camp que es vol modificar. 7. L'usuari introdueix la informació que vol modificar i salva els canvis. 8. El sistema emmagatzema els canvis efectuats.
Errors possibles i cursos alternatius	**[Error 5.1]** El fitxer desitjat no existeix. **[Alternatiu 5.1]** El sistema mostra un missatge d'error *[No existeix el document amb títol i autor especificats]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari a la interfície d'inici.

Cas d'ús 6	
Nom	Convertir format document
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Convertir" 2. El sistema mostra un formulari en una interfície per tal que l'usuari introdueixi les dades *[autor i nom document]* 3. L'usuari introdueix la informació *[autor i nom document]* per la cerca del document. 4. El sistema cerca el document desitjat. Inicia *"Cas d'Ús: Buscar Document"* 5.L'usuari selecciona el document cercat *"Cas d'Ús: Seleccionar Document"* 6.El sistema mostra una interfície a l'usuari per què seleccioni el nou format del document *[xml o txt].* 7. L'usuari escull el format al que vol convertir el document entre les opcions existents *[xml o txt].* 8. El sistema converteix el document desitjat al format seleccionat i el desa.
Errors possibles i cursos alternatius	**[Error 6.1]** El fitxer desitjat no existeix. **[Alternatiu 6.1]** El sistema mostra un missatge d'error *[No existeix el document amb títol i autor especificats]* **[Error 6.2]** El format del document al que es vol convertir és el format actual del document **[Alternatiu 6.2]** El sistema mostra un missatge d'error *[No es pot convertir un document al mateix format que ja té]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari a la interfície d'inici.

Cas d'ús 7	
Nom	Carregar Document
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Carregar" 2. El sistema mostra un formulari en una interfície per tal que l'usuari introdueixi les dades *[autor i nom document]* del document que es desitja carregar 3. L'usuari introdueix la informació *[autor i nom document]* del document que desitja carregar. 4. El sistema cerca el document desitjat. *"Cas d'Ús: Buscar Document"* 5. L'usuari selecciona el document usant *"Cas d'Ús: Seleccionar Document"* 6. El sistema carrega el document seleccionat i el mostra en una interfície.
Errors possibles i cursos alternatius	**[Error 7.1]** Carregar un document no existent. **[Alternatiu 7.1]** El sistema mostra un missatge d'error *[No existeix el document amb títol i autor especificats]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari a la interfície d'inici.

Cas d'ús 8	
Nom	Renombrar document
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Renombrar" 2. El sistema mostra un formulari en una interfície per tal que l'usuari introdueixi les dades *[autor i nom document]* 3. L'usuari introdueix la informació *[autor i nom document]* per la cerca del document que es vol renombrar. 4. El sistema cerca el document desitjat. *"Cas d'Ús: Buscar Document"* 5. L'usuari selecciona el document usant *"Cas d'Ús: Seleccionar Document"* 6. El sistema mostra una interfície juntament amb un formulari per tal que l'usuari introdueixi la nova informació *[nou nom document]* 7. L'usuari introdueix la informació *[nou nom document]* per fer el canvi de nom. 8. El sistema enregistra el document amb la nova informació introduïda.
Errors possibles i cursos alternatius	**[Error 8.1]** El document no existeix. **[Alternatiu 8.1]** El sistema mostra un missatge d'error *[No existeix el document amb títol i autor especificats]* **[Error 8.2]** Renombrar un document amb el mateix nom. **[Alternatiu 8.2]** El sistema mostra un missatge d'error *[No es pot renombrar un document amb el mateix nom]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari a la interfície d'inici.

Cas d'ús 9	
Nom	Buscar document
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Buscar" 2. El sistema mostra un formulari en una interfície per tal que l'usuari introdueixi les dades *[autor i nom document]* 3. L'usuari introdueix la informació *[autor i nom document]* per la cerca del document. 4. El sistema cerca el document desitjat.
Errors possibles i cursos alternatius	**[Error 9.1]** El document no existeix **[Alternatiu 9.1]** El sistema mostra un missatge d'error *[No existeix el document amb títol i autor especificats]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari a la interfície d'inici.

Cas d'ús 10	
Nom	Seleccionar Document
Actor	Usuari
Comportament	1. L'usuari tria alguna funcionalitat del sistema. 2. El sistema mostra un formulari en una interfície per tal que l'usuari introdueixi les dades *[autor i nom document]* 3. L'usuari introdueix la informació *[autor i nom document]* per la cerca del document. 4. El sistema cerca el document desitjat, "Cas d'Ús: Buscar Document" 5. L'usuari selecciona el document que prèviament s'ha buscat.
Errors possibles i cursos alternatius	**[Error 10.1]** El document no existeix **[Alternatiu 10.1]** El sistema mostra un missatge d'error *[No existeix el document amb títol i autor especificats]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari a la interfície d'inici.

Cas d'ús 11	
Nom	Importar Document
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Importar Document" 2. El sistema mostra un formulari en una interfície per tal que l'usuari introdueixi les dades *[path]* 3. L'usuari introdueix la informació *[path]* per importar el document. 4. El sistema importa el document desitjat.
Errors possibles i cursos alternatius	**[Error 11.1]** El path no existeix **[Alternatiu 11.1]** El sistema mostra un missatge d'error *[No existeix cap document en el path indicat]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema redirigeix a l'usuari a la interfície d'inici.

1.1.2. Gestionar consultes

Cas d'ús 1	
Nom	Llistar títols autor
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Llistar títols autor" 2. El sistema mostra un formulari juntament amb la interfície per tal que l'usuari introdueixi la informació *[nom autor]* 3. L'usuari introdueix el *[nom autor]* que desitja. 4. El sistema processa les dades i mostra tots els títols del autor introduït.
Errors possibles i cursos alternatius	**[Error 1.1]** L'autor no existeix . **[Alternatiu 1.1]** El sistema mostra un missatge d'error *"L'autor no existeix"* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema mostra un missatge d'error *"Hi ha hagut un error"* i redirigeix a l'usuari a la interfície de llistats

Cas d'ús 2	
Nom	Llistar autors prefix
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Llistar autors prefix" 2. El sistema mostra un formulari juntament amb la interfície per tal que l'usuari introdueixi el prefix desitjat *[prefix autor]* 3. L'usuari introdueix el *[prefix autor]* que desitja. 4. El sistema processa les dades i mostra tots els autors que continguin el prefix introduït.
Errors possibles i cursos alternatius	**[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema mostra un missatge d'error *"Hi ha hagut un error"* i redirigeix a l'usuari a la interfície de llistats

Cas d'ús 3	
Nom	Llistar document per títol & autor
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Llistar títols autor" 2. El sistema mostra un formulari juntament amb la interfície per tal que l'usuari introdueixi la informació *[títol i nom autor]* 3. L'usuari introdueix el *[títol i nom autor]* que desitja. 4. El sistema processa les dades i mostra el document associat amb la informació definida.
Errors possibles i cursos alternatius	**[Error 3.1]** El document no existeix **[Alternatiu 3.1]** El sistema mostra un missatge d'error *[No existeix el document amb títol i autor especificats]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema mostra un missatge d'error *"Hi ha hagut un error"* i redirigeix a l'usuari a la interfície de llistats

