GESTIÓN DE DOCUMENTOS

Marc Igarza Mateo *marc.igarza*

David George Williams Corral *david.george.williams*

Marina Sauca Ortiz *marina.sauca*

Código del proyecto: 6.1

Versión:1.0

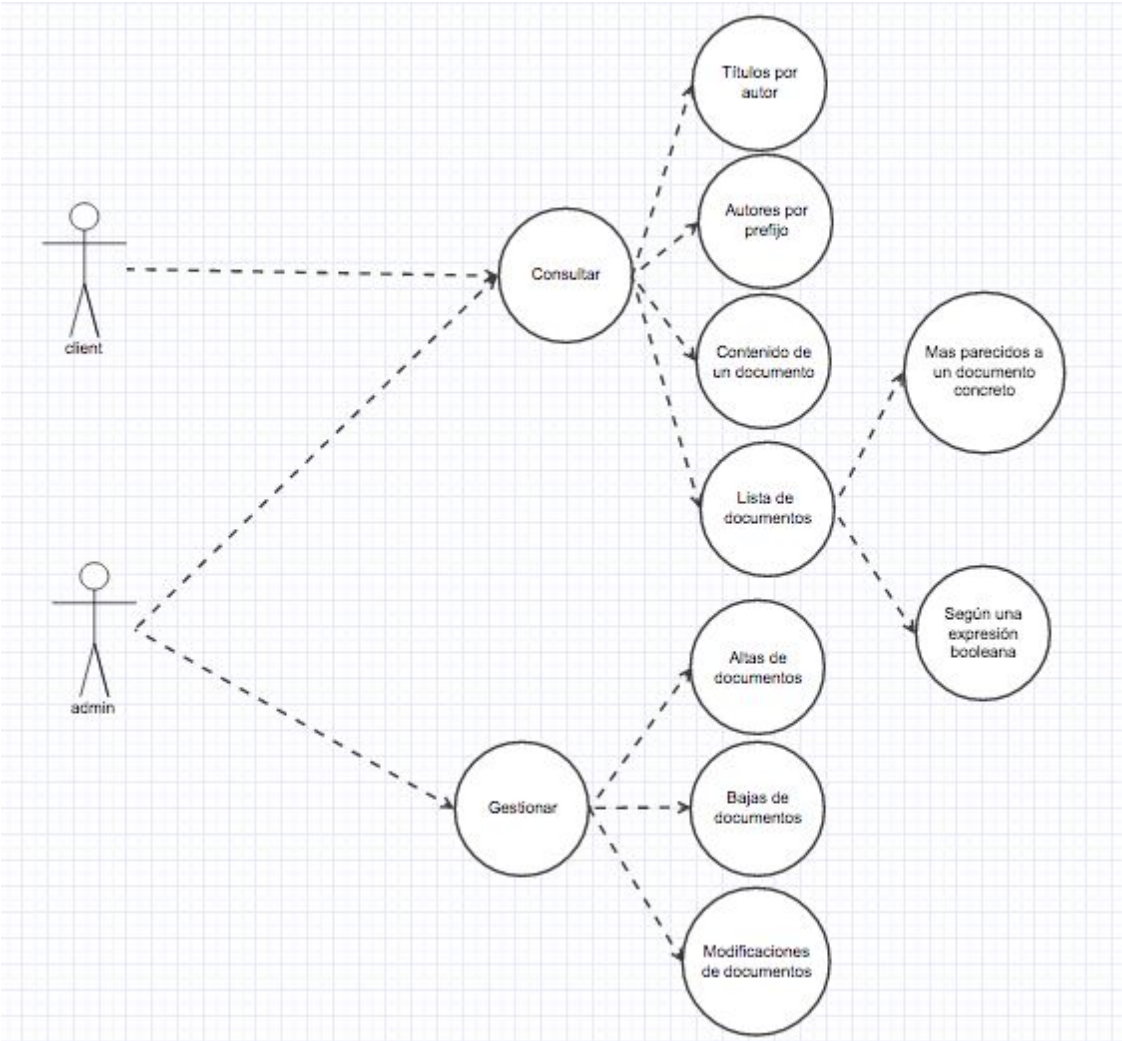
Barcelona, 11 de Noviembre de 2016

Facultad de Informática de Barcelona

Índice

1. Definición de los casos de uso

a. Diagrama de los casos de uso



