



ugr

Universidad  
de Granada

TRABAJO FIN DE GRADO  
INGENIERÍA EN INFORMÁTICA

# SulalR-L

---

Una herramienta para la enseñanza y aprendizaje de la recuperación de  
información basada en Lucene

**Autor**

José Antonio Medina García

**Tutor**

Juan Manuel Fernández Luna



Escuela Técnica Superior de Ingenierías Informática y  
de Telecomunicación

Granada, mayo de 2017









ugr

Universidad  
de Granada

# SulalR-L

---

**Una herramienta para la enseñanza y aprendizaje de la recuperación de información basada en Lucene.**

**Autor**

José Antonio Medina García

**Tutor**

Juan Manuel Fernández Luna



# SulalR-L: Una herramienta para la enseñanza y aprendizaje de la recuperación de información basada en Lucene

José Antonio Medina García

**Palabras clave:** Recuperación de información, Lucene

## Resumen

El desarrollo de este TFG (Trabajo Fin de Grado) tiene como finalidad mostrar a los alumnos como se realiza el proceso de creación de un sistema de recuperación de información a partir de una colección de documentos, como es el estado del índice invertido una vez creado y como es el procesamiento de las consultas utilizando para ello la API de Lucene para JAVA.

Para alcanzar tal fin se pretende crear una aplicación en JAVA que permita cargar colecciones de documentos en castellano o inglés, extraer su texto, procesarlo para después crear el índice y poder realizar consultas de tres tipos más importantes: Vectorial, booleana y probabilística.

La primera fase es la fase de procesamiento y que estará formada por tres procedimientos que transforman el texto de los documentos. Son los siguientes:

Una fase de *tokenización* dónde se toma los documentos originales de la colección y se “parsean” para así obtener cada uno de los términos o tokens y se eliminan elementos como son los signos de puntuación o algunas palabras sin sentido como algunas preposiciones o conjunciones.

La segunda fase y que será opcional en este proyecto es la *eliminación de palabras vacías*. Las palabras vacías son términos que son un pobre indicador de contenido por su alto número de apariciones y por tanto se pueden quitar.

La tercera fase y última es la fase de *stemming* o de *lematización*. En esta fase es dependiente del idioma y trata de agrupar palabras utilizando su forma base. De esta forma se consigue sustituir los términos de una colección por los conceptos subyacentes en los términos de la colección.

La siguiente fase del proceso en la construcción de un sistema de recuperación de información es la creación de un índice con el que podamos extraer los documentos que cumplen los requisitos de una consulta introducida por un usuario. El índice será un índice invertido, es decir, a partir de un término sabremos en qué documentos se encuentra.

La última parte de este proyecto será la construcción de un sistema que, a partir de una consulta, nos muestre los documentos que mejor se adapten a ella. Para ello utilizaremos tres modelos de búsqueda: el modelo vectorial, el modelo probabilístico y el modelo booleano.

El primero de ellos será implementado utilizando la similitud coseno, el segundo vendrá dado por la implementación del modelo BM25 y el tercero por el modelo booleano, todos ellos con la implementación que ofrece Lucene.





## **SulalR-L: Una herramienta para la enseñanza y aprendizaje de la recuperación de información basada en Lucene**

José Antonio Medina García

**Keywords:** Information retrieval, Lucene,

### **Abstract**

The development of this final grade Project aims to show to students how the process of creating a information retrieval system from a collection of documents is, how the inverted index and how the queries are processed the Lucene API for Java is.

In order to achieve this goal, it aims to create an application in JAVA that allows to load collections of documents in Spanish or English, to extract its text, to process it to create the index and to be able to make queries of the three most important types: Vectorial, Boolean and probabilistic.

The first part is the processing stage. It is formed by three steps which transforms the documents text. They are the following:

A first sub-step of tokenization where it takes the original documents of the collection and are parsed to get all the terms or tokens. This process removes some terms as punctuation marks, prepositions or conjunctions.

A optional second sub-step is to remove the stopwords. The stopwords are terms which are worthless as index terms and it can be removed.

The third and last phase is the stemming. This phase is language dependent and tries to group words using its base form. In this way it is possible to substitute the terms of a collection for the concepts underlying the terms of the collection.

The next process stage is the creation of an index with which we can retrieve documents that meet the requirements of a query entered by a user. The index will be an inverted index, that is, from a term we will know what documents where it occurs.

The last step will be to build a system which retrieves documents from a query. For it we will use three models: the Vector Space Model (VSM), that use the cosine similarity, a Probabilistic Model (BM25) and the Boolean Model. All they are implemented by Lucene.



---

Yo, **José Antonio Medina García**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75482863-Z, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo.: José Antonio Medina García

Granada a 02 de mayo de 2017.



---

D. **Juan Manuel Fernández Luna** Profesor del área de Ciencias de la Computación e Inteligencia Artificial del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

**Informa:**

Que el presente trabajo, titulado **SulalR-L: Una herramienta para la enseñanza y aprendizaje de la recuperación de información basada en Lucene** ha sido realizado bajo su supervisión por **José Antonio Medina García**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 2017.

**El tutor:**

**Juan Manuel Fernández Luna**



# Agradecimientos

A mis padres, por todo lo que me han dado en mi vida y haber permitido que haya podido llegar hasta aquí a base de mucho sacrificio. Por sus consejos y sus ánimos en todos estos años.

A mi hermano Alejandro que es todo un ejemplo de esfuerzo, de constancia y de lucha para alcanzar todas las metas que uno se propone.

A mi novia Lucía, por haberme acompañado durante todos estos años por disfrutar conmigo en los buenos momentos y ser apoyo y ánimos en los menos buenos.

A mis abuelos paternos, que allá donde estén espero que estén orgullosos. Y a mis abuelos maternos, que has sido testigos de toda esta etapa y espero que de muchas más.

A toda mi familia.

A Juan Manuel Fernández Luna, que en todo momento ha estado disponible para cualquier cosa que he necesitado. Enhorabuena por tu forma de enseñar, de tratar a tus alumnos. Gracias por todo.





## Contenido

Resumen .....	7
Abstract.....	10
Agradecimientos .....	16
1. Introducción .....	21
1.1. Objetivos .....	21
2. Qué es un sistema de recuperación de información.....	23
2.1. Información frente a recuperación de datos.....	23
2.2. El porqué de un sistema de recuperación de información .....	23
2.3. Los grandes problemas .....	24
2.4. Como funciona un sistema de recuperación de información.....	25
2.5. Pasos en la creación de un sistema de recuperación .....	26
2.5.1. Tokenización.....	27
2.5.2. Palabras vacías .....	28
2.5.3. Lematización (Stemming) .....	29
2.5.4. Índice invertido.....	29
2.5.5. Consultas.....	29
2.5.6. Modelos de recuperación .....	30
3. Fases del trabajo .....	33
3.1. Enunciado del problema.....	33
3.2. Fase de análisis .....	33
3.3. Selección del lenguaje de programación .....	33
3.4. Estudio de los recursos necesarios .....	34
3.5. Fase de desarrollo.....	34
3.6. Planificación del proyecto .....	34
4. Ingeniería del software.....	35
4.1. Ciclo de vida.....	35
4.2. Objetivos del sistema .....	36
4.3. Requisitos funcionales.....	36
4.4. Requisitos no funcionales.....	37
Implementación .....	37
Interfaz .....	37
Soporte (Facilidad de mantenimiento y portabilidad) .....	37
Rendimiento .....	37
4.5. Casos de uso .....	37
4.5.1 Descripción de los actores.....	37
4.5.2 Diagramas de casos de uso.....	38
4.6. Diagramas de secuencia .....	50
5. Diseño .....	57

5.1. Diagrama de clases.....	57
5.2. Diseño de la interfaz.....	59
5.3. Prototipos de pantalla.....	60
6. Desarrollo .....	66
6.1. Herramientas utilizadas .....	66
6.1.1. Lenguaje utilizado .....	66
6.1.2. Entorno de programación.....	66
6.1.3. Apache Tika [1].....	67
6.1.4. JFreeChart [2].....	67
6.1.5. Apache Lucene [3] [4] .....	67
6.1.6. Luke [5].....	69
6.2. Implementación .....	70
6.2.1. Carga de archivos.....	70
6.2.2. Procesamiento de documentos.....	71
6.2.3. Creación del índice .....	76
6.2.3. Tratamiento de consultas.....	78
7. Manual de uso de la aplicación .....	80
7.1. Pantalla inicial del programa.....	80
7.2. Pantalla de configuración de SulaIR-L.....	80
Creación de una nueva colección .....	80
Carga de una colección existente .....	81
Eliminar una colección del sistema .....	81
7.3. Pantalla del proceso de tokenización .....	81
7.4. Pantalla de palabras vacías.....	82
7.5. Pantalla de lematización.....	83
7.6. Pantalla del índice .....	84
7.7. Pantalla de gráficas.....	85
7.8. Pantalla de búsqueda vectorial.....	86
7.9. Pantalla de búsqueda booleana .....	87
7.10. Pantalla de búsqueda probabilística .....	88
7.11. Pantalla de visor de documentos.....	90
7.12. Pantalla de ayuda.....	90
7.13. Pantalla de acerca de... ..	91
8. Conclusiones finales y trabajo futuro .....	92
8.1. Desarrollo temporal real .....	92
8.2. Conclusiones.....	92
8.3. Trabajo futuro.....	93
8.4. Valoración personal.....	94
9. Anexo .....	95

9.1. Código fuente de la aplicación.....	95
9.2. Descarga de la aplicación y ayuda .....	95
10. Bibliografía.....	96

# 1. Introducción

El crecimiento de la información digital hoy en día supera todas las previsiones hechas y por hacer. Tal crecimiento requiere que esa información sea accedida casi al instante cuando el usuario la solicite. Por eso se crean los sistemas de recuperación de información. Sistemas que sean capaces de ofrecer al usuario, en un periodo ínfimo de tiempo, toda la información al usuario de la manera más precisa posible.

Al principio se pensaba en que las bases de datos acabarían con los problemas de la búsqueda de información gracias a su estructura, fácil manejo y su forma de extraer la información, pero a pesar de solucionar algunos de esos problemas esto requería reducir a un conjunto de caracteres parte de la realidad y con una semántica exacta.

Esto fue lo que causó a que se empezara a desarrollar otros tipos de sistemas, sistemas que se fuesen acercando a cubrir las necesidades de obtener información a los usuarios de un ordenador, sistemas que sin tener que utilizar un lenguaje técnico fuese capaz de “comprender” lo que el usuario quiere. Estos son los sistemas de recuperación de información.

La recuperación de información está extendida por numerosas áreas como puede ser la inteligencia artificial, informática, computación paralela, sistemas multimedia... Esto hace que no sea extensamente conocida por la comunidad de las ciencias de la computación. Suelen confundirse con bases de datos, con las que comparte intereses, pero de la cual es diferente.

Para crear un sistema de recuperación de información es necesario realizar un conjunto de pasos con los documentos de los que se desea extraer la información, utilizar las estructuras de datos adecuadas para almacenar esa información y que nos permita acceder a ellos de forma rápida y sencilla.

Existen numerosos libros en los que se cuenta cómo es el proceso para obtener un SRI<sup>1</sup> pero no existen muchas aplicaciones que muestren paso a paso como crea dicho sistema y de ahí surge la idea de crear este trabajo fin de grado.

El motivo de llamar a esta aplicación SulaIR-L viene de un juego de palabras entre “Yabal **Sulayr**”, nombre con el que los árabes llamaban a Sierra Nevada y que significaba “montaña del sol”<sup>2</sup>, e “**I**nformation **R**etrieval” de recuperación de información. La “L” final hace referencia a que el índice y la recuperación de los documentos se realiza con **Lucene**, una API para desarrollar sistemas de recuperación de información.

## 1.1. Objetivos

El objetivo de este trabajo fin de grado es el de crear una aplicación que muestre paso a paso como se realiza un sistema de recuperación de información. Para ello se nos pide como requisito trabajar con la API de Lucene.

La aplicación deberá de permitir la carga de documentos de los cuales se puede extraer texto (txt, XML, HTML, PDF y doc) y poder ver paso a paso como va realizándose los procesos de tokenización, la eliminación de las palabras vacías y por último la lematización (stemming).

---

<sup>1</sup> Sistema de Recuperación de Información

<sup>2</sup> <http://masdehistoria.blogspot.com.es/2010/01/sierra-nevada-en-la-historia-el-origen.html>

Una vez obtenido el texto con los lemas correspondientes se visualizará el índice obtenido usando para su creación la API de Lucene y, además, se podrá ver algunos datos sobre los términos en la colección.

Por último, la aplicación deberá de permitir la formulación de consultas para la recuperación de documentos en los tres modelos más utilizados en los sistemas de recuperación de información, como son: el modelo de espacio vectorial, el modelo booleano y el modelo probabilístico BM25.

Además, también se pide que la aplicación sea fácil de usar, agradable a la vista y que permita al usuario comprender el proceso de recuperación de información de manera clara e intuitiva.

## 2. Qué es un sistema de recuperación de información<sup>3</sup>

La recuperación de información se ocupa de representar, almacenar, organizar y dar acceso a elementos de información. La representación y organización de la información debe proporcionar al usuario un acceso sencillo a la información en la cual está interesado.