Cas d'ús 4	
Nom	Llistar k documents semblants
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Llistar documents semblants" 2. El sistema mostra un formulari juntament amb la interfície per tal que l'usuari introdueixi la informació *[nom autor, títol document i num_documents]* 3. L'usuari introdueix el *[nom autor, títol document i num_documents]* 4.El sistema cerca el document *[nom autor i títol document]* "Cas d'Ús: Buscar Document" 5.El sistema processa les dades i mostra els [num_documents] documents semblants al document especificat.
Errors possibles i cursos alternatius	**[Error 4.1]** El document no existeix **[Alternatiu 4.1]** El sistema mostra un missatge d'error *[No existeix el document amb títol i autor especificats]* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema mostra un missatge d'error *"Hi ha hagut un error"* i redirigeix a l'usuari a la interfície de llistats

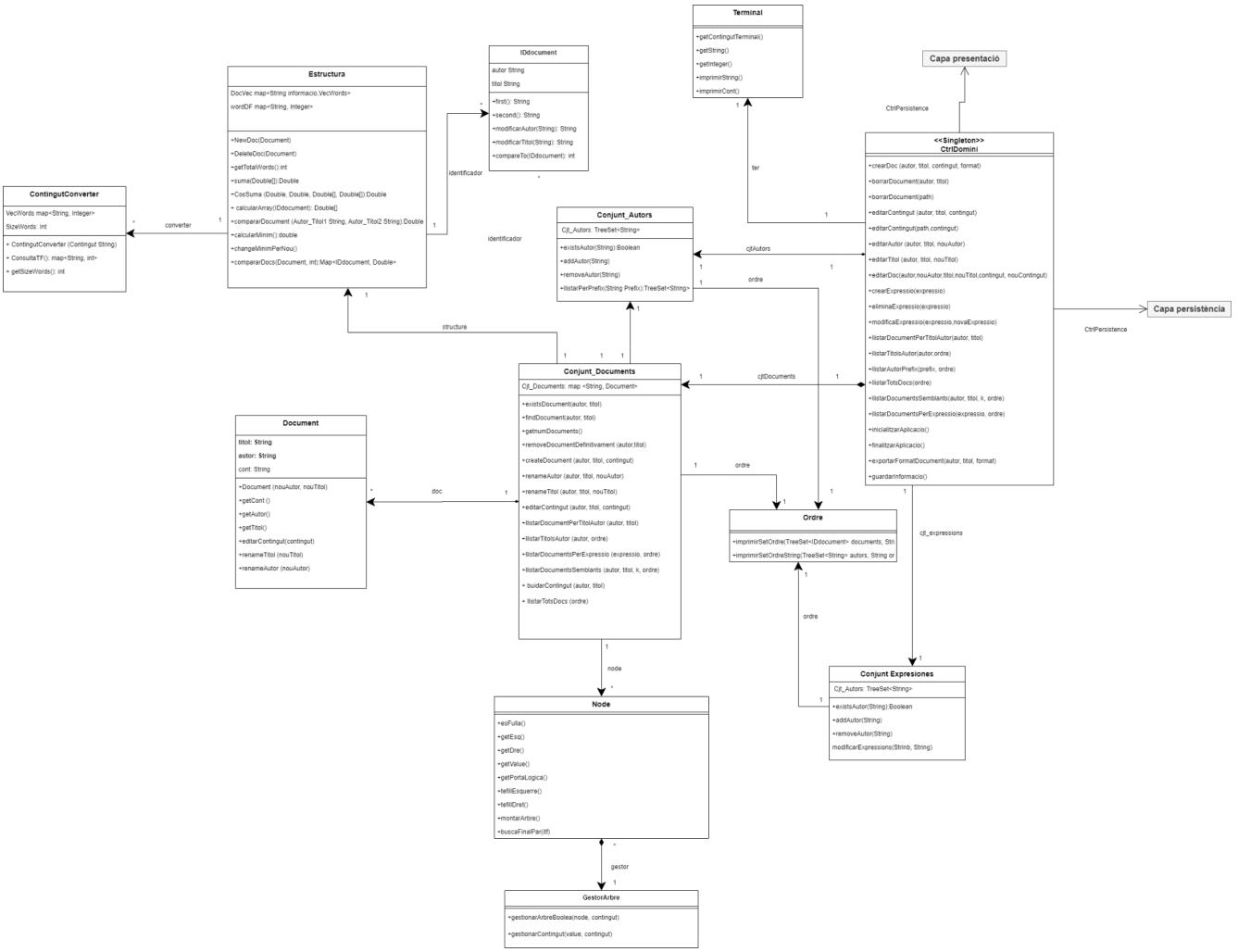
Cas d'ús 5	
Nom	Llistar documents per expressió
Actor	Usuari
Comportament	1. L'usuari executa la funcionalitat "Llistar documents per expressió" 2. El sistema mostra un formulari juntament amb la interfície per tal que l'usuari introdueixi l'expressió desitjada 3. L'usuari introdueix la *[expressió]* que desitja. 4. El sistema processa les dades i mostra tots documents que continguin l'expressió introduïda.
Errors possibles i cursos alternatius	**[Error 5.1]** Expressió no vàlida **[Alternatiu 5.1]** Es mostra un missatge d'error *"L'expressió introduïda no és vàlida"* **[Alternatiu general]** Si salta cap error, a més del comportament associat a l'error si escau,el sistema mostra un missatge d'error *"Hi ha hagut un error"* i redirigeix a l'usuari a la interfície de llistats

2. Capa Domini

2.1. Diagrama estàtic del model conceptual de dades

2.1.1. Model conceptual

A continuació adjuntem el model conceptual. Aquest, es pot trobar detalladament en el directori *DOCS*.



2.2. Breu explicació de les classes

En aquest apartat justificarem l'ús de les següents classes i la funció que ocupa cadascuna, a més de comentar alguns dels mètodes més destacats.

2.2.1. Document

Aquesta classe conté tots els elements que formen l'arxiu, és a dir, autor, títol i el seu contingut. S'encarrega principalment de la dels atributs propis, així com modificar contingut, renombrar el títol o autor.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *editarContingut (nouContingut)*: donat un document identificat mitjançant els paràmetres d'entrada, es modifica el contingut per un nou i s'emmagatzema aquesta modificació.
- *editarAutor (nouAutor)* : donat un document identificat mitjançant els paràmetres d'entrada, es modifica el nom del autor que hi havia, i s'actualitza l'identificador del document.
- *getCont()* : donat un document identificat mitjançant els paràmetres d'entrada, retorna el contingut d'aquest.

2.2.2. ConjuntDocuments

Aquesta classe conté tots els documents existents en el sistema, juntament amb un identificador que consisteix en el seu títol seguit del seu autor per facilitar la cerca dels documents.
S'encarrega de gestionar els documents a través de la creació, l'edició i la supressió d'aquests, a més d'efectuar certes comprovacions per tal que el funcionament sigui el correcte.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *existsDocument (autor,titol)* : donats uns paràmetres d'entrada que identifiquen un document, es retorna si aquest existeix en el sistema.
- *getNumDocumentsAutor (autor)* : donat un nom d'autor com a paràmetre d'entrada, es retorna el nombre total de documents que li pertanyen.
- *createDocument (autor,titol,contingut)* : donats uns paràmetres d'entrada, es crea un document amb aquests en el cas que no hagi fallat cap comprovació asociada als paràmetres.