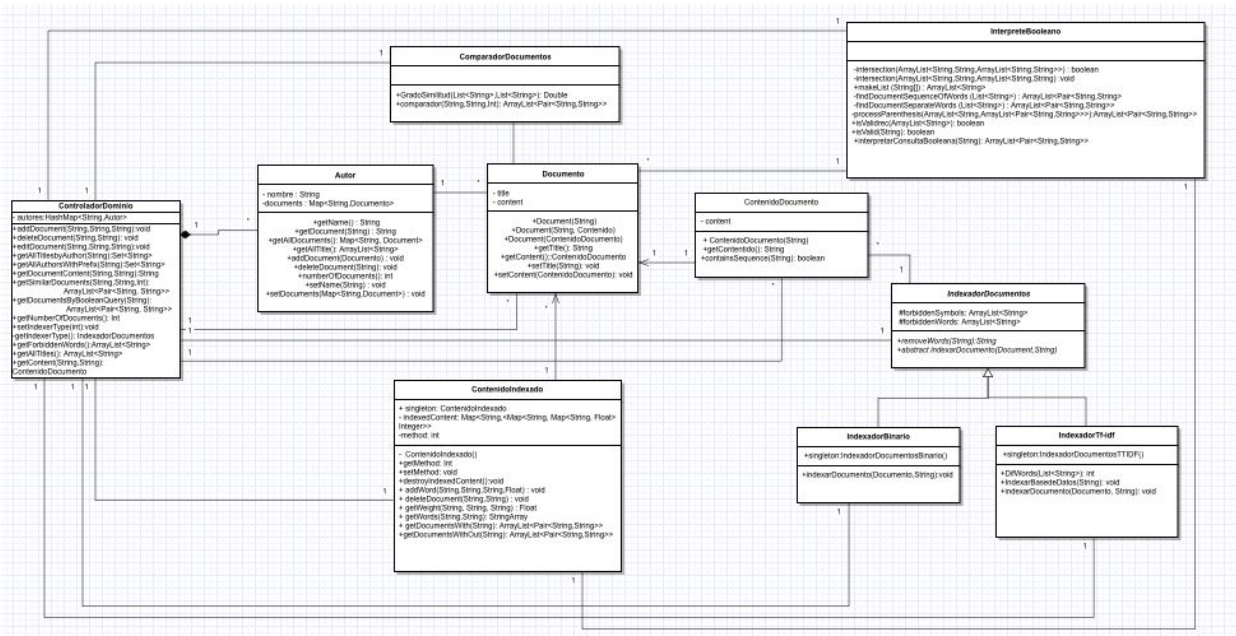
b. Descripción detallada de cada caso de uso

- Consultar títulos por autor: Sirve para que el usuario y/o el administrador pueda obtener todos los títulos de los documentos de un autor específico en la base de datos.
- Consultar autores por prefijo: Sirve para que el usuario pueda obtener todos los autores cuyo nombre empieza por el prefijo indicado.