### 2.1. Información frente a recuperación de datos

La recuperación de datos consiste en determinar los documentos que contienen las palabras de la consulta que el usuario realiza pero que, en la mayoría de los casos no satisfacen la necesidad de información del usuario. Un usuario de un sistema de recuperación de información está más interesado en recuperar información sobre un asunto que recuperar datos sobre una consulta. El objetivo de un sistema de recuperación de datos es el de recuperar toda la información almacenada sobre una consulta utilizando para ello expresiones regulares o álgebra relación, mostrando al usuario gran cantidad de documentos que no cumplen el fin de su consulta. Un sistema de recuperación de información podría subsanar ese inconveniente ya que puede recuperar documentos mediante un lenguaje natural no siempre bien estructurado gracias a que tiene una semántica y estructura bien definida.

En la recuperación de datos, al proporcionar información al usuario de una base de datos, no soluciona el problema de la recuperación de información sobre un tema o asunto, ya que la consulta debe de ser escrita de manera exacta. Sin embargo, el sistema de recuperación de información debe, de algún modo, interpretar el contenido de los documentos y mostrarlos al usuario acorde al grado de relevancia respecto a la consulta del realizada. De hecho, el primer objetivo de un sistema de recuperación de información es recuperar todos aquellos documentos que son relevantes para la consulta del usuario.

### 2.2. El porqué de un sistema de recuperación de información

En los últimos años la recuperación de información ha crecido más allá de recuperar información e indexar texto. Con la aparición de la web cambió esa percepción que se tenía sobre la recuperación de información.

La web hoy en día es una fuente infinita de conocimiento y cultura. Hoy en día se puede compartir información e ideas a una escala nunca vista y cada día es mayor. A pesar de todo esto, se ha producido el gran problema de encontrar información útil ya que lo ha convertido en una tarea tediosa y dificultosa. Por ejemplo, imaginemos que queremos buscar información sobre un tema y tenemos que ir navegando enlace por enlace por todo el ciberespacio, tardaríamos muchísimo tiempo y quizás nunca encontraríamos lo que buscamos. Estas dificultades son las que han traído a que se creen los sistemas de recuperación de información con nuevas técnicas a los que existían antiguamente.

---

<sup>3</sup> Parte del texto de este apartado está extraído y traducido de los libros [11] y [12]

## 2.3. Los grandes problemas

Los esfuerzos en la investigación de los motores de los sistemas de recuperación de información se han centrado en algunas cuestiones que hoy en día sigue siendo importantes. Uno de esos temas es la relevancia.

La relevancia es un concepto fundamental en la recuperación de información. Un documento es relevante si contiene la información que una persona está buscando cuando ejecuta una consulta en un motor de búsqueda. Hay muchos factores que llevan a que un documento sea relevante o no para un usuario. Esos factores deben ser tomados en cuenta cuando diseñamos un algoritmo para comparar texto y establecer un ranking de documentos. Simplemente con comparar el texto de una consulta con el texto de un documento y buscar una coincidencia exacta produciría unos resultados muy pobres en términos de relevancia.

Es importante también diferenciar entre relevancia tópica y relevancia del usuario. Un texto es tópicamente relevante a una consulta si es del mismo tema. Sin embargo, la relevancia del usuario es la relevancia del documento respecto a la necesidad de información original por parte del usuario. Es decir, de los documentos que son recuperados, lo que el usuario considera que son relevantes.

Para abordar el problema de la relevancia, los investigadores proponen modelos de recuperación de información y probar cómo se comportan. Un modelo de recuperación de información es una representación de un proceso que hace coincidir una consulta y un documento. Son la base para establecer un algoritmo de ranking que son usados en motores de búsqueda para obtener los documentos ordenados por ese ranking. Un buen modelo de recuperación encontrará los documentos que son más acordes a ser considerados relevantes para el usuario que para la consulta.

Otro problema para la recuperación de información es la evaluación. Dado que la calidad de la clasificación de documentos depende de lo bien que coincida con las expectativas de una persona, es necesario desarrollar medidas de evaluación y procedimientos experimentales para adquirir estos datos y utilizarlos para comparar con algoritmos de clasificación. Estas medidas son la precisión y el recall.

La precisión consiste en la proporción de documentos recuperados que son relevantes. El recall es la proporción de documentos relevantes que son recuperados. Cuando se utiliza la medida de recall se asume que todos los documentos relevantes son conocidos. Esta medida puede ser algo errónea en colecciones grandes tales como motores de búsqueda web, pero puede ser útil cuando se trate de colecciones pequeñas.

Alguna de las evaluaciones en motores de búsqueda y modelos de recuperación es teniendo en cuenta los clicks que se realizan en los resultados que se recuperan al hacer una consulta. Aquellos documentos que más clicks tienen obtendrán una mejor puntuación a la hora de realizar una nueva consulta.

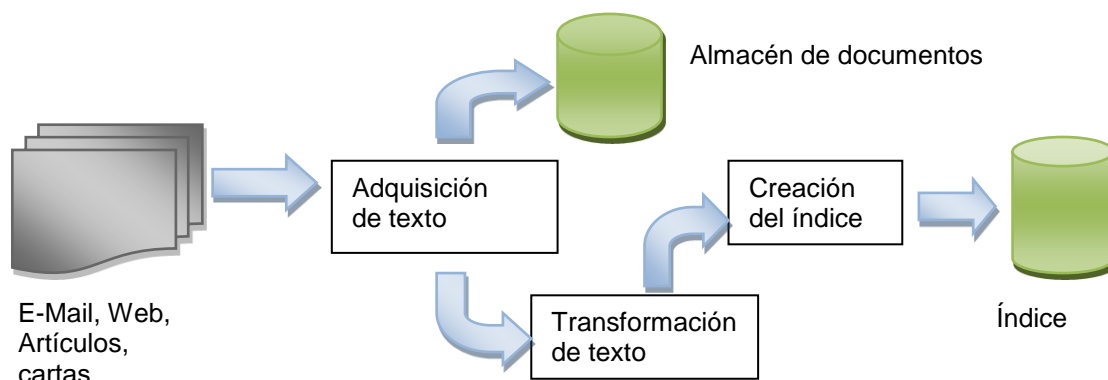
Otro de los problemas a abordar en la recuperación de información es el énfasis en los usuarios y la información que necesitan. Los usuarios son los que van a evaluar la calidad de un motor de búsqueda. Esto ha llevado a numerosos estudios sobre cómo las personas interactúan con los motores de búsqueda y al desarrollo de técnicas para ayudarles a realizar sus consultas.



## 2.4. Como funciona un sistema de recuperación de información

Los sistemas de recuperación tienen dos tareas principales que son el proceso de indexación y el de consulta. El proceso de indexación construye las estructuras que harán posible la búsqueda y el proceso de consulta utiliza esas estructuras y una consulta proporcionada por el usuario para recuperar una lista de documentos ordenados por un ranking.

El proceso de indexación ([Ilustración 1](#)) está formado por tres importantes procedimientos: La adquisición del texto, la transformación del mismo para obtener un índice lo más eficiente posible, y la creación del índice.



*Ilustración 1: Proceso de indexación*

La tarea del proceso de adquisición de texto es la de identificar y hacer disponible los documentos que serán buscados. Aunque en algunos casos esto puede implicar el uso de una colección existente, la adquisición de texto requiere construir una colección a través de crawling o escaneo web, intranet u otras fuentes de información. Además, cuando se pasa el documento a la siguiente etapa, el proceso de adquisición de texto crea un almacén de directorios en el cual se almacena el texto y los metadatos del fichero.

El proceso de transformación de texto transforma el documento en términos del índice. Estos términos son parte del documento que son almacenados en el índice y son usado en las búsquedas. La palabra es la unidad más pequeña del índice, pero no todas las palabras servirán a la hora de hacer la búsqueda. Para obtener las palabras del índice, en la transformación del texto se realiza la tokenización de los documentos, la eliminación de las palabras vacías (stopwords) y la lematización.

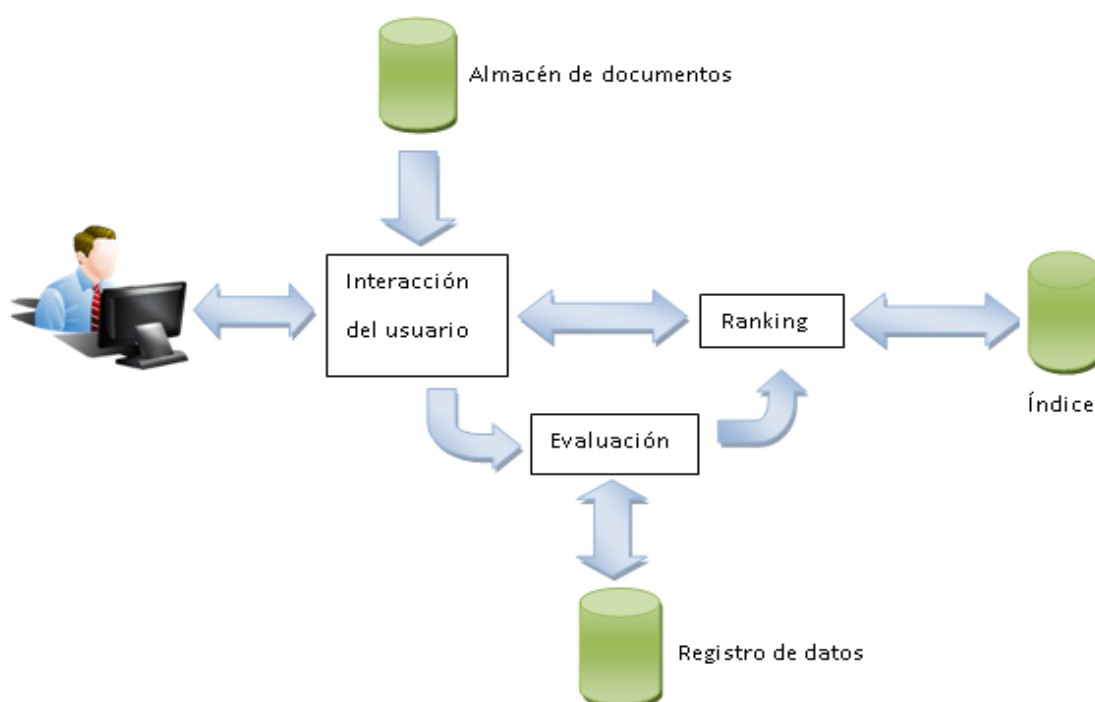
El proceso de creación del índice toma la salida de la etapa anterior y crea el índice o las estructuras de datos que permiten una búsqueda rápida. Dado el gran número de documentos in muchas aplicaciones de búsqueda, la creación del índice debe ser eficiente, tanto en tiempo como en espacio. También tienen que ser eficiente la actualización de nuevos documentos cuando estos sean adquiridos.

El índice más común en los sistemas de recuperación de información es el índice invertido. Un índice invertido contiene una lista, para cada término, de los documentos que lo contiene. Existen muchos tipos de índices invertidos y la particularidad de cada uno es uno de los aspectos más importantes en los motores de búsqueda.

El proceso de consulta ([Ilustración 2](#)) está formado por los siguientes componentes: La interacción del usuario, el ranking y la evaluación.

La interfaz gráfica proporciona el enlace entre el usuario que realiza la consulta y el motor de búsqueda. Una de las tareas de este componente es transformar la consulta del usuario en términos del índice. Otra tarea es la de listar los documentos recuperados por el motor de búsqueda según un ranking y mostrarlos de manera organizada al usuario. El almacén de documentos es una de las fuentes de información usadas en la generación de resultados ya que contiene los documentos a recuperar.

El ranking es el núcleo del motor de búsqueda. Toma la consulta transformada procedente de la interacción del usuario y genera una lista ordenada de los documentos usando un ranking basado en un modelo de recuperación. Un ranking debe ser eficiente y efectivo. Eficiente en el aspecto de encontrar los resultados en un leve periodo de tiempo, y efectivo, ya que la calidad del motor de búsqueda va ser determinada por el número de documentos relevantes que se recuperen. La eficiencia del ranking depende del índice mientras que la eficacia vendrá dada por el modelo de recuperación.



*Ilustración 2: Proceso de consulta*

La tarea de evaluación es una medida y monitoriza la efectividad y la eficacia. Una importante tarea de este componente es guardar y analizar el comportamiento de los usuarios en un registro de datos. Los resultados de la evaluación son usados para mejorar el ranking. Es principalmente una actividad que se realiza offline, pero es crítica en cualquier aplicación de búsqueda.

## 2.5. Pasos en la creación de un sistema de recuperación

La adquisición de texto es la tarea fundamental de un sistema recuperación de información. Sin texto no tenemos nada qué indexar y qué buscar.

Una vez tenemos el texto lo que queremos es buscar. El paso siguiente es la transformación del texto o procesamiento del texto. El objetivo es convertir las muchas formas en las que puede aparecer una palabra en términos de índice más consistentes. Los términos del índice son la representación del contenido de un documento que es usado para la búsqueda.