2.2.3. ConjuntAutors

Aquesta classe conté tots els autors introduïts a l'aplicació. S'encarrega d'emmagatzemar els autors, afegint i eliminant autors quan sigui necessari.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *existsAutor (autor)* : donats un paràmetre d'entrada que identifiquen un autor, es retorna si aquest existeix en el sistema.
- *llistarPerPrefix (prefix)* : donat un paràmetre d'entrada que conté un prefix, es retornen tots els autors que contenen aquest.

2.2.4. ConjuntExpressions

Aquesta classe conté totes les expressions booleanas que han sigut donades d'alta pels usuaris.
S'encarrega de gestionar les expressions emmagatzemades, afegint i eliminant-les quan resulta necessari.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *existsExpressio (expressio)* : donats un paràmetre d'entrada que identifica una expressió, es retorna si aquest existeix en el sistema.
- *modificaExpressio (expressio, novaExpressio)* : donats uns paràmetres d'entrada, es modifica una expressió existent en el sistema per una nova.

2.2.5. Node

Aquesta classe conté la lògica necessària per poder realitzar el tractament d'una expressió booleana i estructura en el format d'un arbre.
S'encarrega de convertir una expressió booleana donada per l'usuari a un arbre que contingui l'expressió per tal que resulti més senzilla la seva gestió.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *montarArbre()* : donada una expressió booleana, s'encarrega de convertir-lo en un arbre realitzant els canvis necessaris.
- *buscarfinalpar (iterador)* : donada una expressió s'encarrega de buscar el final d'un parèntesis i col·locar-hi un interador per facilitar la seva gestió.

2.2.6. GestorArbre

Aquesta classe conté la lògica necessària per gestionar les expressions booleanas emmagatzemades.
S'encarrega de gestionar les expressions estructurades en arbres, realitzant una distinció entre els diferents formats que pot tenir aquest.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *gestionarContingut (value, contingut)* : donat un contingut i el value d'una fulla es comprova si aquest està o no dintre de contingut.

2.2.7. ContingutConverter

Aquesta classe és l'encarregada de convertir un String contingut en un Map<String, Integer> on cada String del Map representa un subString del String original, és a dir una paraula i totes les seves repeticions al llarg del document.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *consultaTF()* : mètode que permet obtenir una estructura de dades amb tots els valors del TF per efectuar una sèrie de càlculs necessaris en algunes funcionalitats requerides.

2.2.8. Estructura

Aquesta classe emmagatzema la informació de tots els documents estructurat en Maps emmagatzemant els pesos dels paràmetres freqüència de paraula **(*tf*)** i freqüència inversa del document **(i*df*)**. S'encarrega de comparar els diferents documents i veure quins s'assemblen més entre ells, mitjançant la realització dels càlculs necessaris fent ús del algorisme Tf-idf.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *comparaDoc (document, k)* : donats un document i un enter com a paràmetres d'entrada, s'encarrega de comparar els documents existents en el sistema amb el que s'ha introduït, cercant els k documents més semblants.
-

2.2.8 Terminal

Aquesta classe s'encarrega de realitzar les lectures i escriptures a la Terminal, per tal de proveir al sistema la informació necessària.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- getContingutTerminal() : gestiona l'entrada d'informació a través de la consola de l'ordinador de l'usuari i l'emmagatzema per proveir-la als mètodes que la requereixen.

2.2.8 IDdocument

Aquesta classe s'encarrega d'emmagatzemar el id de un document, format per un String autor i un String títol.

2.2.9. ConjuntPaths

Aquesta classe conté tots els *paths* dels documents han sigut creats en el sistema pels usuaris.

2.3. Breu descripció del controlador

En aquest apartat justifiquem les funcionalitats del controlador i les seves relacions amb altres classes.

2.3.1. CtrDomini

Aquesta classe és l'encarregada d'invocar la classe Terminal per llegir la informació introduïda per l'usuari i mostrar la informació necessària a través de la Terminal, a més de comunicar-se amb el Conjunt_Autor i Conjunt_Documents, i proporcionar tots els paràmetres que resulten necessaris.

2.4. Relació entre les classes

A continuació adjuntem la explicació de les relacions que hi ha establerta entre les diferents classes implementades.

- Conjunt_Documents 1 - * Document : El conjunt de documents accedeix a la classe Document per actualitzar, introduir o consultar atributs d'aquesta classe.
- Conjunt_Documents 1 - 1 Estructura : El conjunt de documents accedeix a la classe Estructura per poder calcular freqüències com el tf o el df i llistar els documents més semblants.
- Conjunt_Documents 1 - 1 ExpressioBooleana : El conjunt de documents accedeix a la classe Expressió booleana per consultar si una expressió es compleix per un contingut d'un document.
- Expressio_Booleana * - 1 GestorEpressióBoleana : ExpressioBooleana accedeix a la classe GestorExpressióBoleana per comprovar que les expressions tenen el format correcte i convertir l'expressió String a una expressió formada per true, false, and, or i/o parèntesis.
- Conjunt_Documents 1 - 1 CtrlDomini : El CtrlDomini accedeix a Conjunt de Documents per poder administrar els documents, creant, borrant, llistant o modificant arxius.
- Conjunt_Autors 1 - 1 CtrlDomini : El Controlador de Domini accedeix al conjunt d'Autors per actualitzar el set guardant tots els autors del sistema i també accedeix al conjunt per poder llistar tots els autors que comencin per un prefix.
- Estructura 1 - * ContingutConverter : La classe Estructura accedeix a contingutConverter per poder consultar el VecWords que té cada document, és necessari per poder calcular els valors del tf il df.
- IDdocument * - 1 Estructura: La classe estructura pot tenir cap o molts IDdocuments per tal d'identificar el conjunt de paraules de cada document. Un IDdocument només pot estar una vegada a Estructura, ja que no admet repetits.
- IDdocument * - 1 Conjunt_Documents: La classe Conjunt_Documents pot tenir cap o Molts IDdocuments per tal d'identificar cada document. Un IDdocument només pot estar una vegada a Conjunt_Documents, ja que no admet documents repetits.
- CtrlDomini 1 - 1 Terminal : El controlador Domini accedeix a la classe terminal per llegir o escriure, com diu el nom, per terminal.

2.5. Breu descripció de les Estructures de dades i algorismes

ConjuntDocuments

Estructura: TreeMap <IDdocument, Document>

Es va decidir emprar aquesta estructura de dades davant la resta degut pels següents motius.

En primer lloc, ens resulta necessària una estructura de dades que emmagatzemi dos components, un identificador dels documents junt amb el document esmentat. Per això, vam optar per un Map.

Quant a costos, sabem que aquesta estructura està implementada fent ús d'arbres, per la qual cosa és prou eficient, tenint un cost logarítmic pels mètodes com contains, get o put.

Es va dubtar entre utilitzar fer ús d'un TreeMap o un HashMap, però com requerim tenir les dades ordenades, sense saber el nombre de documents totals, i evitar l'entrada de valors nulls, es va decidir per TreeMap.

Per altra banda, el HashMap tot i que té cost constant en contains, get i put, sense saber un tamany aproximat d'aquest i afegint i esborrant constantment components del hash, eventualment aquest farà rehash. El fet de fer el rehash té un cost lineal en funció dels elements d'aquest. Addicionalment, es necessita també una quantitat de temps proporcional a la mida del nou hash per a crear-lo. El TreeMap sempre té cost logarítmic per fer get, put i contains.