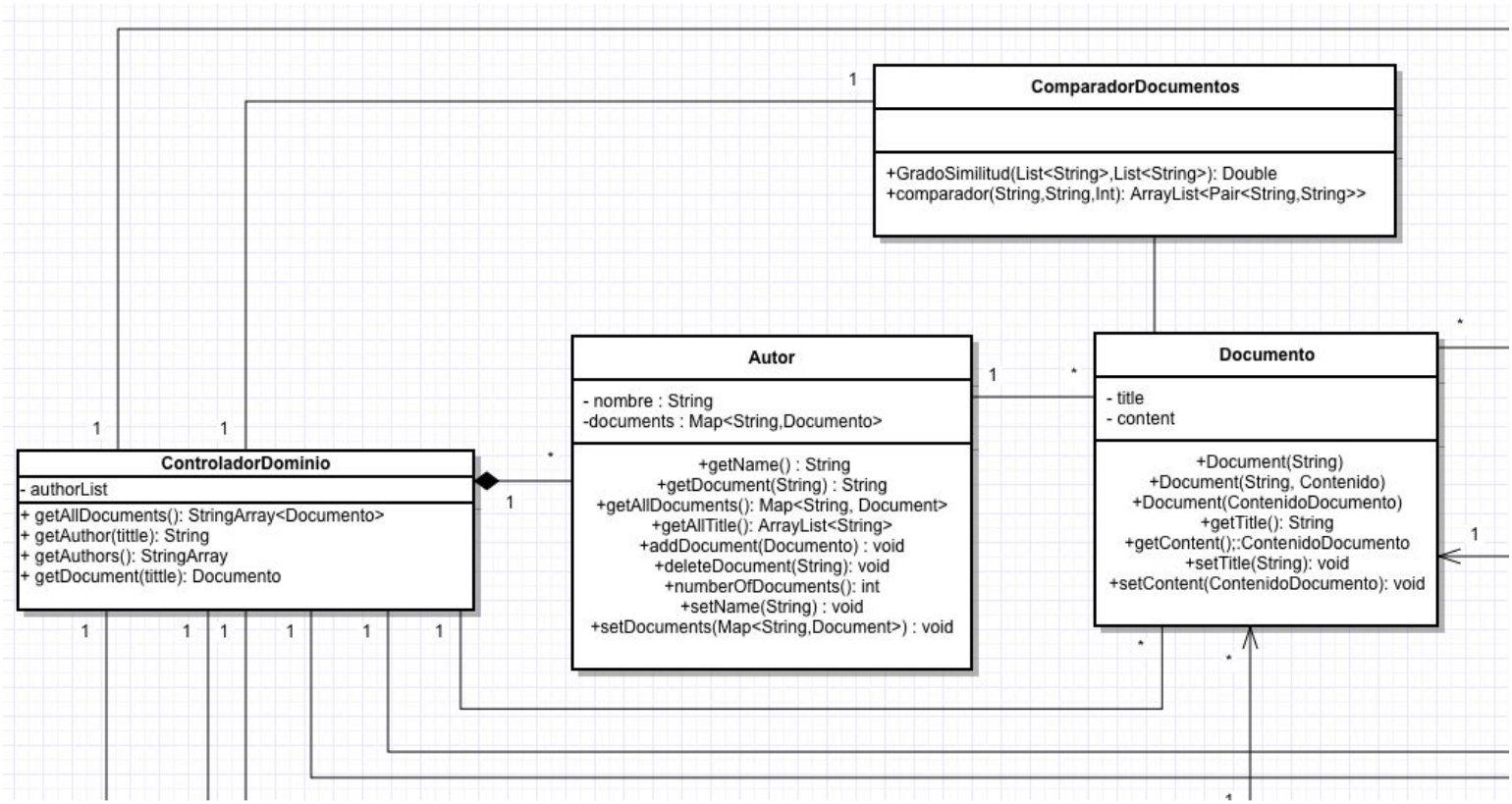
- Consultar contenido de documento: Sirve para que el usuario pueda obtener el contenido de documento dado su título y su autor.
- Consultar Lista de Documentos más parecidos a un documento concreto: Dado un documento T y un natural k el usuario puede obtener una lista de los k documentos más parecido a T.
- Consultar Lista de Documentos según expresión booleana: Dado una expresión booleana de palabras el usuario puede obtener una lista de los documentos que cumplen dicha expresión.
- Gestionar Altas de Documentos: Permite al administrador poder subir un documento con su respectivo autor, título y autor a la base de datos.
- Gestionar Bajas de Documentos: Permite al administrador poder eliminar un documento de la base de datos.
- Gestionar Modificaciones de Documentos: Permite al administrador por modificar un título, contenido y/o autor de un documento.