La búsqueda de una palabra exacta es bastante restrictiva. Por ejemplo, una restricción es la sensibilidad de mayúsculas y minúsculas. Imaginemos que queremos buscar “recuperación de información” y en el documento está como “Recuperación de Información”. Nuestra consulta no nos devolvería el documento deseado ya que nosotros la buscamos en minúscula y el documento la tiene en mayúscula. Por eso una de las cosas que debemos hacer es convertir todo el texto en minúscula, tanto el del documento como el de la consulta. Por eso se realiza una serie de pasos en el procesamiento del texto para hacer luego más fácil la búsqueda de información. Estos pasos son: Dividir las palabras en un proceso llamado tokenización. Después se podría ignorar algunas palabras para hacer el procesamiento de la consulta más eficaz y eficiente en un proceso llamado eliminación de palabra vacías (stopwords) y acto seguido realizar el proceso de lematización (stemming) que permite que palabras similares coincidan entre sí. De todos estos pasos se hablará más adelante.

Esas técnicas son bastante simples, aunque sus efectos en resultados de búsqueda pueden ser profundos. Ningunas de esas técnicas involucran que el ordenador tenga que hacer ningún tipo de razonamiento complejo o comprensión del texto. Comprender la estadística del texto es fundamental para comprender los modelos de recuperación y los algoritmos de ranking. Más sofisticadas son las técnicas de procesamiento de lenguaje natural que involucra análisis semánticos y sintácticos.

Aunque el lenguaje es muy rico y variado, es muy predecible. Hay muchas maneras de describir un tema en particular, pero si contamos las palabras que aparecen en estas descripciones podemos observar que unas palabras ocurren mucho más que otras. Estas estadísticas son muy importantes en la recuperación de información y son usadas en muchos de los componentes del núcleo de un motor de búsqueda como pueden ser algoritmos de ranking, transformaciones de consulta y técnicas de indexación. La ley de Zipf es una ley empírica que afirma que “para un idioma determinado la frecuencia de aparición de distintas palabras sigue una distribución que puede aproximarse a  $P_n \sim 1/n^a$  donde  $P_n$  es la frecuencia de la  $n$ -ésima palabra y el exponente ‘a’ es un real positivo ligeramente cercano a 1.” Por lo que significa que la segunda palabra más frecuente de una colección ocurre aproximadamente la mitad que la primera, la tercera un tercio de la primera, y así sucesivamente. Esta ley nos sirve para comprobar que palabras serán más adecuadas para representar mejor el documento.

### **2.5.1. Tokenización**

El siguiente paso que se debe de realizar en la construcción de un sistema de recuperación de información es el análisis o “parseo” de los documentos. Este proceso involucra el reconocimiento del contenido y estructura del texto de los documentos. El contenido principal de la mayoría de los documentos son las palabras de las que se hablaba y contaba en el párrafo anterior usando la ley de Zipf. Reconocer cada ocurrencia de una palabra en el texto de un documento es lo que se llama tokenización o análisis léxico. Además de palabras también hay otro tipo de contenido en un fichero llamados metadatos. Esto metadatos pueden contener información útil como el autor, fechas o etiquetas que pueden permitirnos identificar un documento con más facilidad. El parser utiliza parte de esos metadatos y etiquetas para identificar la estructura del documento mediante marcas del lenguaje y reproducir una representación textual del contenido. En los motores de búsqueda se utilizará la salida del parser para construir el índice.

La tokenización es el proceso de formar palabras procedentes de una secuencia de caracteres en un documento. En sistemas antiguos, una palabra o término se define como una secuencia de caracteres alfanuméricos de tres o más caracteres terminados

por un espacio o un carácter especial y todos los caracteres en mayúscula pasan a estar en minúscula. Esto generaba un problema ya que se perdía mucha información. Otros problemas que surgían eran, por ejemplo, cómo tratar las palabras compuestas unidas por guiones, caracteres especiales, palabras escritas en minúscula que significan diferente a escribirla con mayúscula, los números, los signos de acentuación...

El proceso de tokenización debe ser simple y flexible, para eso el primer paso de la tokenización tiene que estar enfocado en identificar el marcado o etiquetas en el documento. Esto se puede hacer usando un tokenizador y un parser diseñado para el lenguaje específico utilizado y acomodando la recuperación a los problemas antes descritos. Otro segundo paso es eliminar las marcas o etiquetas que son propias de cada una de las estructuras de los documentos, cómo pueden ser las etiquetas de los documentos HTML.

Dado que casi todo el contenido de un documento puede ser importante para algunas consultas, las reglas de la tokenización deben de convertir la mayoría del texto en tokens que sean aptos para su búsqueda.

Resumiendo, generalmente el proceso de tokenización debe de identificar la estructura del documento y luego identificar las palabras en el texto como cualquier secuencia de caracteres alfanuméricos, terminada por espacios o caracteres especiales, con todo convertido a minúsculas. En muchos casos se añaden reglas adicionales al tokenizador para manejar algunos caracteres especiales para asegurar de que los tokens de la consulta y del documento coinciden.

### **2.5.2. Palabras vacías**

Los idiomas están repletos de palabras que tienes más o menos significado. Los más populares son los determinantes. Estas palabras son unidades gramaticales que permiten o bien limitar el referente potencial de un sintagma nominal, o bien cuantificar este sintagma nominal. Otras palabras también son las preposiciones, que tradicionalmente se ha definido como la parte invariable de la oración que une palabras denotando la relación que tienen entre sí. Estos dos tipos de palabras nos lleva a tratarlas de una manera especial en el procesamiento del texto. Primero, porque son dos tipos de palabras muy comunes y segundo, porque esas palabras raramente dan algo de relevancia al contenido del documento. Si estamos considerando palabras individuales en el proceso de recuperación y no frases la función de esas palabras no nos sirven de mucho.

En recuperación de información ese tipo de palabras son llamadas palabras vacías (stopwords). Quitando esas palabras conseguimos disminuir el tamaño del índice e incrementar la eficiencia de la recuperación y, generalmente, mejora la eficacia.

Una lista de palabras vacías se debe de hacer con precaución ya que eliminar demasiadas podría interferir en la efectividad de la recuperación de información. Esta lista debe de construirse con las  $n$  palabras más comunes en la colección. Esto nos podría también llevar a quitar palabras que son importantes para las consultas.

Si el espacio de almacenamiento nos lo permite, es mejor conservar todas estas palabras en el índice. Si debemos de quitarlas, las palabras vacías tienen que ser también eliminadas en las consultas.

### **2.5.3. Lematización (Stemming)**

Parte de la expresividad de un idioma viene dado por el enorme número de formas de transmitir una sola idea. Esto puede ser un problema ya que los sistemas de recuperación confían en encontrar coincidencias de palabras para recuperar documentos relevantes. En vez de restringir los emparejamientos a palabras que son iguales, se ha desarrollado técnicas que permiten a un motor de búsqueda emparejar palabras que están semánticamente relacionadas. La lematización o stemming es un componente del procesamiento del texto que captura las relaciones entre diferentes variaciones de una palabra. Más concretamente, reduce las diferentes formas de una palabra a causa de plurales, géneros, tiempos... a su lema o raíz.

En general, utilizar un stemmer o lematizador en una aplicación de búsqueda una pequeña pero notable mejora en la calidad de los resultados.

Hay dos tipos básicos de stemmers: algorítmicos y basados en diccionarios. El algorítmico utiliza un pequeño programa para decidir si dos palabras están relacionadas o no usando una fuente de sufijos de un idioma en particular. Sin embargo, uno basado en diccionarios no usa esta lógica, sino que se basa en diccionarios de términos ya creados en donde estos están almacenados en cada una de sus formas.

### **2.5.4. Índice invertido**

Todos los modernos motores de búsqueda están basados en el índice invertido. Un índice invertido es la equivalencia computacional de los glosarios encontrados en un libro. El glosario del libro está organizado en orden alfabético de términos. Cada término está seguido por una lista de páginas sobre esta palabra.

Similarmente, un índice invertido está organizado por términos. El índice es invertido porque, normalmente, nosotros pensamos que las palabras son parte de los documentos, pero si invertimos nuestro pensamiento, los documentos están asociados con las palabras. Los términos de un índice están a menudo ordenados alfabéticamente, como en un libro, pero no es necesario ya que puedes buscar el término mediante una tabla hash. Cada término del índice tiene su propia lista invertida que contiene los datos relevantes para ese término. Esos datos pueden ser una lista de documentos o una lista de ocurrencias. Cada entrada en esa lista es denominada posting y la parte del posting que designa a un documento específico o una posición es llamado puntero. Cada documento en la colección tiene un número que lo identifica y que hace más eficiente el almacenamiento de los punteros de los documentos.

### **2.5.5. Consultas**

A pesar de que la estructura del índice y los algoritmos de ranking son componentes claves en un motor de búsqueda, desde el punto de vista del usuario el motor de búsqueda es principalmente una interfaz para realizar una consulta y ver los resultados. Los usuarios no pueden cambiar la forma en que un algoritmo realiza el ranking, pero pueden interaccionar con el sistema mediante la formulación y la reformulación de la consulta y mientras revisa los resultados. Estas interacciones son fundamentales en el proceso de recuperación de información y pueden determinar si el motor de búsqueda realizar un servicio eficaz.

La forma más común de una consulta en los actuales motores de búsqueda consiste en un pequeño número de palabras clave. Algunas de estas consultas pueden usar marcas para indicar una frase, si una palabra debe de estar presente...

Las palabras en el texto de la consulta deben ser transformadas en los mismos términos en los que fue transformado el texto del documento. La conservación de las palabras vacías en el índice incrementa la exactitud del sistema ya que suele tratar con consultas que también tienen palabras vacías. Por ejemplo, si en el Quijote queremos buscar el capítulo donde aparece “En un lugar de la Mancha”, se encontrará antes el documento en el que aparece esa consulta que si se ejecuta “lugar mancha”.

Las palabras vacías pueden ser tratadas como palabras normales, quitarse o mantener bajos algunas condiciones.

Recordemos que la lematización reduce las diferentes formas de una palabra a su lema o raíz. La consulta basada en lematización es una técnica que incrementa la flexibilidad del motor de búsqueda. Si las palabras en los documentos son lematizadas durante la indexación, las palabras en la consulta deben ser también lematizadas, pero puede haber ocasiones en las que reduzca la exactitud de los resultados.

### **2.5.6. Modelos de recuperación**

En los últimos 45 años una de las metas de la investigación de la recuperación de información ha sido la de comprender y formalizar los procesos que llevan a una persona hacer que un texto sea relevante o no. Para eso se proponen teorías sobre la relevancia en forma de modelos de recuperación matemáticos y comparan esas teorías con las decisiones de los usuarios. Los algoritmos con un buen modelo de recuperación recuperarán documentos relevantes en las primeras posiciones del ranking. A continuación, se describirán brevemente los principales modelos clásicos de RI.

#### *Modelo Booleano*

El modelo de recuperación booleano fue utilizado por los motores de búsqueda más antiguos e incluso hoy se sigue utilizando. Los documentos son recuperados si coinciden exactamente con la consulta y se desechan los que no. Este modelo generalmente no es descrito como un algoritmo de ranking. Esto es así porque se asume que todos los documentos recuperados son equivalentes en términos de relevancia. Esta relevancia es binaria. El nombre de booleano viene de que solamente hay dos posibles salidas para la consulta, verdadero o falso y porque la consulta normalmente es especificada usando operadores lógicos.

Es un hecho que la cantidad de términos es muy grande en colecciones de importante tamaño. Esto produce unos tiempos de recuperación altos ya que hay una gran cantidad de datos a tratar. Además, la eficacia depende totalmente del usuario ya que se debe formular de forma muy exacta la consulta utilizando los correspondientes operadores lógicos. Debido a la falta de un algoritmo de ranking, las consultas simples no trabajan bien. Todos los documentos que contienen las palabras especificadas en la consulta serán recuperados, y esta recuperación será presentada en un orden que tiene poco que ver con la relevancia.

#### *Modelo del espacio vectorial*

En este modelo, los documentos y las consultas son parte de un  $t$ -espacio dimensional, donde  $t$  es el número de términos en el índice. Un documento  $D_i$  es representado como vector de términos indexados:

$$D_i = (d_{i1}, d_{i2}, \dots, d_{it}),$$



donde  $d_{ij}$  representa el peso del termino  $j$ -ésimo. Una colección de documentos contiene  $n$  documentos pueden ser representado como una matriz de pesos, donde cada fila representa a un documento y cada columna describe el peso que cada término tiene en un documento:

	$Term_1$	$Term_2$	...	$Term_t$
$Doc_1$	$d_{11}$	$d_{12}$	...	$d_{1t}$
$Doc_2$	$d_{21}$	$d_{22}$	...	$d_{2t}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$Doc_n$	$Doc_{n1}$	$Doc_{n2}$	...	$Doc_{nt}$

Las consultas son representadas de la misma forma que los documentos. Una consulta  $Q$  es representada por el vector de  $t$  pesos:

$$Q = (q_1, q_2, \dots, q_t),$$

donde  $q_j$  es el peso del termino  $j$ -ésimo en la consulta.