El cost d'alguns mètodes es el següent:

- llistarDocumentPerTitolAutor(String autor, String títol) - Θ(log n)
  El cost resulta d'usar el mètode find en l'estructura de dades definida.

- llistarDocumentsPerExpressio(String expressio) - Θ(n^2)
  El cost és degut a que hem de recorrer tots els documents, i per cada document comprobar si el contingut compleix amb l'arbre que en cas pitjor és n.

ConjuntAutors

Estructura: TreeSet<String>

Es va decidir emprar aquesta estructura de dades davant la resta degut als següents motius:

En primer lloc, ens resulta necessària una estructura que emmagatzema la informació del nom d'un autor com String. Per això vam optar per un Set format per Strings.

Aquesta estructura ens aporta nombrosos aspectes tals com la no repetició d'elements, l'ordenació d'aquests i un cost logarítmic pels mètodes crear, afegir i buscar entre d'altres.
A escala interna, aquesta estructura es troba implementada en forma d'arbre, resultant d'aquí la seva eficiència.

El cost d'alguns mètodes es el següent:

- llistarPerPrefix(String prefix) - Θ (n)
  El cost resulta de recórrer l'estructura, on n és el nombre d'autors que hi ha inclosos.

ContingutConverter

Estructura: TreeMap <String, Double>

Es va decidir emprar aquesta estructura de dades davant la resta degut pels següents motius.
En primer lloc, ens resulta necessari una estructura que emmagatzema totes les paraules amb el seu tf corresponen. Per això, es va optar per un Map
S'ha optat per aquesta estructura, ja que, ens cal tenir les paraules emmagatzemades ordenades, aprofitant l'eficiència que ens proveeix en crides com put, contains o get, que tenen un cost logarítmic.
A escala interna, aquesta estructura es troba implementada en forma d'arbre, resultant d'aquí la seva eficiència.

El cost d'alguns mètodes es el següent:

- ContingutConverter(String contingut) - Θ (nlog n)
  El cost resulta de la creadora d'aquesta classe, on n és el nombre totals de paraules d'un contingut.
  Cal realitzar un recorregut de les n paraules existents Θ(n), i en cada iteració cal cridar al mètode contains amb un cost Θ(log n).

Estructura

Estructura: TreeMap<IDdocument, ContingutConverter>

Es va decidir emprar aquesta estructura de dades davant la resta degut pels següents motius.
En primer lloc, ens resulta necessari una estructura de dades que emmagatzemi dos components, un identificador pels documents juntament amb les paraules existents. Per això, es va optar per un Map.

S'ha optat per aquesta estructura, ja que, ens cal tenir les paraules emmagatzemades ordenades, aprofitant l'eficiència que ens proveeix en crides com put, contains o get, que tenen un cost logarítmic.
A escala interna, aquesta estructura es troba implementada de tal forma d'un arbre, resultant d'aquí la seva eficiència.

El cost d'alguns mètodes es el següent:

- compararDocs(Document A, INteger k) - n log(n)
  El cost resulta d'haver de recórrer els n documents existents, accedint a l'estructura de dades de cadascun amb un cost logarítmic log(n), per a fer les comparacions.

Node

Comptem amb una estructura de dades que encapsula una expressió booleana emmagatzemant la informació fent ús d'arbres.
Donada una expressió:

a OR b AND (a OR b)

D'un inici, el node contindrà l'expressió sencera, per a partir d'aquí començar el procediment.
S'inicia una cerca d'esquerra a dreta buscant un operador lògic que no estigui dins d'un parèntesis. Un cop trobat, es defineix com a fill esquerre la informació que hi hagi abans d'aquest, deixant en el fill dret tota la expressió restant.

a - fill esquerra

b AND (a OR b) - fill dret

El fill esquerre és una fulla , per tant, ja no seguim processant més, i ens centrem en la informació restant.
Apliquem el mateix procediment esmentat anteriorment per la informació que resta processar

b - fill esquerra

(a OR b) - fill dret

Tornem a tenir una fulla, per tant, ja no seguim processant aquell camí i ens centrem en la informació restant.

a - fill esquerra

b - fill dret

Un cop arribem al punt on no hi ha més portes lògiques a processar, podem afirmar que tenim l'arbre de la expressió booleana muntat.

El cost d'alguns mètodes es el següent:

- montarArbre() - Θ(n)
  El cost resulta d'haver de tractar l'expressió introduïda identificant de quin cas es tracta.

En casos com el següent:

(a OR b) AND (a OR b)

Com el primer operador lògic està entre parèntesis, aquest s'ignora i es comença per l'operador AND. Per tant els seus fills s'organitza de la següent manera:

a OR b - fill esquerra

a OR b - fill dret

A partir d'aquí es segueix amb el mètode com a l'exemple anterior.

GestorArbre

S'encarrega de la gestió de les expressions booleanes organitzades en forma d'arbre aplicant la recursivitat.

- Cas Base: És fulla
  En aquest escenari, haurem de comprovar en el contingut del document que es passa com a paràmetre, la condició corresponent fent ús del mètode contains.

- Cas Recursiu:
  - Té fill dret:
    Es realitza una crida recursiva, acotant cap a la dreta.

  - Té fill esquerre:
    Es realitza una crida recursiva, acotant cap a l'esquerra.

  - Té ambdós fills:
    En cas pitjor es realitzen dues crides recursives, una pel cas del fill dret i una altre pel cas del fill esquerre.
    En el següents casos amb només la crida recursiva esquerra és suficient.
    - El fill esquerra retorna "true" i he de fer una OR amb el dret
    - El fill esquerra retorna "false" i he de fer una AND amb el dret

El cost d'alguns mètodes en cas pitjor és el següent:

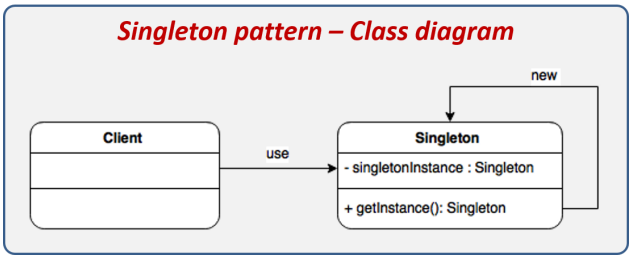
- gestionarArbreBoolea - Θ(n)
  El cost resulta d'haver de recórrer l'estructura sencera en el cas pitjor, però tenim afegida una poda que permet millorar aquest cost en la majoría dels casos fent que no es recorri tot l'arbre en alguns casos (explicat just prèviament).

2.6. Explicació de patrons arquitectònics s'empraran

El coneixement dels patrons de disseny resulten clau a l'hora d'abordar desenvolupaments i de solucionar problemes, ja que presenten solucions a problemes de disseny que ja han estat usats per resoldre problemes similars en ocasions anteriors.

2.6.1. Singleton

Aquest patró permet que només es pugui tenir una única instància per a tota l'aplicació d'una determinada classe. S'aconsegueix restringint la creació d'instàncies d'aquesta classe mitjançant l'operador new i imposant un constructor privat i un mètode estàtic per obtenir la instància.