2. Modelo conceptual de datos

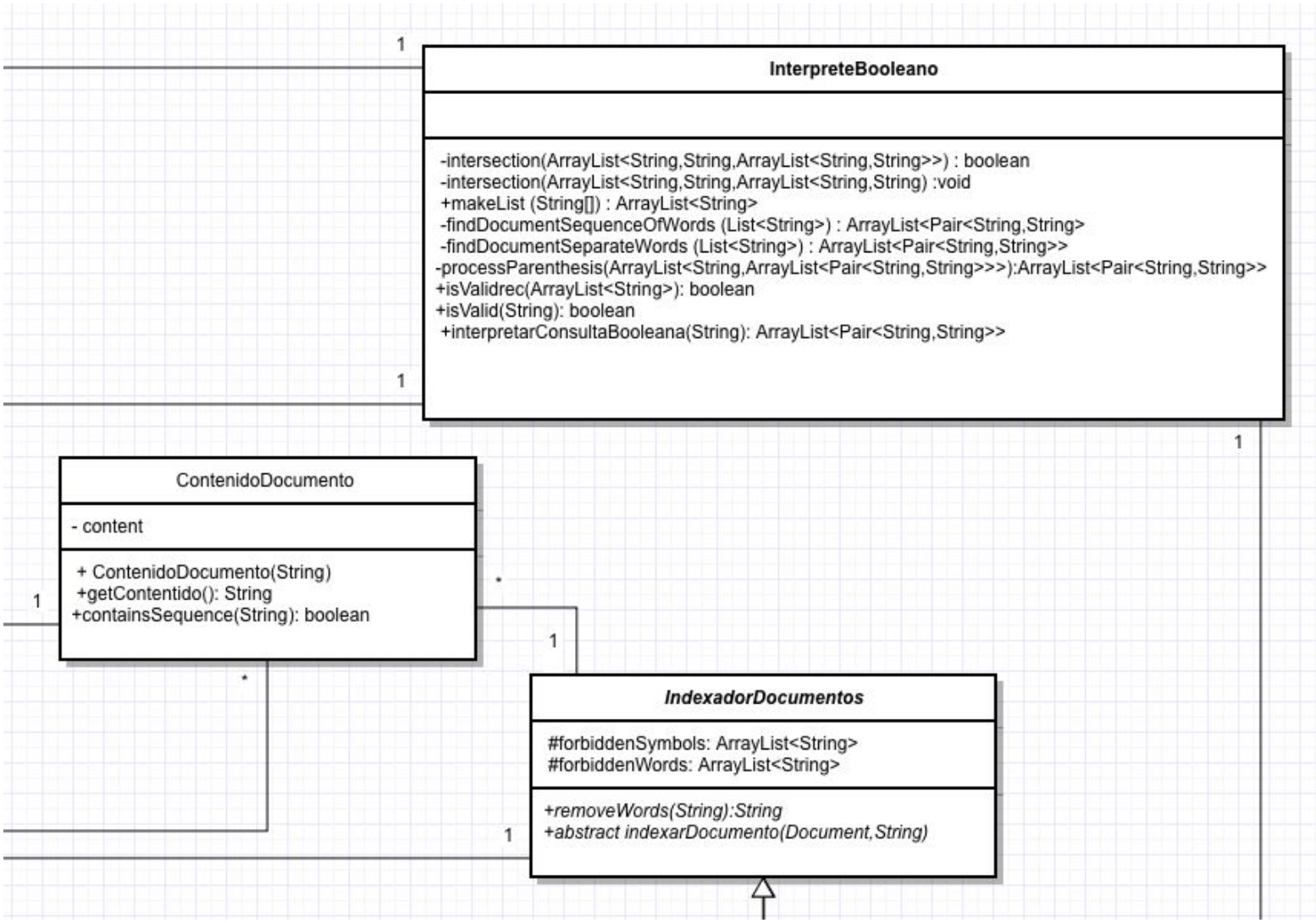
a. Esquema completo



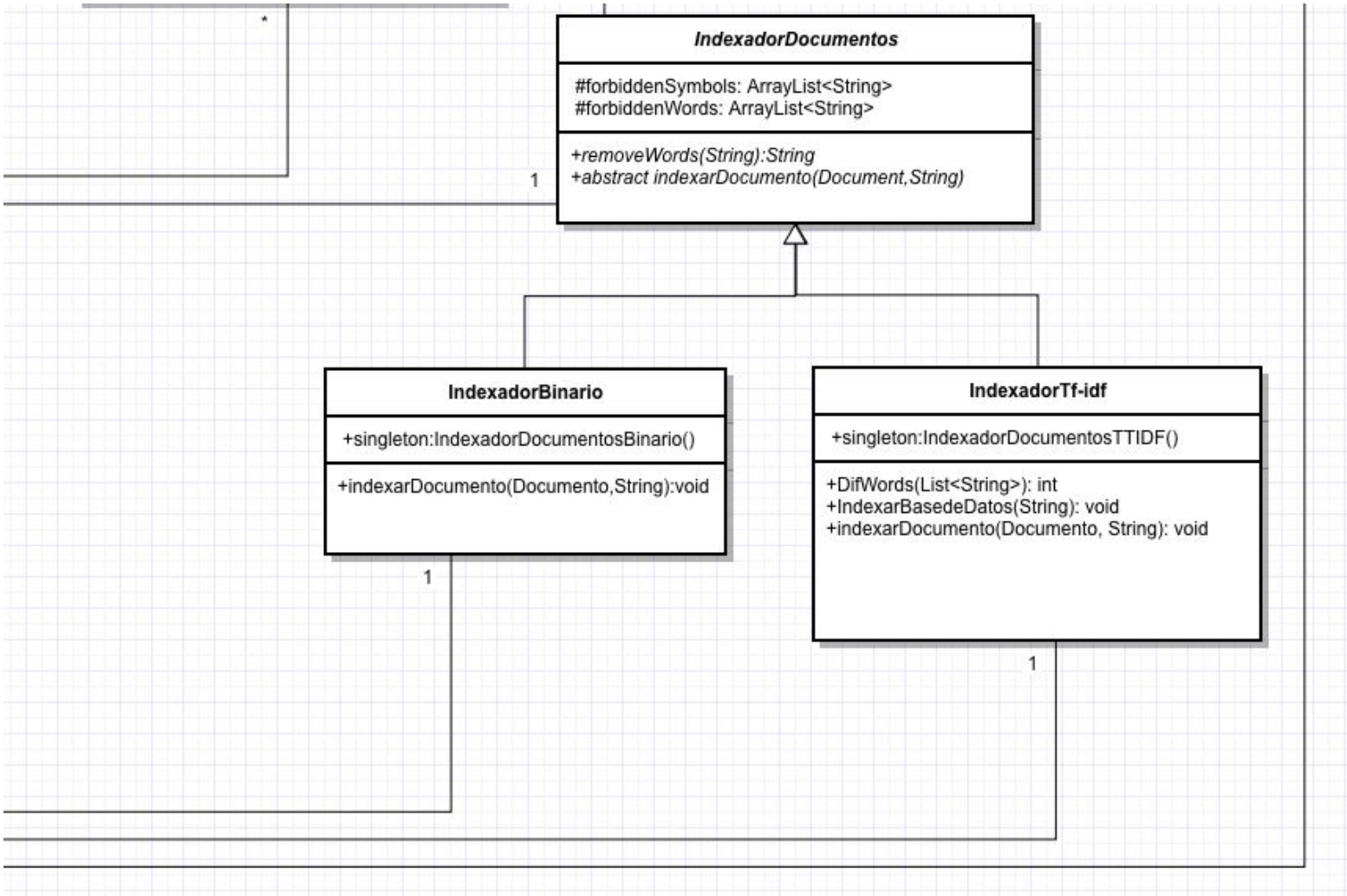
i.Detalle Esquema 1



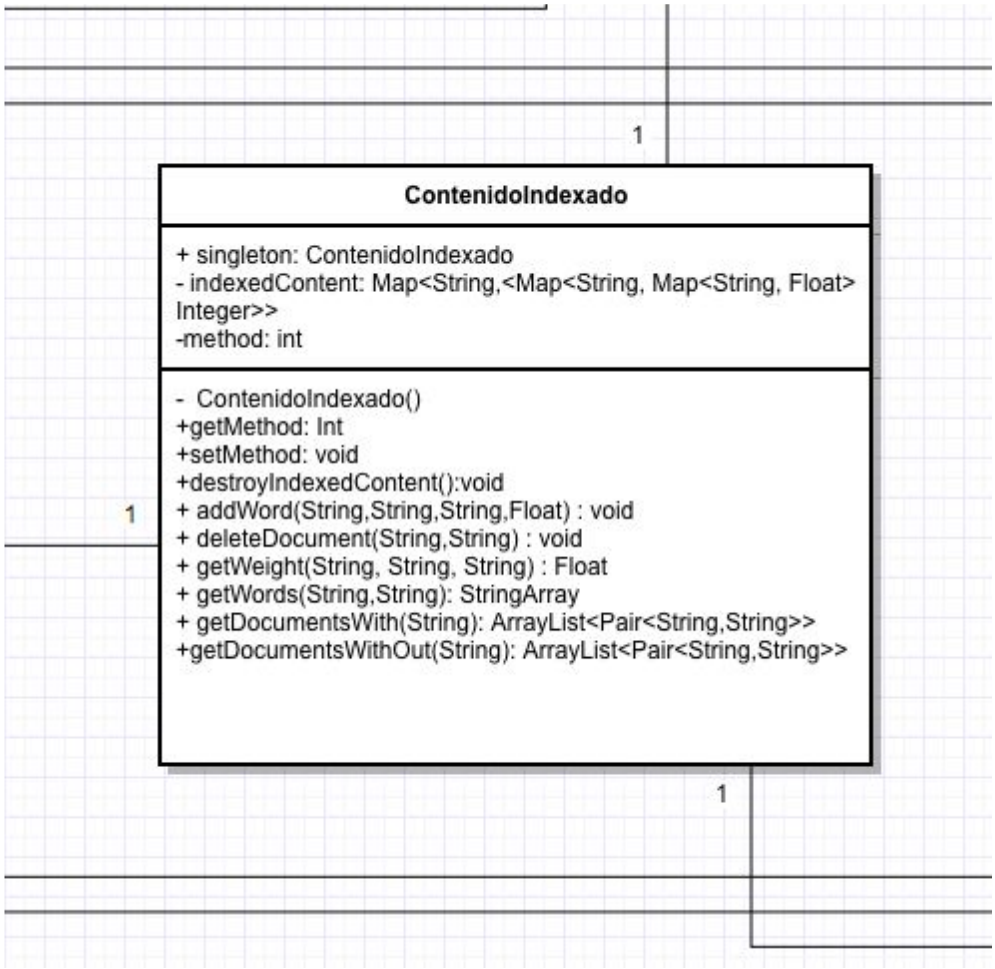
ii.Detalle Esquema 2



ii.Detalle Esquema 3



ii.Detalle Esquema 4



b. Especificación detallada

A. Controlador Dominio

a. Descripción: És la clase de dominio del programa, contiene las funciones que gestionan el programa en sí.

b. Atributos:

i. singleton

1. És la instancia singleton de la clase ControladorDominio
2. Estática

ii. Autores

1. HashMap de todos los autores
2. Ejemplo: “David, Marc Igarza, Marina”

B. Autor
    a. Descripción: És la clase que gestiona el autor de un documento, tanto los títulos de sus obras como sus contenidos.
    b. Atributos:
        i. nombre