Dada esta representación los documentos pueden clasificarse calculando la distancia entre los puntos que representan los documentos y la consulta. Esta distancia es llamada medida de similitud. Así, los documentos con mayor puntuación son los más similares a la consulta. La medida de similitud más fructuosa es la medida por similitud coseno, que mide el coseno del ángulo entre la consulta y los vectores del documento. Cuando los vectores son normalizados todos los documentos y consultas son representados por vectores de igual longitud y, por tanto, el coseno del ángulo entre dos vectores será 1 si el ángulo coincide, y cero si no coincide ningún término. La medida coseno es definida como:

$$Similitud\ coseno\ (D_i, Q) = \frac{\sum_{j=1}^t d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^t d_{ij}^2 \cdot \sum_{j=1}^t q_j^2}}$$

El numerador es la suma del producto de los pesos de los términos por las coincidencias de la consulta y los términos del documento. El denominador normaliza este valor dividiendo por el producto de las longitudes de los dos vectores.

Diferentes esquemas para calcular el peso han sido diseñados a lo largo de los años. La mayoría de ellos son variaciones de la ponderación  $tf \cdot idf$ . El parámetro  $tf$  es la frecuencia del término en el documento  $D_i$ . Un término que aparece muchas veces en un documento es más importante que otro que solo aparece una vez. El  $idf$  representa la importancia del término en la colección de documentos. Un término que aparece pocas veces en la colección tiene un mayor poder discriminador que un término que aparece en todos los documentos. Viene dado por la siguiente expresión:

$$idf(t) = \log\left(\frac{D}{D_t}\right)$$

donde  $D$  es el número de documentos en de la colección y  $D_t$  es el número de documentos en los que aparece el término  $t$ .

El efecto de estos dos parámetros se da multiplicándolos. La razón de ello es, sobre todo, empírica.

### Modelo probabilístico (BM25)

Una de las características que un modelo de recuperación proporciona es una clara declaración sobre las suposiciones en las que se basa. El modelo booleano y el modelo de espacio vectorial hace implícitas las suposiciones sobre la relevancia y de la representación del texto que afectan al diseño y eficacia de los algoritmos de clasificación. Lo ideal sería que dadas unas suposiciones un algoritmo de clasificación consiguiera una efectividad mejor que cualquier otro modelo.

BM25 amplía la función del modelo de independencia binaria<sup>4</sup> incluyendo los pesos para los términos del documento y la consulta. El modelo BM25 ha influido en algoritmos de rankings de motores de búsqueda comerciales. Hay varias variaciones de la función de puntuación para el algoritmo BM25, pero la más común es:

$$\sum_{i \in Q} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1)f_i}{K + f_i} \cdot \frac{(k_2 + 1)qf_i}{k_2 + qf_i}$$

donde las suposiciones ahora se hacen sobre todos los términos de la consulta y donde N es el número total de documentos, R es el número de documentos relevantes para la consulta,  $r_i$  es el número de documentos relevantes que tienen al término  $i$ ,  $n_i$  el número de documentos que contienen al término  $i$ ,  $f_i$  es la frecuencia del término  $i$  en el documento,  $qf_i$  es la frecuencia del término  $i$  en la consulta y  $k_1, k_2$  y  $K$  son parámetros que se establecen de forma empírica. La constante  $k_1$  determina como la componente  $tf$  del término cambia cuando se incrementa  $f_i$ . Experimentalmente se ha observado que los mejores resultados se consiguen para  $k_1$  igual a 1.2. La constante  $k_2$  tiene un rol similar a la anterior en el peso del término en la consulta. El valor de K viene dado por la siguiente expresión:

$$K = k_1 \left( (1 - b) + b \cdot \frac{dl}{avdl} \right),$$

dónde  $b$  es un parámetro,  $dl$  es la longitud del documento, y  $avdl$  es la media de la longitud de los documentos de la colección. La constante  $b$  regula el impacto de la longitud de la normalización, donde si  $b$  vale 0 corresponde a que no hay normalización de la longitud y si vale 1 la normalización es completa. El valor con el que se consiguen mejores resultados es para  $b$  igual a 0.75.

---

<sup>4</sup> Wikipedia – El *modelo de independencia binaria* es una técnica de recuperación de información probabilística que hace algunas suposiciones para hacer más fácil la determinación de la similitud documento/consulta.



## 3. Fases del trabajo

### 3.1. Enunciado del problema

Se nos pide una aplicación que muestre como funciona un sistema de recuperación de información. Para ello se debe mostrar, paso a paso, como se procesan los documentos para poder crear a partir de ellos un índice lo más exacto y eficiente posible, que visualice el índice una vez creado con información relevante de los términos y la posibilidad de realizar consultas sobre este índice.

También se debe de realizar una utilidad para cargar la colección en el punto último que se haya procesado, es decir, si solamente se creó el procesamiento de texto el usuario solamente puede cargar la colección a partir de ahí, al igual que si ha realizado hasta la creación del índice o hasta la fase final que son las consultas. También se debe poder eliminar colecciones o crear de nuevo una colección sustituyendo la que ya exista.

Se debe realizar la internacionalización de la interfaz en castellano e inglés, así como un menú de ayuda que explique cómo funciona cada uno de los contenidos de la aplicación.

Los documentos a tratar deben ser archivos de texto del tipo de los más utilizados.

### 3.2. Fase de análisis

En esta fase se establece qué requisitos debería de cumplir nuestra aplicación, tanto software como hardware, para que se cumplan los objetivos. También se lleva a cabo toda la tarea de ingeniería del software para la implementación del proyecto que será detallado más abajo.

### 3.3. Selección del lenguaje de programación

En esta fase se ha estudiado el lenguaje de programación a utilizar, intentando que la aplicación pudiese ejecutarse en cualquier máquina con cualquier sistema operativo.

Para cumplir con este requisito se pensó en utilizar lenguaje HTML utilizando PHP. Esto no pudo ser así ya que se hubiese necesitado un servidor que fuese capaz de procesar grandes cantidades de documentos, algunos de ellos muy pesados, de forma concurrente y con una gran cantidad de memoria para almacenar las colecciones y los índices. También se pensó en que los usuarios deberían de tener una conexión a banda ancha para poder subir al servidor los documentos a procesar y la carga de tráfico que esto generaría.

Una segunda opción y definitiva fue la de crear una aplicación en JAVA. Esta fue la elegida por el hecho de poder ser ejecutada en cualquier sistema operativo sin necesidad de ser recompilado e independiente de la arquitectura de la máquina, ser robusto, seguro, con capacidad de ejecutar aplicaciones multihilo, modularidad y de alto rendimiento.

### 3.4. Estudio de los recursos necesarios

En este momento se estudió el funcionamiento de la API de Lucene para JAVA y la estructura que tendría nuestro proyecto. También se contempló que librerías adicionales hacían falta para realizar el proyecto.

Se optó por Apache-Tika, una API de extracción de texto para archivos de distinto formato a formato de texto plano. Entre los documentos aceptados tenemos archivos txt, PDF, HTML, XML y doc que hasta la fecha son los más utilizados por los sistemas de recuperación de información.

El entorno de programación elegido ha sido Netbeans. Netbeans permite desarrollar aplicaciones mediante un conjunto de componentes software llamados módulos. El objetivo principal de esta plataforma es la de desarrollar aplicaciones en JAVA, pero también permite programar en otros lenguajes como PHP, HTML o C/C++. Netbeans nos ofrece un modo de generación de interfaz gráfica muy dinámica y sencilla, permitiendo agregar componentes con tan solo arrastrar y soltar en el lugar adecuado. También nos permite agregar de manera muy simple las dependencias necesarias para poder desarrollar nuestra aplicación.

### 3.5. Fase de desarrollo

Esta es la fase que más tiempo ha requerido y es por la cual se hace posible la culminación de nuestro proyecto. Esta fase conlleva la implementación y comprobación del funcionamiento de proyecto.

### 3.6. Planificación del proyecto

El proyecto en un primer momento se hizo el siguiente diagrama de Gannt en el que se planificaba el proyecto con una duración de 69 días. Esta planificación ha servido para controlar la duración de cada uno de los periodos del proyecto. Al final del documento se verá el diagrama de Gannt definitivo, en el que se verá el tiempo real que se ha empleado en realizar la aplicación.

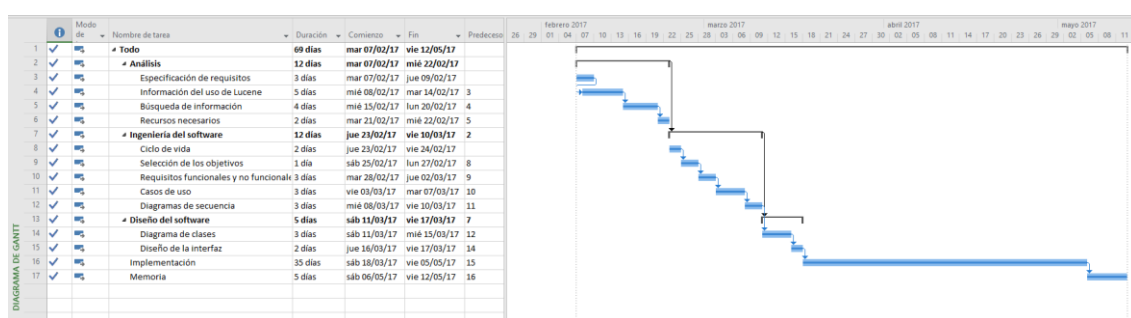


Ilustración 3: Diagrama de Gannt inicial

## 4. Ingeniería del software

Este apartado describirá el proceso de ingeniería del software realizado para llevar a cabo nuestro proyecto. En este proceso se ha analizado las necesidades del proyecto, también llamado análisis de requisitos y después se han realizado los modelos para ser implementados en JAVA y evaluar su funcionamiento.

### 4.1. Ciclo de vida

El ciclo de vida es lo que va ilustrarnos el camino que debemos seguir a lo largo de nuestro proyecto.

Para este caso se ha optado por un ciclo de vida en espiral. Es decir, se ha ido diseñando una fase del sistema, se ha ido implementando y se ha evaluado su funcionamiento y así con las fases incrementando el tamaño del proyecto hasta su implementación final.

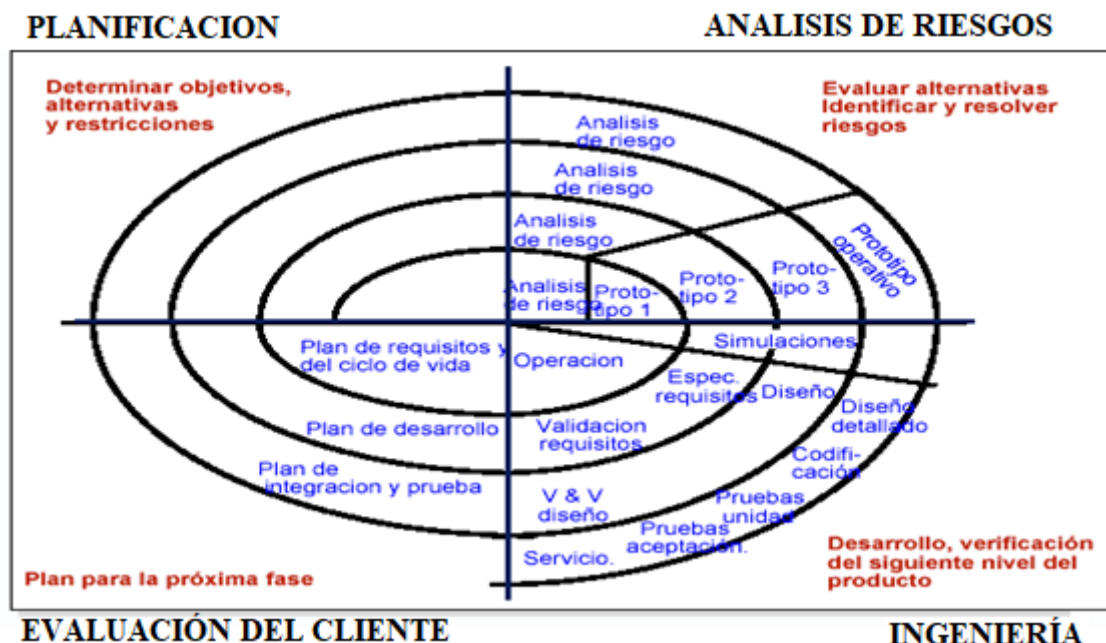


Ilustración 4: Ciclo de vida en espiral utilizado

Este modelo tiene cuatro partes que son las siguientes:

**Planificación:** En esta fase se han establecido cada uno de los objetivos de proyecto, alternativas y restricciones. Es lo que viene a ser una recolección de requisitos del sistema.

**Análisis de riesgos:** Todo proyecto conlleva consigo una serie de riesgos, ya sean referidos al proyecto o por las decisiones que se toman mientras se desarrolla. Aquí se decide cómo se resuelven y las posibles opciones a cada uno de ellos.

**Ingeniería:** En esta fase se crea los prototipos de la aplicación teniendo en cuenta las fases anteriores. Aquí se ha implementado la fase de la aplicación.

Evaluación del cliente: En este caso se ha realizado la prueba del módulo implementado en la etapa anterior y se ha corregido los fallos de su funcionamiento.