L'objectiu de l'ús d'aquest patró és garantir que només hi pugui haver una única instància d'una classe determinada i que hi hagi una referència global en tota l'aplicació, que és el que busquem en el cas del controlador, ja que es busca tenir una única instància i que aquesta tingui una referència global en tota l'aplicació.

2.7. Metodología de Tests

En aquest apartat explicarem la metodologia de tests empleada per la comprovació correcta de les funcionalitats requerides de la capa de domini.

Per a comprovar la correcta execució de cadascun dels mètodes públics amb els quals compten les classes implementades, hem fet ús dels Test JUnit, que és una llibreria que permet assegurar que els mètodes es comporten adequadament davant a diferents situacions d'entrades.

En el directori **`[Path absolut: /FONTS/src/test]`** es poden trobar els tests de cadascuna de les classes, on cada mètode públic apareix explicat per facilitar la seva comprensió i saber la forma que s'ha encarat el test.

A continuació es mostra el format que presenta la explicació dels tests:

* **Objecte de la prova:** Nom del mètode a testejar.
* **Fitxers de dades necessaris:** Informació necessària per executar el test.
* **Valors estudiats:** Estratègia a emprar per realitzar el test..
* **Operativa:** Com s'efectua la comprovació.
* **Explicació del test:** Com es el funcionament del test.

En el directori **`[Path absolut: /FONTS/src/main/drivers]`** és on tenim els drivers del programa, en aquest cas el driver DriverDomini.java, ja que s'ha realitzat driver per controlador i en el nostre disseny, hem tingut un únic controlador.

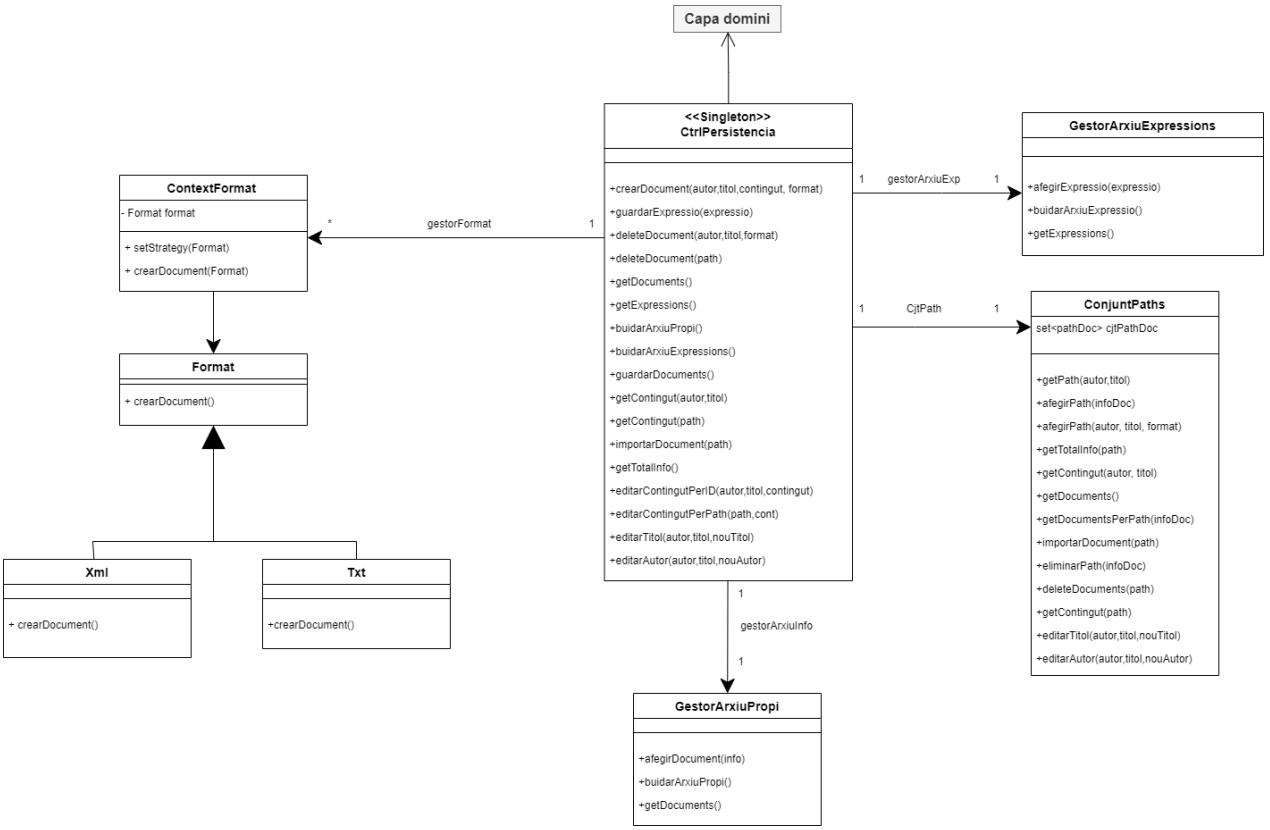
2.8. Distribució de les classes

	Marc	Elias	Dídac	Jaume
Document	X			
Conjunt_Documents		X		
Conjunt_Autors			X	
Node	X			
GestorArbre	X			
Ctrl_Domini				X
ContingutConverter			X	
Estructura			X	
Terminal				X
IDdocument		X		
Ordre	X			

3.Capa de Persistència

3.1. Diagrama de classes de la capa de persistència

A continuació adjuntem el model conceptual de la capa de persistència. Aquest, es pot trobar detalladament en el directori *DOCS*.



3.2. Breu explicació de les classes

En aquest apartat justificarem l'ús de les següents classes i la funció que ocupa cadascuna, també comentarem algun dels mètodes més destacats.

3.2.1. ContexFormat

Aquesta classe encapsula la lògica del patró estratègia empleat.
S'encarrega de decidir el format que es vol tractar d'un document en concret, que pot ser XML o text pla.

Alguns dels mètodes més destacats que trobem en aquesta classe són:

- *setStrategy (Format format)* : mètode que ens permet definir quina tipologia de document es vol tractar, si XML o TXT.

3.2.2. Format

Aquesta classe es tracta d'una interfície necessària per a la implementació del patró estratègia explicat amb posterioritat.
S'encarrega d'encapsular les varietats de càlculs existents i proporciona un mètode per cridar als mètodes concrets.

La interfície té mètodes abstractes que són implementats per les altres classes.

3.2.3. GestorArxiuExpressions

Aquesta classe gestiona les expressions booleanes que s'introdueixen en el sistema i les emmagatzema en un fitxer concret per guardar l'estat del sistema.

S'encarrega de gestionar les expressions booleanas emmagatzemades, afegint i eliminant-les quan resulta necessari, separat per una convenció definida.

[*expressio1***:fiExp:***expressio2*]

Algun dels mètodes més destacats que trobem en aquesta classe són:

- *afegirExpressio (String expressio)* : mètode que permet emmagatzemar en el fitxer d'expressions booleanes propi, per a permetre la seva gestió quan es tanca i es posa en marxa l'aplicació.

3.2.4. GestorArxiuPropi

Aquesta classe gestiona la informació que emmagatzema el sistema per realitzar el preprocessat inicial i el postprocessat al tancar, per guardar l'estat del sistema.

S'encarrega de gestionar l'arxiu del format propi i la informació que es troba a dins, que consta tant dels identificadors dels documents existents, a més del seu contingut, separat per una convenció definida.