            1. Ejemplo: “David” , “Marc Igarza”

        ii. documents
            1. Ejemplo: “Título1, Título2, Título3”

C. Documento
    a. Descripción: És la clase que gestiona un documento, pasando por su título y su contenido.
    b. Atributos:
        i. title
            1. Ejemplo: “Título1”,”Título2”, “Título3”
        ii. Content
            1. Ejemplo: “Contenido1”,”Contenido2”, “Contenido3”

D. Comparador Documento
    a. Descripción: Esta clase tiene la función de comparar un documento dado a los otros documentos de la base de datos y devolver el/los mas similar/es
    b. Atributos:
        i. singleton
            1. És la singleton de la clase ComparadorDocumentos
            2. Estática

E. Contenido Indexado
    a. Descripción: Esta clase realiza la tarea de almacenar y gestionar el contenido indexado de cada unos de los documentos de la base de datos.
    b. Atributos:

i. singleton
    1. És la singleton de la clase ContenidoIndexado
    2. Estática
ii. method
    1. Indica el que metodo se está usando
    2. Ejemplo: “1”,”0”,”2”
    3. Estática
iii. indexedContent
    1. És la estructura de datos que almacena el contenido indexado
    2. Ejemplo: “”Palabra1”,(“David”,(“Título1”, “2.556”))”
    3. Estática

F. Contenido Documento
    a. Descripción: És la clase que gestiona el contenido de un documento, desde su almacenamiento a su modificación.
    b. Atributos:
        i. content
            1. Ejemplo: “ContenidoDocumento1”

G. Intérprete Booleano
    a. Descripción: Esta clase tiene la función de interpretar un expresión booleana y buscar los documentos que cumplen tal expresión.
    b. Atributos: -