En cada iteración se han construido diferentes versiones, más completas y obteniéndose al final el modelo completo.

## 4.2. Objetivos del sistema

Los objetivos que se desean alcanzar una vez desarrollado el software son:

OBJETIVO 1	El sistema deberá de permitir la inicialización de creación de un sistema de recuperación de información a través de colecciones de documentos que tengamos en nuestro ordenador.
OBJETIVO 2	El sistema permitirá ver paso a paso el procesamiento de los documentos, en el cual se produce la tokenización, la eliminación de las palabras vacías y la lematización. Esto deberá de poder hacerse término a término o documento por documento.
OBJETIVO 3	La aplicación deberá de permitir la creación de un índice gracias a las funcionalidades que nos ofrece la API de Lucene. Además, debe de ser visible el estado final del índice de términos.
OBJETIVO 4	Una vez construido el índice se podrá realizar consultas al mismo de tres tipos diferentes: Una consulta debe de ser de tipo booleana, otra del tipo probabilística y otra de tipo espacio vectorial. Además de ser visible algunas características de la consulta como el tiempo de recuperación y la puntuación de los documentos por cada tipo de consulta.
OBJETIVO 5	Se deberá de poder mostrar un visor de documentos para ver los documentos originales que tenemos cargados en la colección.
OBJETIVO 6	Se podrá consultar un menú de ayuda de cada uno de los contenidos de la aplicación.
OBJETIVO 7	El sistema tendrá que permitir al usuario cargar colecciones ya iniciadas a partir del último paso de creación del índice que completó.

## 4.3. Requisitos funcionales

Los requisitos funcionales del sistema son los siguientes:

RF – 1	Añadir colección	El sistema deberá permitir que un usuario agregue una nueva colección para poder realizar todo el proceso de creación de un sistema de recuperación de información.
RF – 2	Eliminar colección	El sistema deberá permitir al usuario eliminar una colección ya existente en el sistema.
RF – 3	Cargar colección	El sistema debe de permitir a un usuario cargar una colección desde el último punto que se haya completado en la creación del sistema de recuperación de información.
RF – 4	Visualizar colección	El sistema permitirá, durante todo el proceso de creación del sistema de información, consultar los archivos originales de la colección.
RF – 5	Ver ayuda	El sistema proporcionará una ayuda por contenidos de la aplicación.

## 4.4. Requisitos no funcionales

### Implementación

- RNF – 1: El sistema debe de ser ejecutable en el mayor número de sistemas operativos posibles.
- RNF – 2: El sistema deberá optimizar, en el grado que sea posible, el procesamiento de los documentos.
- RNF – 3: Es aconsejable que el programa se ejecute en sistemas que posean un procesador de varios núcleos para aprovechar todo el potencial del procesamiento multihilo.
- RNF – 4: La aplicación ha de ejecutarse con la versión 8 de JAVA o superior

### Interfaz

- RNF – 4: La aplicación debe tener su interfaz gráfica en castellano y en inglés.
- RNF – 5: Se aconseja que tener una resolución de pantalla mayor a 1280x750.

### Soporte (Facilidad de mantenimiento y portabilidad)

- RNF – 6: El sistema se podrá expandir a una aplicación web en cuanto se tuviese equipos preparado para ello. Así, también se podrá implementar un sistema de gestión de usuarios para que estos tuviesen sus colecciones en la red.
- RNF – 7: La aplicación deberá poderse utilizar en el mayor número de sistemas diferentes posibles.

### Rendimiento

- RNF – 8: Sería aconsejable que la aplicación procesase los documentos de la forma más rápida posible.

## 4.5. Casos de uso

En esta aplicación solamente encontramos un actor y que será el usuario que utiliza la aplicación.

### 4.5.1 Descripción de los actores

Actor	Usuario	Usuario
Descripción	Será el encargado de proporcionar la colección de documentos y de hacer, paso por paso, todos los procedimientos para obtener un sistema de recuperación de información. También tendrá la posibilidad de cargar una colección ya creada o de eliminarla cuando ya no la necesite.	
Características	Es el único y principal actor de la aplicación y, por tanto, tendrá la posibilidad de interactuar con todos los módulos del sistema.	
Relaciones	Ninguna, no existe otro actor	
Referencias	RF – 1, RF – 2, RF – 3, RF – 4, RF – 5,	
Autor	Fecha	09/05/20017 Versión

## 4.5.2 Diagramas de casos de uso

### Diagrama de casos de uso completo

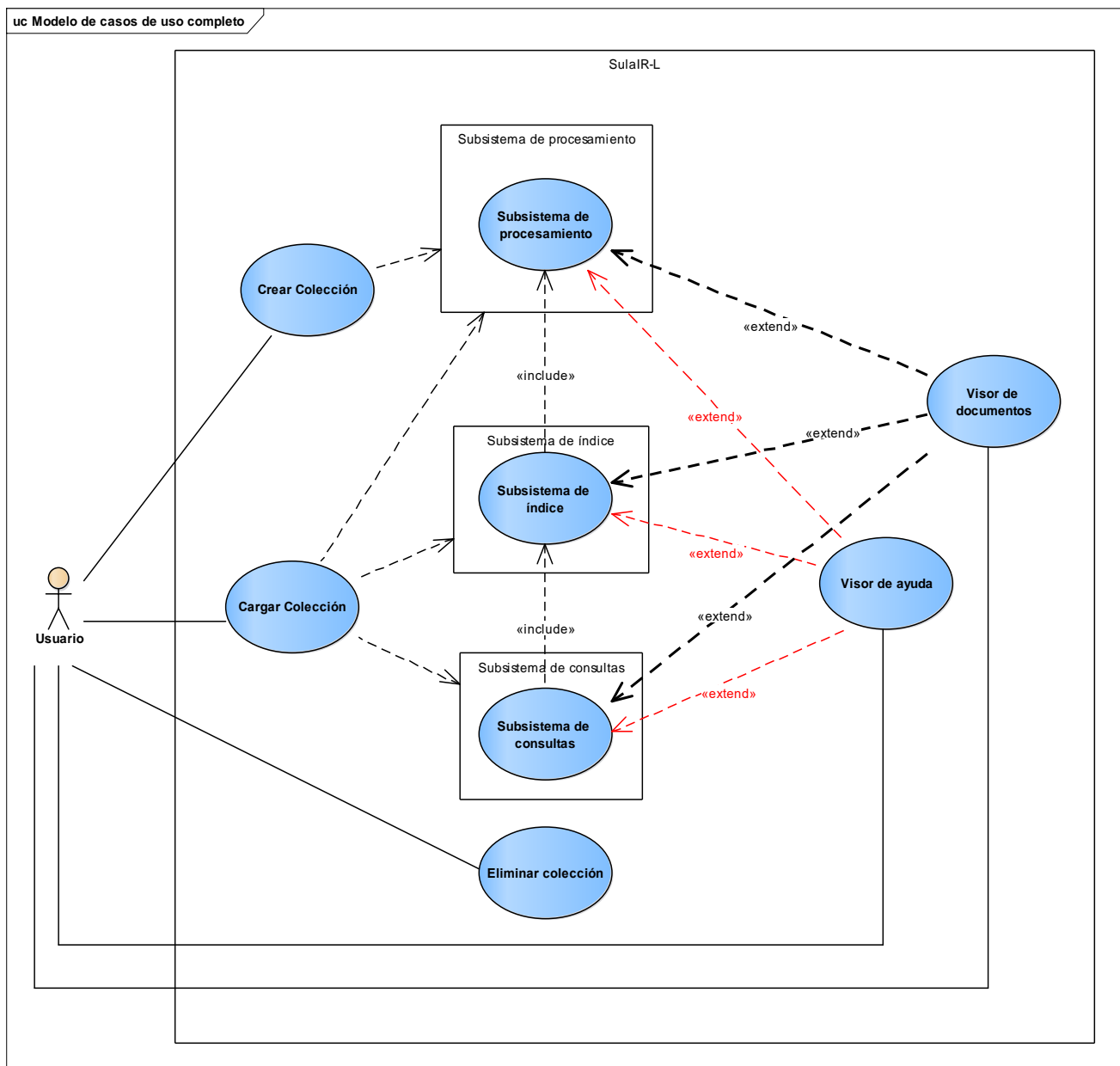


Ilustración 5: Diagrama de casos de uso

En este diagrama podemos ver que el sistema se encuentra dividido en otros tres subsistemas, como son: El subsistema de procesamiento, el subsistema de índice y el subsistema de consultas. Cada uno de ellos se detallará más adelante.

También podemos observar las funciones que puede realizar el actor con el sistema. Estas son: Cargar una colección existente, crear una nueva colección, eliminarla del sistema o abrir el visor de ayuda al programa y el visor de los documentos originales.

Ahora detallaremos los casos de uso de este diagrama y después de detallarán los correspondientes a los distintos subsistemas.

<b>CU – 01</b>	<b>Crear colección</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá permitir a un usuario crear una colección nueva para la creación de un nuevo sistema de recuperación de información.	
<b>Precondición</b>	Debe de existir una colección de documentos válida.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea crear una colección nueva.
	2	El sistema pide la ruta de los documentos de la nueva colección.
	3	El usuario establece una dirección de la carpeta de la colección.
	4	El sistema pide introducir un nombre para la colección.
	5	El usuario proporciona un nombre para la nueva colección.
	6	El sistema pide si se va a utilizar archivo de palabras vacías.
	6.1	Si el usuario decide realizar el proceso sin eliminar las palabras vacías debe de indicarlo en la interfaz
	6.2	Si el usuario decide realizar el proceso eliminando las palabras vacías deberá introducir la ruta del disco donde se encuentra dicho archivo.
<b>Secuencia Alternativa</b>	7	El usuario acepta la creación de la nueva colección
	<b>Paso</b>	<b>Acción</b>
	3.a	Si el usuario no establece ninguna carpeta el sistema deberá mostrar un aviso.
	5.a	Si el usuario no introduce ningún nombre para la colección el sistema deberá de informar del error.
	6.a	Si el usuario indica que va a introducir un archivo de palabras vacías y no lo introduce el sistema deberá de mostrar un aviso.
<b>Postcondición</b>	7.a	Si la colección ya existe, el sistema deberá de preguntar al usuario si desea eliminar la colección existente.
	La colección se creó correctamente.	
<b>Extensiones</b>	7	Para que se cree la colección completamente se hace uso del <b>subsistema de procesamiento</b> .
<b>Rendimiento</b>	Lo más eficientemente posible	
<b>Frecuencia</b>	Este caso de uso se puede realizar bastantes veces al día	
<b>Importancia</b>	Muy importante	
<b>Urgencia</b>		
<b>Comentarios</b>		

<b>CU – 02</b>	<b>Cargar Colección</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá permitir a un usuario cargar una colección ya procesada desde el último paso que se completó	
<b>Precondición</b>	Debe de existir una colección ya creada.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea cargar una colección existente.
	2	El sistema muestra una lista con las colecciones existentes en el sistema.
	3	El usuario establece la colección a cargar.
	4	El sistema pide introducir la fase de creación del sistema de recuperación de información desde la que se va a cargar la colección.
	5	El usuario introduce la etapa desde la cual se va a seguir con la creación del sistema de recuperación de información
	6	El usuario acepta la carga de la colección.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
	7.a	Si el usuario ha seleccionado una fase no completada del proceso deberá de dar un error e informar al usuario.
	7.b	Si los documentos de la carpeta original no existen, falta alguna o algún archivo, el sistema deberá dar un error e informar al usuario.
	7.c	Si falta algún archivo en la carpeta que el sistema creó al crear la colección este deberá de dar un error e informar al usuario.
<b>Postcondición</b>	La colección se cargó correctamente.	
<b>Extensiones</b>	7	Para que se cargue una de la etapa del procedimiento de creación del sistema de recuperación de información debe completarse cada uno de los <b>subsistemas anteriores</b> al de la etapa seleccionada.
<b>Rendimiento</b>	Lo más eficientemente posible	
<b>Frecuencia</b>	Este caso de uso se puede realizar bastantes veces al día	
<b>Importancia</b>	Muy importante	
<b>Urgencia</b>		
<b>Comentarios</b>		

<b>CU – 03</b>	<b>Eliminar colección</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá permitir a un usuario eliminar una colección ya creada en el sistema.	
<b>Precondición</b>	Debe de existir una colección ya creada.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea eliminar una colección del sistema.