[*nom_autor*:**titol**:*titol_document*:**path**:*path_document*:***fiDocu:***]

Algun dels mètodes més destacats que trobem en aquesta classe són:

- *afegirDocument (String info)* : mètode que permet emmagatzemar en el fitxer de format propi la informació d'un document per permetre la seva gestió quan es tanca i es posa en marxa l'aplicació.

3.2.5. TXT

Aquesta classe conte la lògica necessària pel tractament dels documents amb el format txt (*Plain Text*).
S'encarrega de la creació dels documents en aquest format a partir dels atributs especificats.

Algun dels mètodes més destacats que trobem en aquesta classe són:

- *crearDocument():* mètode que permet crear un document i emmagatzemar-lo en la memòria seguint el format TXT.

3.2.6. XML

Aquesta classe conte la lògica necessària pel tractament dels documents amb el format XML (*Extensible Markup Language*).
S'encarrega de la creació dels documents amb aquest format a partir dels atributs especificats.

Algun dels mètodes més destacats que trobem en aquesta classe són:

- crearDocument(): mètode que permet crear un document i emmagatzemar-lo en la memòria seguint el format XML.

3.2.7. ConjuntPaths

Aquesta classe conte la lògica necessària per emmagatzemar els paths dels documents creats. També obté és continguts dels documents quan és necessari cargar-los a memòria de programa.

Algun dels mètodes més destacats que trobem en aquesta classe són:

- *getContingut(String autor, String titol)*: mètode que permet extreure el contingut de un document, ja sigui txt o xml. Retorna un String i s'ajuda dels següents mètodes (també d'aquesta classe):
    - getContXml(String path)
    - getContTxt(String path)
- *afegirPath(String autor, String titol, String format):* mètode que permet crear i afegir el path d'un document al conjunt donat un autor, titol i format. També té una versió per a documents importats.

3.3. Breu explicació del controlador

En aquest apartat justifiquem les funcionalitats del controlador i les seves relacions amb altres classes.

3.3.1. CtrlPersistence

Aquesta classe és l'encarregada d'invocar i comunicar les classes GestorArxiuExpressions, GestorArxiuPropi, GestorPaths i ContextFormat i de proporcionar tots els paràmetres que resultin necessaris.

3.4. Relació entre les classes

- CtrlPersistència 1 - * ContextFormat: El controlador de Persistència accedeix a la classe per instanciar-la i proporcionar els atributs necessaris, també activa la lògica necessària a dur a terme. Tindrem tants ContextFormat com documents creats en XML i TXT.
- CtrlPersistència 1 - 1 GestorArxiuPropi: El controlador de Persistència accedeix a la classe per instanciar-la i proporcionar-li els atributs necessaris, a més d'activar la lògica necessària a dur a terme. Com només tenim un arxiu propi és una relació 1-1. Si en un futur en aquesta aplicació fos necessari utilitzar més arxius d'aquest tipus, la relació canviaria.
- CtrlPersistència 1 - 1 GestorArxiuExpressions: El controlador de Persistència accedeix a la classe per instanciar-la i proporcionar-li els atributs necessaris, a més d'activar la lògica necessària a dur a terme. Com només tenim un arxiu d'expressions és una relació 1-1. Si en un futur en aquesta aplicació fos necessari utilitzar més arxius d'aquest tipus, la relació canviaria.
- CtrlPersistència 1 - 1 ConjuntPaths: El controlador de Persistència accedeix a la classe ConjuntPaths per instanciar-la i proporcionar-li els atributs necessaris, a més d'activar la lògica necessària a dur a terme.

3.5. Possibles escenaris

Per tal de guardar la informació de l'aplicació després de finalitzar-la hem creat les funcions inicialitzarAplicacio() i finalitzarAplicacio(). Usem aquestes funcions per fer un preprocessat de la informació que tenim guardada, i introduïm a l'app els autors, títols i paths de tots els documents. D'aquesta manera podem guardar tota la informació entre usos de l'app sempre i quan aquesta finalitzi correctament.

Com l'usuari pot estar constantment creant i esborrant documents i expressions, veiem poc eficient actualitzar per cada cas el contingut dels arxius propis, ja que en cas d'eliminació o modificació hem de fer tantes cerques amb cost lineal com modificacions faci.

Sabem però, que l'usuari pot tenir la necessitat de guardar la informació sense tancar l'app, ja sigui perquè té poca bateria o per prevenir qualsevol possible motiu que faci que l'app s'apagui de manera incorrecta. Per donar tranquil·litat a l'usuari hem afegit una funció (guardarInformacio()) per tal de guardar tota la informació al format propi en qualsevol moment donat.

3.6. Breu descripció de les Estructures de dades i algorismes

En aquest apartat justificarem l'ús de les següents classes i la funció que ocupa cadascuna, a més de comentar alguns dels mètodes més destacats.

3.6.1. Arxiu.prop

Estructura que emmagatzema tota la informació necessària pel correcte funcionament del sistema i un processat eficient de la informació un cop es tanca i es torna a obrir l'aplicació.
La informació emmagatzemada consta dels atributs de cadascun dels documents existents en el sistema, identificats amb un autor i títol, que tenen associat un contingut.

Aquesta estructura es troba organitzada de la següent forma:

[*nom_autor*:**titol:***titol_document*:**path:***path_document*:***fiDocu:***]

3.6.2. Expressions.prop

Estructura que emmagatzema tota la informació necessària relacionada amb les expressions booleanes pel correcte funcionament dels llistats mitjançant expressions booleanes donades d'alta prèviament.
La informació emmagatzemada consta de les expressions booleanes que han estat donades d'alta en el sistema.

Aquesta estructura es troba organitzada de la següent forma:

[*expressio1*:**fiExp:***expressio2*]

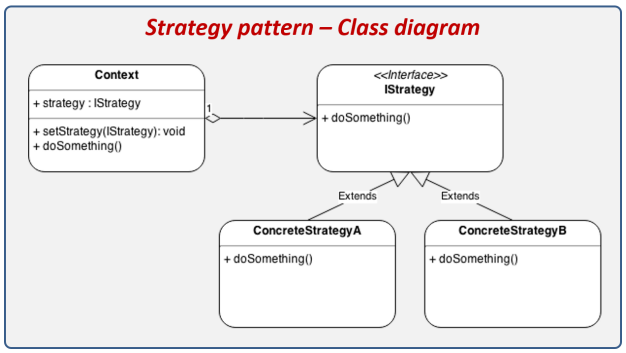
3.7. Breu descripció dels patrons arquitectònics emprats

El coneixement dels patrons de disseny resulten clau a l'hora d'abordar desenvolupaments i de solucionar problemes, ja que presenten solucions a problemes de disseny que ja han estat usats per resoldre problemes similars en ocasions anteriors.

3.7.1. Patró Estrategia

Aquest patró permet establir en temps d'execució el rol de comportament d'una classe, es basa en el polimorfisme per implementar una sèrie de comportaments que podran ser

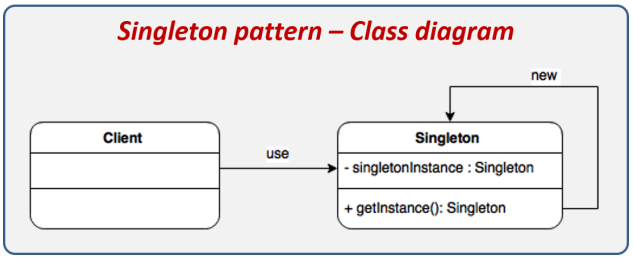
intercanviats durant l'execució del programa, aconseguint amb això que un objecte es pugui comportar de forma diferent segons l'estratègia establerta.