H. Indexador Documentos
    a. Descripción:
    b. Atributos:
        i. forbiddenSymbols
            1. Representa los símbolos que queremos eliminar al tratar el contenido del documento
            2. Ejemplo: “,”,”!”,”?”
            3. Estática

ii. forbiddenWords

1. Representa las palabras que queremos eliminar al tratar el contenido del documento
2. Ejempo: “la”,”mucho”
3. Estática

I. Indexador Documentos Binario

a. Descripción: És una clase que extiende a IndexadorDocumentos que tiene la función de transformar un contenido de una documento(String) en un contenido indexador(Palabra,Peso) mediante un método de indexación binario.

b. Atributos:

i. singleton

1. És la singleton de la clase IndexadorDocumentosBinario
2. Éstatica

J. Indexador Documentos TF-IDF

a. Descripción: És una clase que extiende a IndexadorDocumentos que tiene la función de transformar un contenido de una documento(String) en un contenido indexador(Palabra,Peso) mediante un método de indexación TF-IDF.

b. Atributos:

i. singleton

1. És la singleton de la clase IndexadorDocumentosTFIDF
2. Éstatica

3. Breve descripción de las Estructuras de Datos y algoritmos utilizados para implementar las funcionalidades principales

a. Map<String, Map<String, Map<String, Float>>>

Usada para representar el contenido indexado de la base de datos, el primer String representa la palabra del documento, el segundo String representa el autor del documento y el tercer String representa el título del documento, finalmente el último valor(Float) representa el peso de la palabra en el documento. Se ha usado este tipo de estructura de datos porque la mayoría de accesos serán consultas y altas, debido a su eficiencia respecto a este tipo de accesos. En el caso de las eliminaciones, esta estructura de datos es ineficiente, pero como las eliminaciones seràn poco frecuentes hemos considerado que esta estructura seguirá siendo más eficiente que otras.

b. Pair<Double,Pair<String,String> >

Se ha usado esta estructura en la función comparar(...) de ComparadorDocumentos para poder gestionar el grado de similitud y los datos de autor y título.

c. ArrayList<Pair<String, ArrayList<Pair<String, String>>>>

Esta estructura de datos se usa en la clase InterpreteBooleano, específicamente en la función processParenthesis. Se creó esta estructura de datos para que fuese más fácil aplicar la unión y intersección de dos conjuntos, y que se pudieran hacer llamadas recursivas de forma más sencilla. El primer elemento del pair de dentro del array contiene un string con un valor que dice que contiene la ArrayList (una secuencia de autores y títulos de documentos o un operador lógico). El segundo elemento del pair es vacío en caso de que fuese un operador lógico, o contiene la lista de pares de autores y documentos que satisfacen una condición.

d. IndexadorDocumentosTFIDF.IndexarDocumento(Documento,String)

Esta función realiza una indexación a un contenido indexado mediante el algoritmo TF-IDF, muy utilizado actualmente gracias a su simplicidad y eficacia.

e. ComparadorDocumentos.GradoSimiltud(List<String>,List<String>)

Este algoritmo tiene la función de comparar dos lista de string y devolver su grado de similitud, la fórmula que el algoritmo intenta imitar es

$$g = PalabrasComunes/PalabrasTotales$$

f. InterpreteBooleano.processParenthesis(ArrayList<Pair<String, ArrayList<Pair<String,String>>>>)

Procesa la consulta booleana dando prioridad a aquello que esté dentro de paréntesis y va llamando recursivamente para procesar aquello que hay dentro de un paréntesis. Esta función devuelve la ArrayList del par autor y documento que satisface una consulta booleana.

4. Relación/lista de las clases implementadas por cada miembro del grupo

a. Marc Igarza:
   i. ComparadorDocumentos
   ii. Documento
   iii. IndexadorDocumentosTFIDF

b. David George Williams:
   i. Autor
   ii. InterpreteBooleano
   iii. IndexadorDocumentosBinario

c. Marina Sauca:
   i. ContenidoDocumento
   ii. ContenidoIndexado
   iii. ControladorDominio
   iv. IndexadorDocumentos

5. Relación de librerías externas utilizadas.

El estándar de Java no posee una clase Pair, pero esta está implementada por Oracle en la librería com.sun.tools.javac.util. Dado que la librería com.sun.* no está disponible de forma trivial usando la compilación mediante javac, hemos optado por copiar la clase Pair en nuestro código. La autoría de Oracle está convenientemente referenciada.