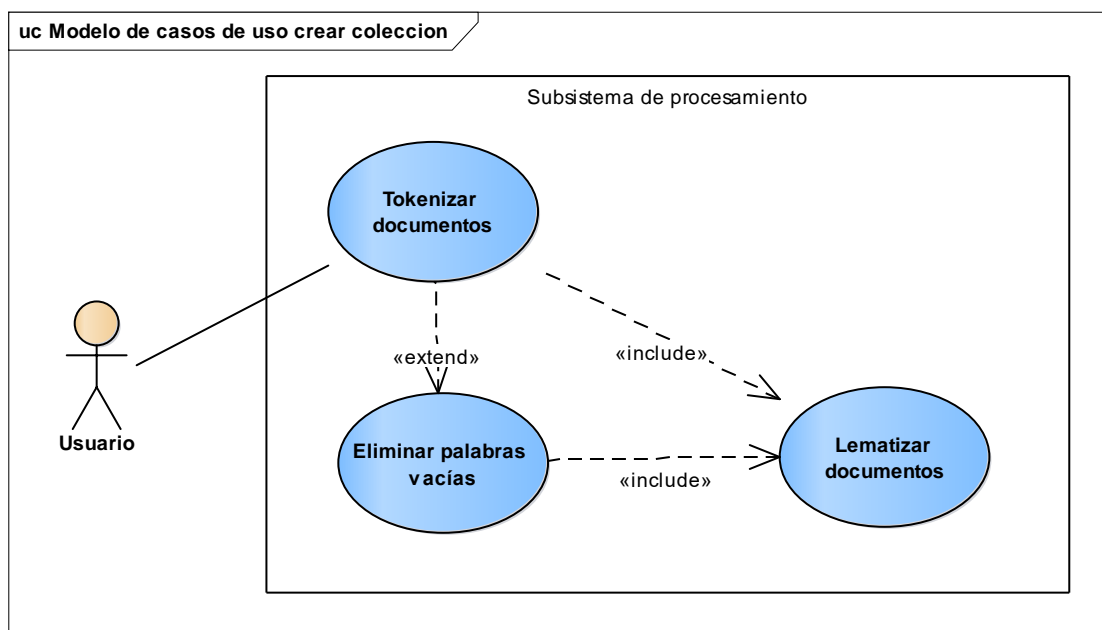


	2	El sistema muestra una lista con las colecciones existentes en el sistema.
	3	El usuario establece la colección a eliminar.
	4	El usuario selecciona en eliminar colección.
	5	EL sistema informa de que se va a eliminar la colección y pide confirmación
	6	El usuario acepta la eliminación de la colección
	7	El sistema elimina la colección.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
<b>Postcondición</b>	La colección se eliminó correctamente.	
<b>Extensiones</b>		
<b>Rendimiento</b>	Lo más eficientemente posible	
<b>Frecuencia</b>	Este caso de uso se puede realizar bastantes veces al día	
<b>Importancia</b>	Muy importante	
<b>Urgencia</b>		
<b>Comentarios</b>		

<b>CU – 04</b>	<b>Visor de documentos</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá de permitir visualizar un archivo que fue utilizado para crear el sistema de recuperación de información.	
<b>Precondición</b>	Debe de existir una colección ya creada o cargada y el archivo debe de existir en la carpeta original que se utilizó para crear la colección.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea visualizar un archivo de la colección que fue utilizado para crear el sistema de recuperación de información.
	2	El sistema muestra una lista con los archivos utilizados para crear la colección
	3	El usuario selecciona un archivo.
	4	El sistema muestra el archivo seleccionado.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
<b>Postcondición</b>	El archivo se visualizó correctamente.	
<b>Extensiones</b>		
<b>Rendimiento</b>	No precisa rapidez	
<b>Frecuencia</b>	Este caso de uso se puede realizar bastantes veces al día	
<b>Importancia</b>	Media	
<b>Urgencia</b>		
<b>Comentarios</b>		

<b>CU – 05</b>	<b>Visor de ayuda</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá de permitir visualizar al usuario información sobre la obtención de cada uno de los procesos del sistema de recuperación de información.	
<b>Precondición</b>	Debe de haber una colección cargada en el sistema.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea visualizar la ayuda sobre algún contenido o apartado del proceso de creación del sistema de información.
	2	El sistema muestra una lista con los contenidos de ayuda.
	3	El usuario selecciona un contenido.
	4	El sistema muestra la ayuda del contenido que desea el usuario.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
<b>Postcondición</b>	El sistema mostró la ayuda sobre el contenido solicitado por el usuario.	
<b>Extensiones</b>		
<b>Rendimiento</b>	No precisa rapidez	
<b>Frecuencia</b>	Este caso de uso se puede realizar bastantes veces al día	
<b>Importancia</b>	Media	
<b>Urgencia</b>		
<b>Comentarios</b>		

*Diagrama de casos de uso del subsistema de crear colección*



*Ilustración 6:Diagrama de casos de uso: Subsistema de procesamiento*

Este subsistema es el que permite crear la colección de documentos. Primero toma los documentos de la carpeta original, realiza sobre ellos algunas modificaciones para que puedan ser leídos correctamente y ejecutar su tokenización. Una vez realizado esto, puede pasar al proceso de eliminación de palabra vacías o directamente al de lematizar, ya que el procesamiento de las palabras vacías es opcional. Si se opta por eliminar las palabras vacías el siguiente paso es la lematización.

Los casos de uso referentes a este subsistema serán descritos a continuación:

<b>CU – 06</b>	<b>Tokenizar documentos</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá de permitir al usuario ver paso a paso como se tokeniza cada uno de los documentos de la colección.	
<b>Precondición</b>	Debe de haber una colección creada en el sistema.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El sistema pide iniciar la tokenización.
	2	El usuario inicia la tokenización y carga el primer documento de la colección.
	3	El sistema muestra el documento sin tokenizar.
	4	4.1. El usuario realiza la tokenización del documento término a término. 4.1. El usuario realiza la tokenización del documento de una vez.
	5	El sistema muestra el resultado de la tokenización del documento.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
<b>Postcondición</b>	La visualización de la tokenización de los documentos se ha realizado correctamente.	
<b>Extensiones</b>		
<b>Rendimiento</b>	Algo eficiente.	
<b>Frecuencia</b>	Este caso de uso se puede realizar todas las veces que el usuario quiera con todos los documentos de la colección	
<b>Importancia</b>	Alta	
<b>Urgencia</b>		
<b>Comentarios</b>	La rapidez puede depender del tamaño del documento que se procese.	

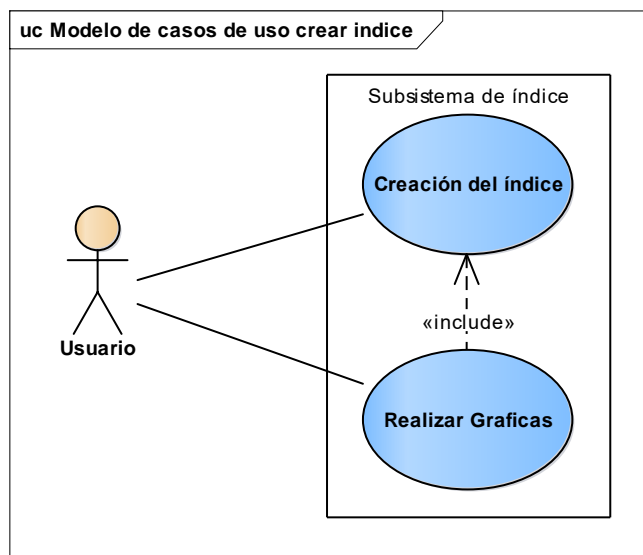
<b>CU – 07</b>	<b>Eliminar palabras vacías</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá de permitir al usuario ver paso a paso como se eliminan las palabras vacías de cada uno de los documentos de la colección.	
<b>Precondición</b>	Debe de haber una colección creada en el sistema.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>

	1	El sistema pide iniciar la eliminación de las palabras vacías.
	2	El usuario inicia la eliminación de las palabras y carga el primer documento de la colección.
	3	El sistema muestra el documento con las palabras vacías.
	4	4.1. El usuario realiza la eliminación de las palabras vacías del documento término a término. 4.1. El usuario realiza la eliminación de las palabras vacías del documento de una vez.
	5	El sistema muestra el resultado de la eliminación de las palabras vacías del documento.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
<b>Postcondición</b>	La visualización de la eliminación de las palabras vacías de los documentos se ha realizado correctamente.	
<b>Extensiones</b>		
<b>Rendimiento</b>	Algo eficiente.	
<b>Frecuencia</b>	Este caso de uso se puede realizar todas las veces que el usuario quiera con todos los documentos de la colección	
<b>Importancia</b>	Alta	
<b>Urgencia</b>		
<b>Comentarios</b>	La rapidez puede depender del tamaño del documento que se procese.	

<b>CU – 08</b>	<b>Lematizar documentos.</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá de permitir al usuario ver paso a paso como se va realizando la lematización de los términos de cada uno de los documentos de la colección.	
<b>Precondición</b>	Debe de haber una colección creada en el sistema.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El sistema pide iniciar la lematización de los documentos.
	2	El usuario inicia la lematización y carga el primer documento de la colección.
	3	El sistema muestra el documento sin el proceso de lematización.
	4	4.1. El usuario realiza la lematización del documento término a término. 4.1. El usuario realiza la lematización del documento de una vez.
	5	El sistema muestra el resultado de la lematización de los términos del documento.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
<b>Postcondición</b>	La visualización de la lematización de los términos de los documentos se ha realizado correctamente.	
<b>Extensiones</b>		

<b>Rendimiento</b>	Algo eficiente.
<b>Frecuencia</b>	Este caso de uso se puede realizar todas las veces que el usuario quiera con todos los documentos de la colección
<b>Importancia</b>	Alta
<b>Urgencia</b>	
<b>Comentarios</b>	La rapidez puede depender del tamaño del documento que se procese.

*Diagrama de casos de uso del subsistema de crear colección*



*Ilustración 7: Diagrama de casos de uso: Subsistema de Indexación*

Este subsistema es el encargado de crear el índice y el que nos habilita ver los detalles del mismo y crear las gráficas de las frecuencias/documento y de la distribución de los términos por documento. Los casos de uso de este subsistema se detallan a continuación:

<b>CU – 09</b>	<b>Creación del índice.</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá crear el índice invertido.	
<b>Precondición</b>	La colección debe estar cargada hasta el subsistema de procesamiento y haber llegado hasta la lematización.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea crear el índice
	2	El sistema habilita la creación del sistema al usuario.
	3	El usuario acepta la creación del índice invertido.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
<b>Postcondición</b>	El índice invertido ha sido creado	
<b>Extensiones</b>		
<b>Rendimiento</b>	Muy eficiente.	
<b>Frecuencia</b>	Una vez por colección.	
<b>Importancia</b>	Alta	

<b>Urgencia</b>	
<b>Comentarios</b>	La rapidez puede depender del número de documentos que tenga la colección.

CU – 10	Realizar Gráficas.		
Actores	Usuario		
Descripción	El sistema debe de permitir al usuario realizar las gráficas que visualizan la distribución las frecuencias de un término en la colección y la que visualiza la distribución de las frecuencias de los términos de un documento.		
Precondición	El índice tiene que estar creado.		
Secuencia Normal	Paso	Acción	
	1	1.1	El usuario desea realizar una visualización de la frecuencia de un término en los documentos de la colección.
		1.2	El usuario desea realizar una visualización de la frecuencia de los términos en un documento de la colección.
	2	2.1	El sistema muestra todos los términos de la colección
		2.2	El sistema muestra todos los documentos de la colección
	3	3.1	El usuario selecciona un término
		3.2	El usuario selecciona un documento
	4	El sistema dibuja la gráfica.	
Secuencia Alternativa	Paso	Acción	
	4.a	El usuario desea guardar la gráfica en un archivo	
	4.b	El sistema pide la localización y nombre del archivo	
	4.c	El usuario introduce la localización donde se guardará el archivo y el nombre de dicho archivo	
	4.d	El sistema guarda el archivo de la gráfica	
Postcondición	El sistema ha visualizado la gráfica al usuario.		
Extensiones			
Rendimiento	Poco eficiente.		
Frecuencia	Escasas veces.		
Importancia	No importante.		
Urgencia			
Comentarios	La rapidez puede depender del número de términos y/o documentos que tenga la colección.		

## Diagrama de casos de uso del subsistema de consultas

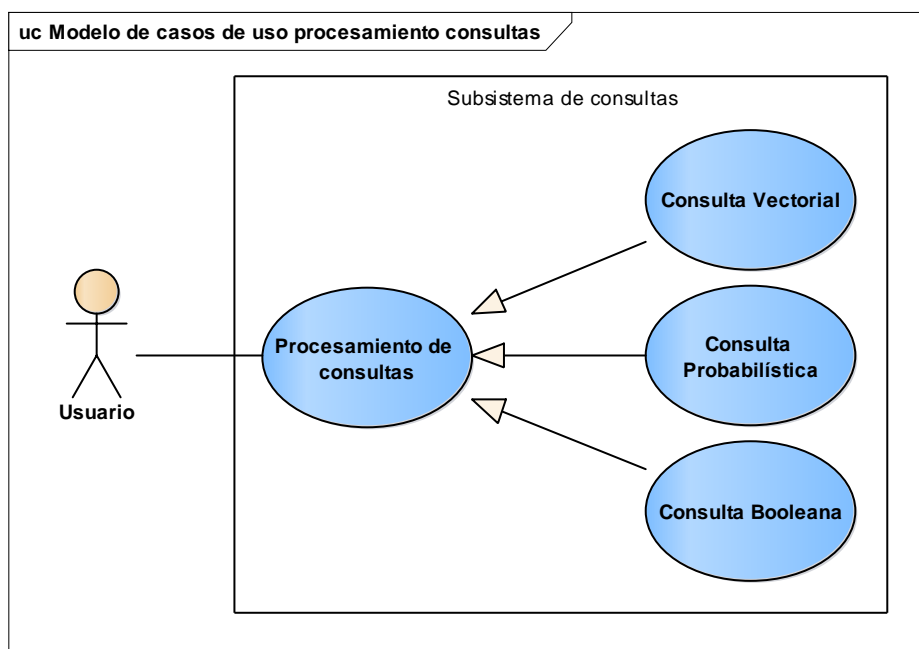


Ilustración 8: Diagrama de casos de uso: Subsistema de consultas

Este subsistema es el encargado de ofrecer la posibilidad al usuario de realizar consultas sobre el índice anteriormente creado. Estas consultas pueden ser de tres tipos: Una consulta basada en el modelo de espacio vectorial, otra basada en el modelo de las consultas booleanas y otra basada en el modelo de las consultas probabilísticas.