L'objectiu de l'ús d'aquest patró és poder treballar amb objectes tant de XML com de Text Pla segons el format especificat en el document.

3.7.2. Patró Singleton

Aquest patró permet que només es pugui tenir una única instància per a tota l'aplicació d'una determinada classe, que s'aconsegueix restringint la creació d'instàncies d'aquesta classe mitjançant l'operador new i imposant un constructor privat i un mètode estàtic per obtenir la instància.



L'objectiu de l'ús d'aquest patró és garantir que només hi pugui haver una única instància d'una classe determinada i que hi hagi una referència global en tota l'aplicació, que és el que busquem en el cas del controlador, ja que es busca tenir una única instància i que aquesta tingui una referència global en tota l'aplicació.

3.8. Metodología de Tests

En aquest apartat explicarem la metodologia de tests empleada per la comprovació correcta de les funcionalitats requerides de la capa de persistència.

Per a efectuar la prova de les funcionalitats relacionades amb l'àmbit de la persistència, és a dir aquelles gestions de la capa de memòria, s'ha optat per la realització de proves d'acord amb uns casos determinats, buscant l'obtenció dels resultats esperats per aquella situació.

A continuació, s'especifiquen alguns dels tests més destacats que han sigut requerits per tal d'avaluar el correcte funcionament de la casuística respectiva.

- Comprovar la creació correcta dels documents en memòria.
- Comprovar que la creació de documents amb els mateixos atributs està controlat.
- Comprovar la correcta localització del document creat en el path indicat.
- Comprovar la mida dels arxius emmagatzemats no és molt desbaratada.

3.9. Distribució de les classes

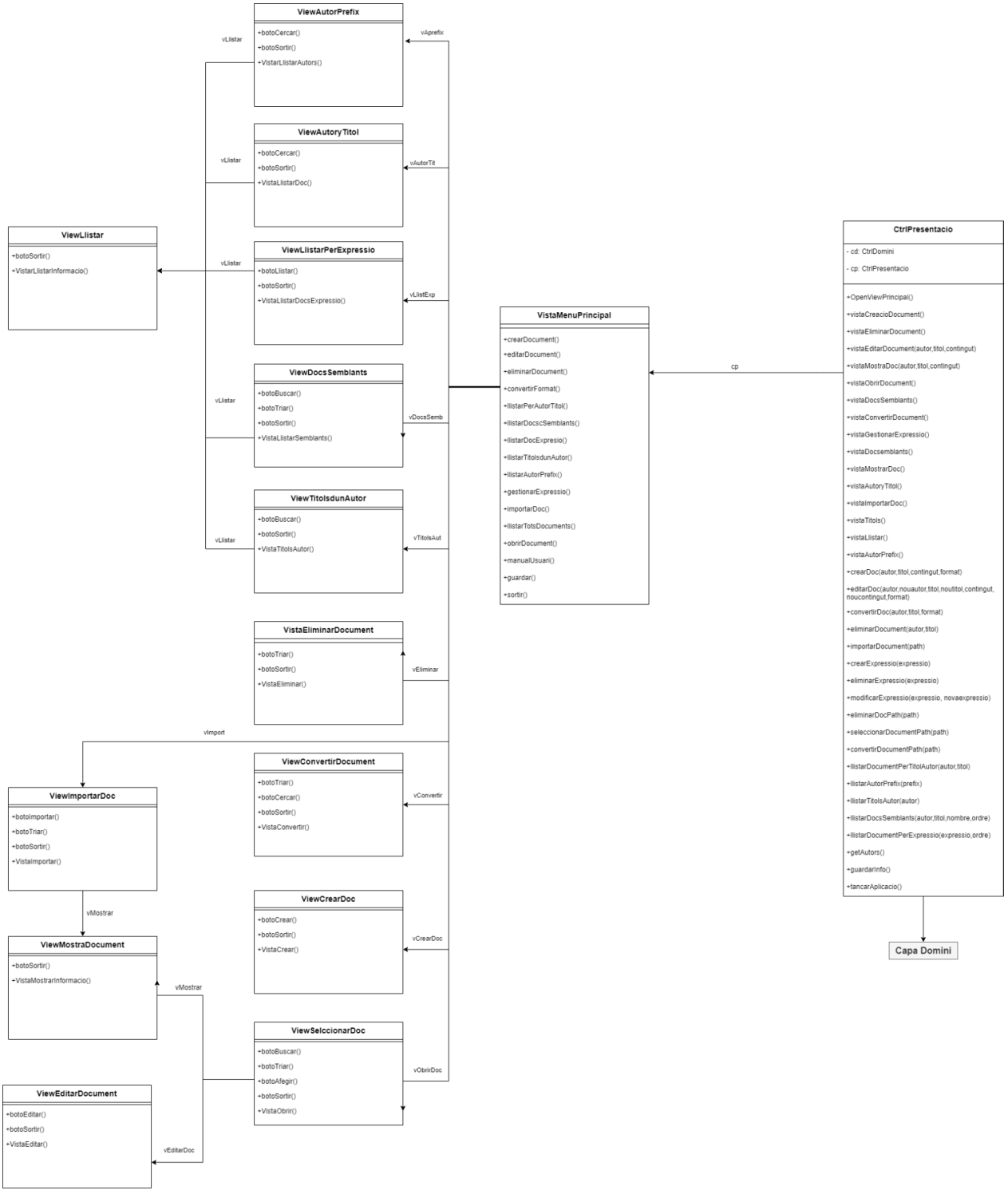
A continuació detallem l'autor de la implementació de cadascuna de les classes que compren el nostre projecte.

	Marc	Elias	Dídac	Jaume
CtrlPersistence			X	
ContextFormat	X			
Format			X	
ConjuntPaths	X			
GestorArxiuExpressions			X	
GestorArxiuPropi	X			
Xml			X	
Txt	X			

4. Capa de Presentació

4.1. Diagrama de classes de la capa de Presentació

A continuació adjuntem el model conceptual de la capa de persistència. Aquest, es pot trobar detalladament en el directori *DOCS*.



4.2. Breu explicació de les vistes

En aquest apartat justifiquem l'ús de les següents vistes i la funció que ocupa cadascuna, a més de comentar alguns dels mètodes més destacats.