El detalle de los casos de uso de este subsistema es el siguiente:

<b>CU – 11</b>	<b>Procesamiento de consultas.</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá permitir al usuario la realización de tres tipos de consultas sobre el índice creado en el subsistema anterior.	
<b>Precondición</b>	El índice tiene que estar creado.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea realizar una consulta
	2	El sistema ofrece los tipos de consulta disponibles
	3	El usuario selecciona el tipo de consulta.
	4	El usuario realiza la consulta
	5	El sistema muestra el resultado de la consulta.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
<b>Postcondición</b>	El sistema ha visualizado la consulta del usuario.	
<b>Extensiones</b>	4	CU – 12, CU – 13, CU - 14
<b>Rendimiento</b>	Muy eficiente.	
<b>Frecuencia</b>	Muchas veces.	
<b>Importancia</b>	Muy importante.	

<b>Urgencia</b>	
<b>Comentarios</b>	

<b>CU – 12</b>	<b>Consulta vectorial.</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá permitir al usuario la realización de una consulta al índice usando el modelo de espacio vectorial.	
<b>Precondición</b>	El índice tiene que estar creado.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea realizar una consulta de tipo vectorial
	2	El sistema pide la consulta
	3	El usuario introduce la consulta
	4	El sistema pide el número máximo de documentos a recuperar
	5	El usuario introduce el número de documentos a recuperar como máximo.
	6	El sistema procesa la consulta
	7	El sistema muestra el resultado de la consulta.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
	6.a	Si el usuario no ha introducido ninguna consulta el sistema deberá de dar un mensaje de error informando al usuario que debe de introducirla.
<b>Postcondición</b>	El sistema ha visualizado la consulta vectorial al usuario.	
<b>Extensiones</b>		
<b>Rendimiento</b>	Muy eficiente.	
<b>Frecuencia</b>	Muchas veces.	
<b>Importancia</b>	Muy importante.	
<b>Urgencia</b>		
<b>Comentarios</b>		

<b>CU – 13</b>	<b>Consulta Booleana.</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá permitir al usuario la realización de una consulta al índice usando el modelo booleano.	
<b>Precondición</b>	El índice tiene que estar creado.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea realizar una consulta de tipo booleano
	2	El sistema pide la consulta
	3	El usuario introduce la consulta
	4	El sistema pide el número máximo de documentos a recuperar



	5	El usuario introduce el número de documentos a recuperar como máximo.
	6	El sistema procesa la consulta
	7	El sistema muestra el resultado de la consulta.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
	6.a	Si el usuario no introduce la consulta booleana de la forma correcta el sistema debe de mostrar el error informando de ello.
	6.b	Si el usuario no ha introducido ninguna consulta el sistema deberá de dar un mensaje de error informando al usuario que debe de introducirla.
<b>Postcondición</b>	El sistema ha visualizado la consulta booleana al usuario.	
<b>Extensiones</b>		
<b>Rendimiento</b>	Muy eficiente.	
<b>Frecuencia</b>	Muchas veces.	
<b>Importancia</b>	Muy importante.	
<b>Urgencia</b>		
<b>Comentarios</b>		

<b>CU – 14</b>	<b>Consulta Probabilística.</b>	
<b>Actores</b>	<b>Usuario</b>	
<b>Descripción</b>	El sistema deberá permitir al usuario la realización de una consulta al índice usando el modelo probabilístico.	
<b>Precondición</b>	El índice tiene que estar creado.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El usuario desea realizar una consulta de tipo probabilístico
	2	El sistema pide la consulta
	3	El usuario introduce la consulta
	4	El sistema pide el número máximo de documentos a recuperar
	5	El usuario introduce el número de documentos a recuperar como máximo.
	6	El sistema procesa la consulta
	7	El sistema muestra el resultado de la consulta.
<b>Secuencia Alternativa</b>	<b>Paso</b>	<b>Acción</b>
	6.b	Si el usuario no ha introducido ninguna consulta el sistema deberá de dar un mensaje de error informando al usuario que debe de introducirla.
<b>Postcondición</b>	El sistema ha visualizado la consulta probabilística al usuario.	
<b>Extensiones</b>		
<b>Rendimiento</b>	Muy eficiente.	
<b>Frecuencia</b>	Muchas veces.	
<b>Importancia</b>	Muy importante.	
<b>Urgencia</b>		

## 4.6. Diagramas de secuencia

Los diagramas de secuencia tratan de mostrar cómo se procesarían los diferentes casos de uso detallados anteriormente. Estos son los siguiente