4.2.1. ViewAutorPrefix

Aquesta vista és l'encarregada de gestionar l'entrada dels paràmetres necessaris i de llistar els autors que contenen un prefix determinat, que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada del prefix que es desitja a més d'indicar l'ordre en què es vol el llistat a través d'un desplegable amb dues opcions, [A a Z o de Z a A].
Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.2. ViewAutoryTitol

Aquesta vista és l'encarregada llistar els documents existents per títol i autor, que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris, autor i títol, dintre d'uns quadres de text, per un cop haver pitjat el botó Cercar, aquest document sigui mostrat en la vista corresponent. *{ViewMostrarDoc}*

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.3. ViewConvertirDocument

Aquesta vista és l'encarregada de convertir un document, que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris, autor i títol, dintre d'uns quadres de text per cercar el document que es desitja convertir, un cop pitjat el botó de Convertir.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.4. ViewCrearDoc

Aquesta vista és l'encarregada de crear un document, que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris, autor, títol i contingut, dintre d'uns quadres de text per crear un document amb aquests atributs i emmagatzemar-lo a memòria un cop pitjat el botó de Crear.
Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.5. ViewDocsSemblants

Aquesta vista és l'encarregada de llistar els documents semblants al document que se li indica,,que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris, autor, títol i un nombre enter, dintre d'uns quadres de text per identificar el document que es vol tenir com a referència, a més del nombre de documents que es desitja llistar, per un cop haver pitjat el botó Cercar, es mostri la informació necessària a la vista corresponent {VistaLlistar}.
Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.6. ViewEditarDocument

Aquesta vista és l'encarregada d'editar la informació que conté un document, que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris que es desitja modificar, dintre d'uns quadres de text perquè aquesta informació sigui canviada i emmagatzemada en pitjar el botó Editar.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.7. ViewMenuPrincipal

Aquesta vista és l'encarregada de mostrar la primera pantalla que apareix a l'executar el nostre sistema que consisteix en un menú amb les diferents funcionalitats que pot executar l'usuari, on cadascuna d'aquestes té associat un botó que portaran a vistes diferents encarregades de la funcionalitat,a més de l'existència d'un botó que servirà per sortir i tancar el programa.

4.2.8. ViewMostraDocument

Aquesta vista és l'encarregada de mostrar la informació d'un document, que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona la visualització de la informació d'un document seleccionat prèviament en la vista {ObrirDoc}.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.9. ViewObrirDoc

Aquesta vista és l'encarregada de llistar els documents per a poder seleccionar-ne un i obrir-lo, que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.
La vista obre l'explorador d'arxius on es podran seleccionar únicament fitxers que el nostre sistema és capaç d'obrir.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.10. ViewTitolsdunAutor

Aquesta vista és l'encarregada de llistar els documents que pertanyen a un autor en concret, que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada del paràmetre necessari, autor, dintre d'un quadre de text a més d'indicar l'ordre en què es vol el llistat mitjançant un desplegable amb dues opcions [A a Z o de Z a A], per un cop haver pitjat el botó de Cercar, es mostri la informació en la vista pertinent {ViewLlistar}.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.11. ViewGestionarExpressio

Aquesta vista és l'encarregada de gestionar les expressions que existeixen dintre del sistema, funcionalitat que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris, si escau com podría ser una nova expressió o eliminar una expressió ja existent, però també permet a l'usuari llistar aquelles expressions existents en el sistema dintre d'un quadre de text.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.12. ViewEliminarDocument

Aquesta vista és l'encarregada d'eliminar els documents que s'indiquen, funcionalitat que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris,autor i títol, dintre d'un quadre de text, per un cop haver pitjat el botó d'Eliminar, retirar el document del sistema.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.13. ViewLlistarPerExpressio

Aquesta vista és l'encarregada de mostrar els documents que compleixen amb l'expressió introduïda, funcionalitat que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris, expressió, dintre d'un quadre de text, per un cop haver pitjat el botó de Cercar, es mostri la informació en la vista pertinent {ViewLlistar}.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.14. ViewLlistarTotsDocuments

Aquesta vista és l'encarregada de mostrar tots els documents existents en el sistema, funcionalitat que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.
No existeix cap paràmetre d'entrada i mostra tant el autor com el títol dels documents.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.15. ViewImportarDocument

Aquesta vista és l'encarregada d'importar el document que s'indica, funcionalitat que apareix en seleccionar el botó d'aquesta funcionalitat des del Menú Principal.

Gestiona l'entrada dels paràmetres necessaris,path, dintre d'un quadre de text, per un cop haver pitjat el botó de Cercar, es mostri la informació en la vista pertinent {ViewLlistar}.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.2.16. ViewLlistar

Aquesta vista és l'encarregada de llistar tots els documents que compleixen una sèrie de característiques definides per cada funcionalitat, es mostra un cop s'ha definit els criteris definits per les vistes antecessores.

Gestiona la mostra de la informació necessària mitjançant dues columnes, per una banda, els autors i per altra banda els títols.

Es proporciona un botó de Sortir que tornarà a la VistaMenuPrincipal i cancel·larà l'execució d'aquesta funcionalitat.

4.3. Breu explicació del controlador

En aquest apartat justifiquem les funcionalitats del controlador i les seves relacions amb altres classes.

4.3.1. CtrlPresentacion

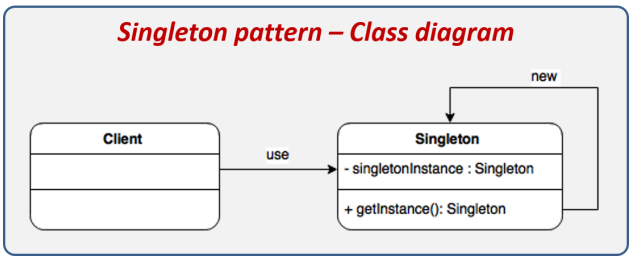
Aquesta classe és l'encarregada d'invocar, seleccionar les vistes apropiades en cada moment i comunicar les classes, a més de proporcionar tots els paràmetres que resultin necessaris.

4.4.Breu descripció dels patrons arquitectònics emprats

El coneixement dels patrons de disseny resulten clau a l'hora d'abordar desenvolupaments i de solucionar problemes, ja que presenten solucions a problemes de disseny que ja han estat usats per resoldre problemes similars en ocasions anteriors.

4.4.1. Patró Singleton

Aquest patró permet que només es pugui tenir una única instància per a tota l'aplicació d'una determinada classe, que s'aconsegueix restringint la creació d'instàncies d'aquesta classe mitjançant l'operador new i imposant un constructor privat i un mètode estàtic per obtenir la instància.



L'objectiu de l'ús d'aquest patró és garantir que només hi pugui haver una única instància d'una classe determinada i que hi hagi una referència global en tota l'aplicació, que és el que busquem en el cas del controlador, ja que es busca tenir una única instància i que aquesta tingui una referència global en tota l'aplicació.

4.5. Metodología de Tests

En aquest apartat explicarem la metodologia de tests empleada per la comprovació correcta de les funcionalitats requerides de la capa de presentació.

Per a efectuar la prova de les funcionalitats relacionades amb l'àmbit de la presentació, és a dir aquelles interficies que arriben als usuaris, s'ha optat per la realització de proves en base a uns casos determinats, buscant l'obtenció dels resultats esperats per aquella situació, tals com l'execució del programa i seleccionant les diferents vistes existents.

4.6. Distribució de les classes

A continuació detallem l'autor de la implementació de cadascuna de les classes que compren el nostre projecte.

	Marc	Elias	Dídac	Jaume
ViewAutorPrefix				X
ViewAutoryTitol				X
ViewConvertirDocument				X
ViewCrearDoc				X
ViewDocsSemblants				X
ViewEditarDocument				X
ViewEliminarDocument				X
ViewGestionarExpressió		X		
ViewLlistar		X		
ViewImportar		X		
ViewLlistarPerExpressio		X		
ViewMostraDocument		X		
ViewSeleccionarDocument		X		
ViewTitolsdunAutor		X		
CtrlPresentacio		X		