### Diagrama de secuencia CU – 01, Crear Colección

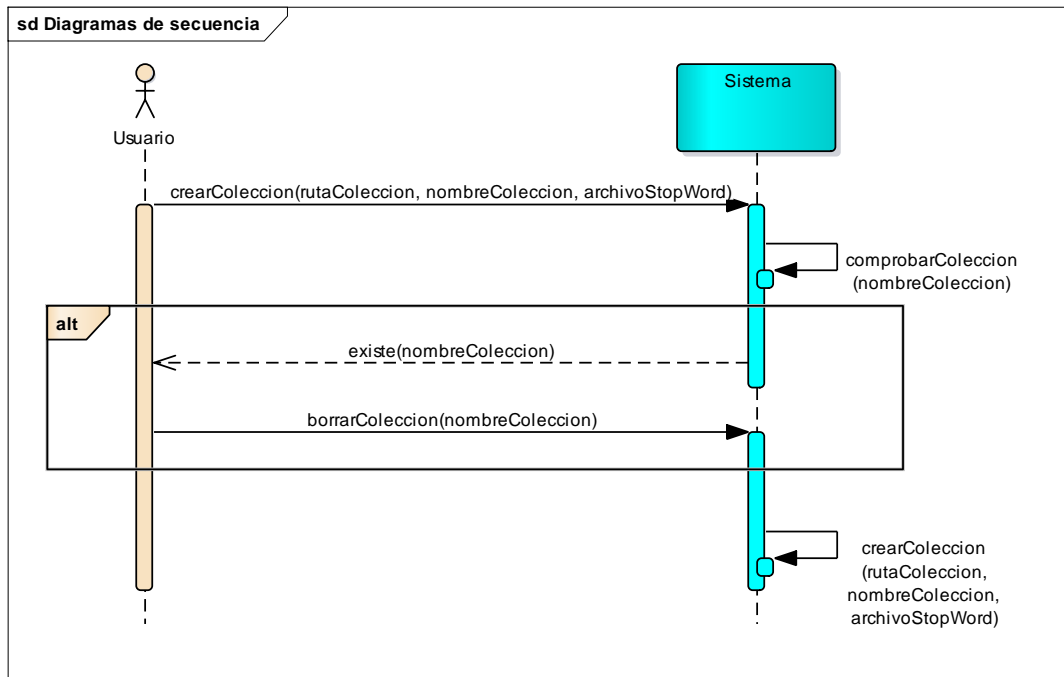


Ilustración 9: Diagrama de secuencia: Crear colección

### Diagrama de secuencia CU – 02, Cargar Colección

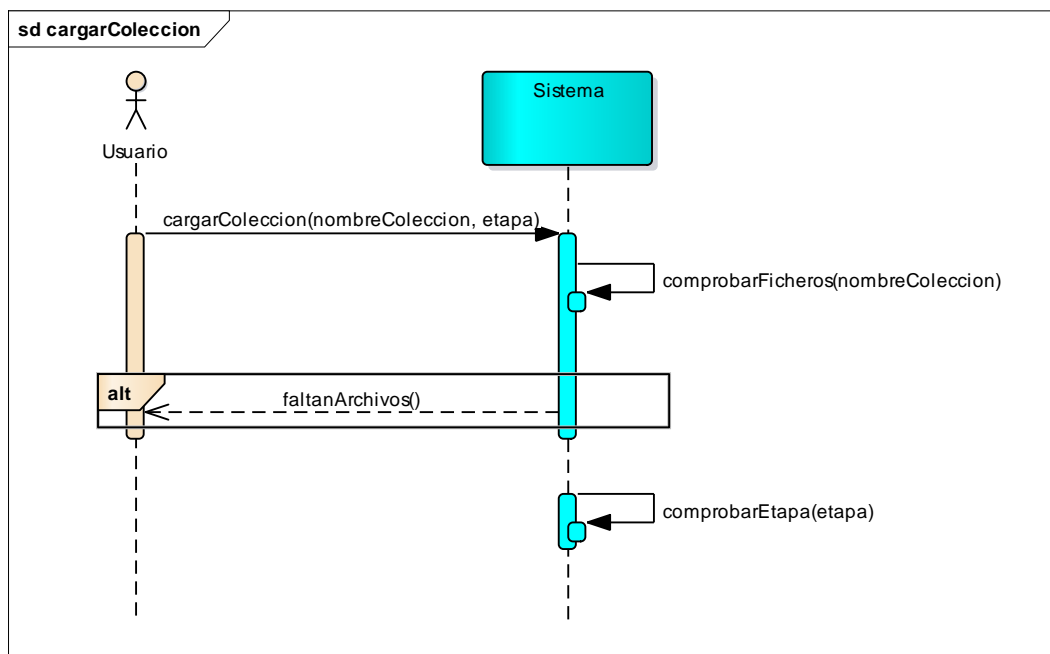


Ilustración 10: Diagrama de secuencia: Cargar Colección

### Diagrama de secuencia CU – 03, Eliminar Colección

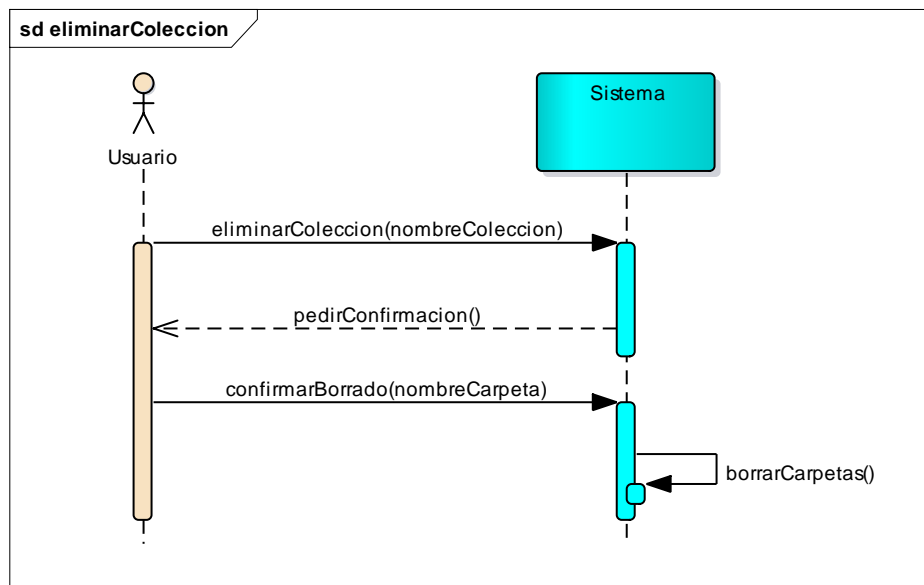


Ilustración 11: Diagrama de secuencia: Eliminar Colección

### Diagrama de secuencia CU – 04, Visor de documentos

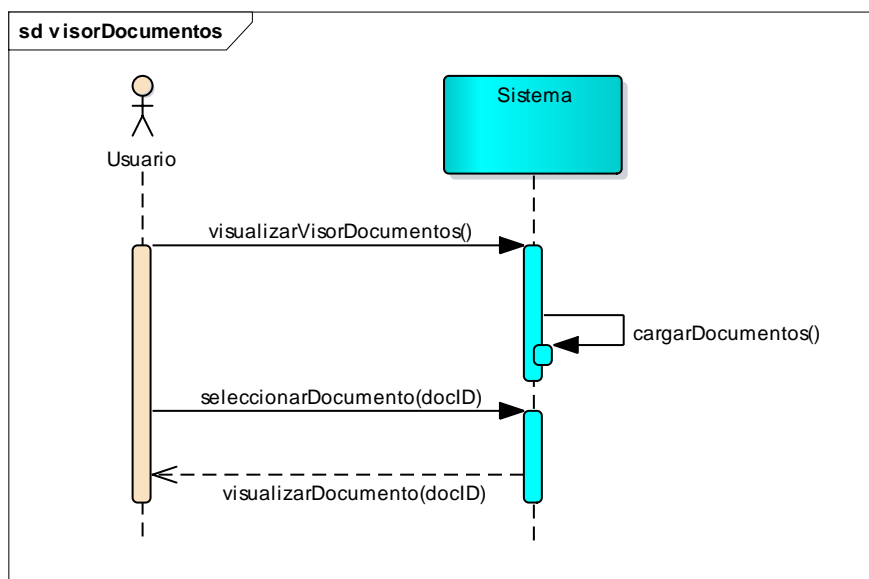


Ilustración 12: Diagrama de secuencia: Visor de documentos

## Diagrama de secuencia CU – 05, Visor de ayuda

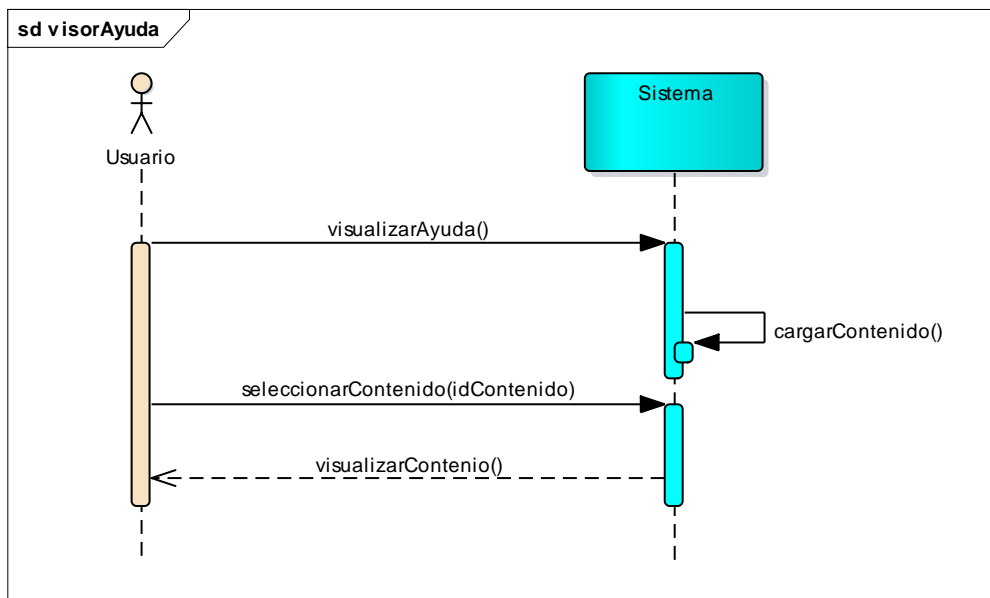


Ilustración 13: Diagrama de secuencia: Visor de ayuda

## Diagrama de secuencia CU – 06, Tokenizar documentos

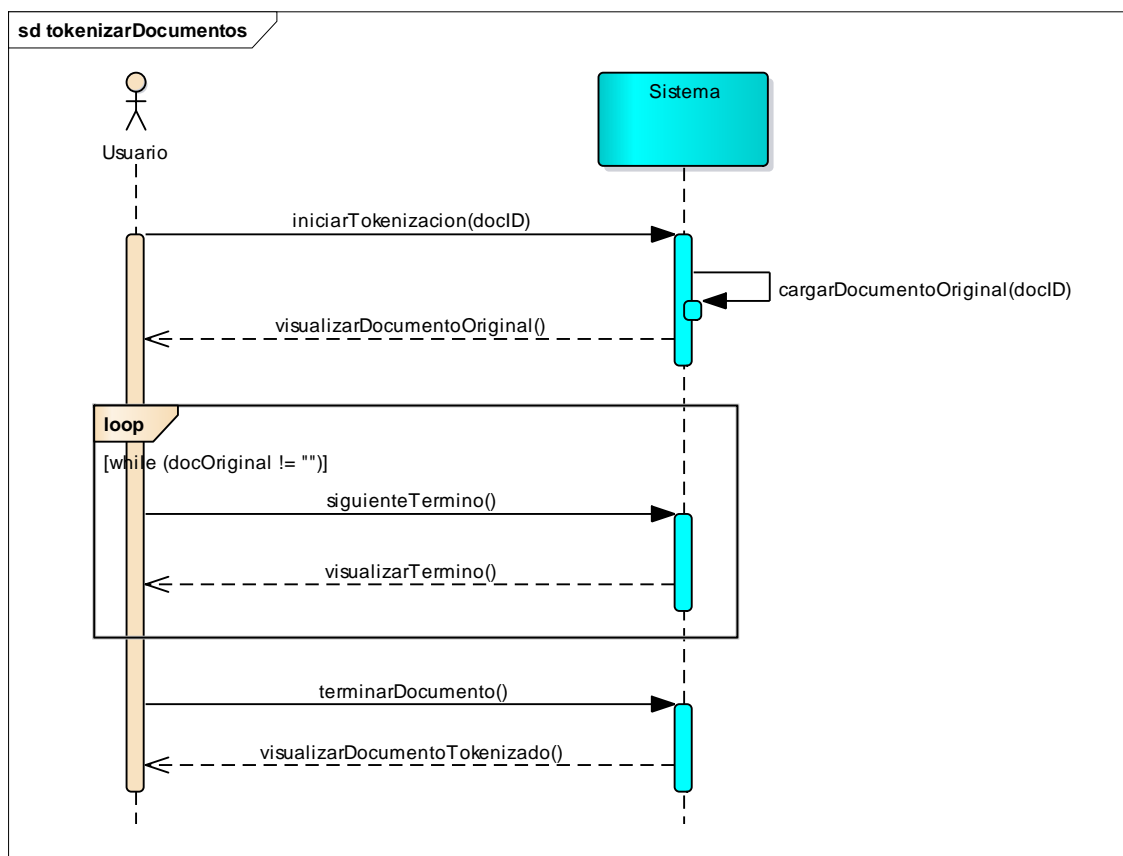


Ilustración 14: Diagrama de secuencia: Tokenizar Documentos

## Diagrama de secuencia CU – 07, Eliminar palabras vacías

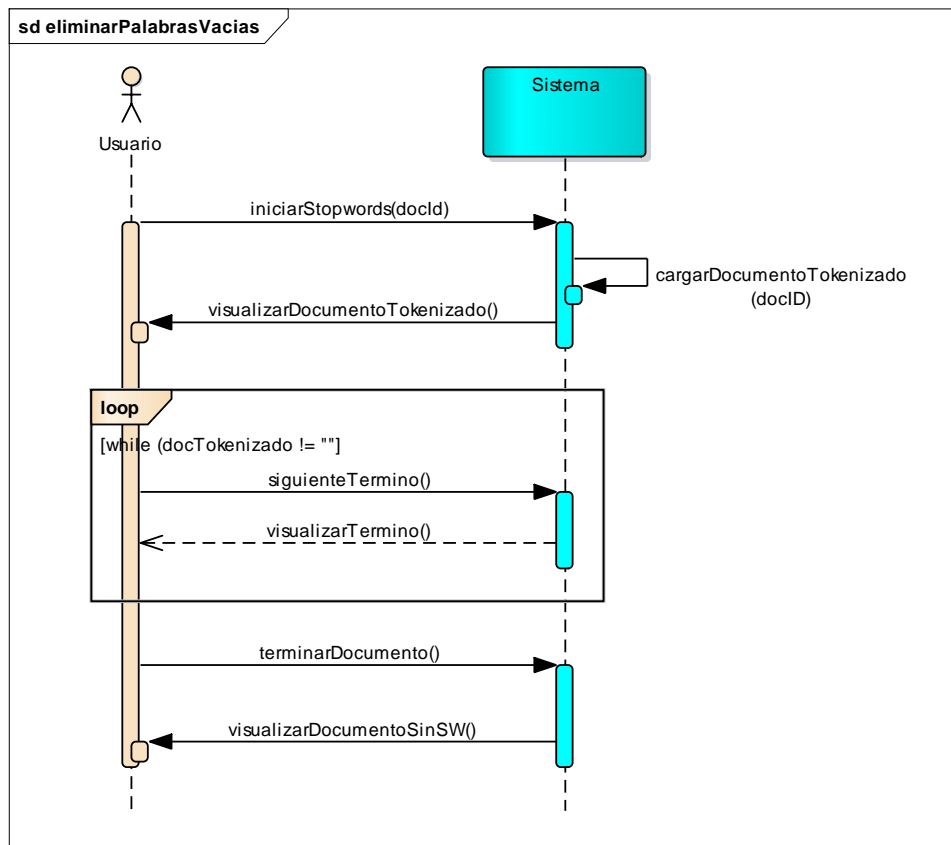


Ilustración 15: Diagrama de secuencia: Eliminar palabras vacías

## Diagrama de secuencia CU – 08, Lematizar documentos

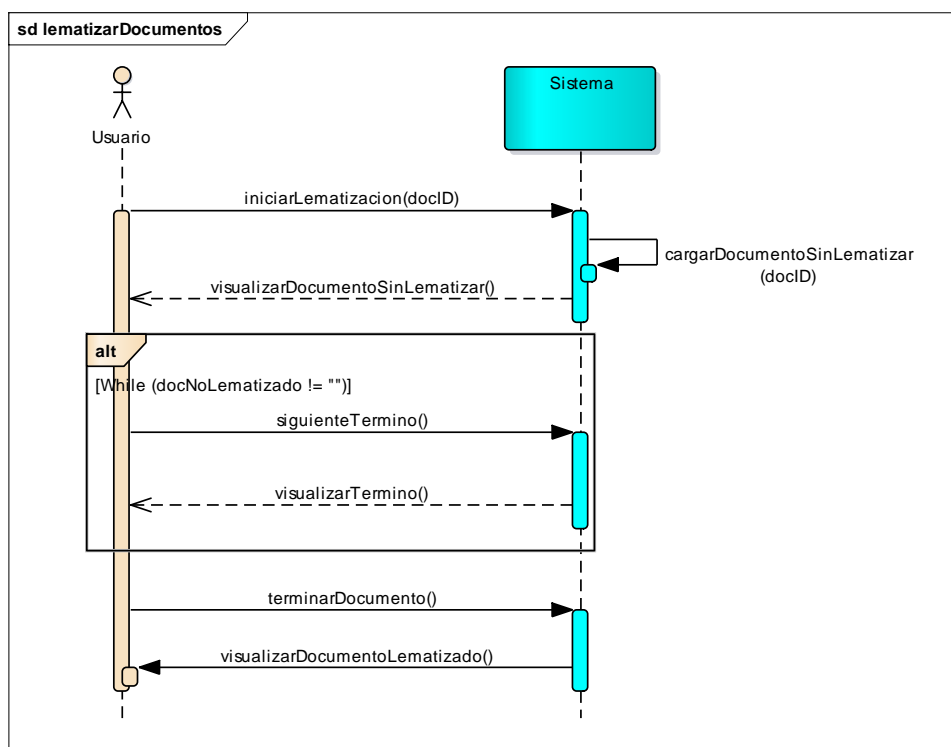


Ilustración 16: Diagrama de secuencia: Lematizar documentos

## Diagrama de secuencia CU – 09, Creación del índice

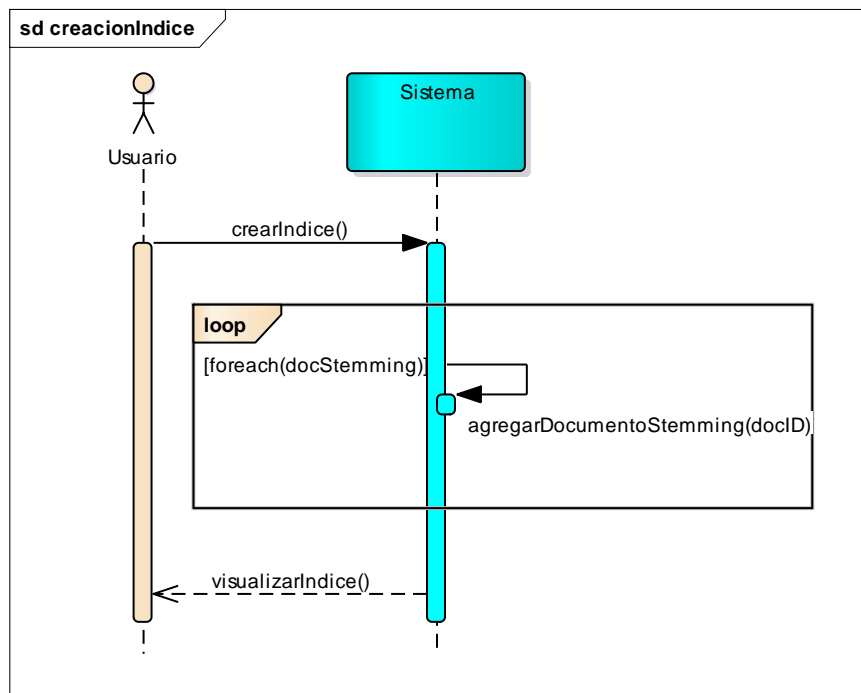


Ilustración 17: Diagrama de secuencia: Creación del índice

## Diagrama de secuencia CU – 10, Realizar gráficas

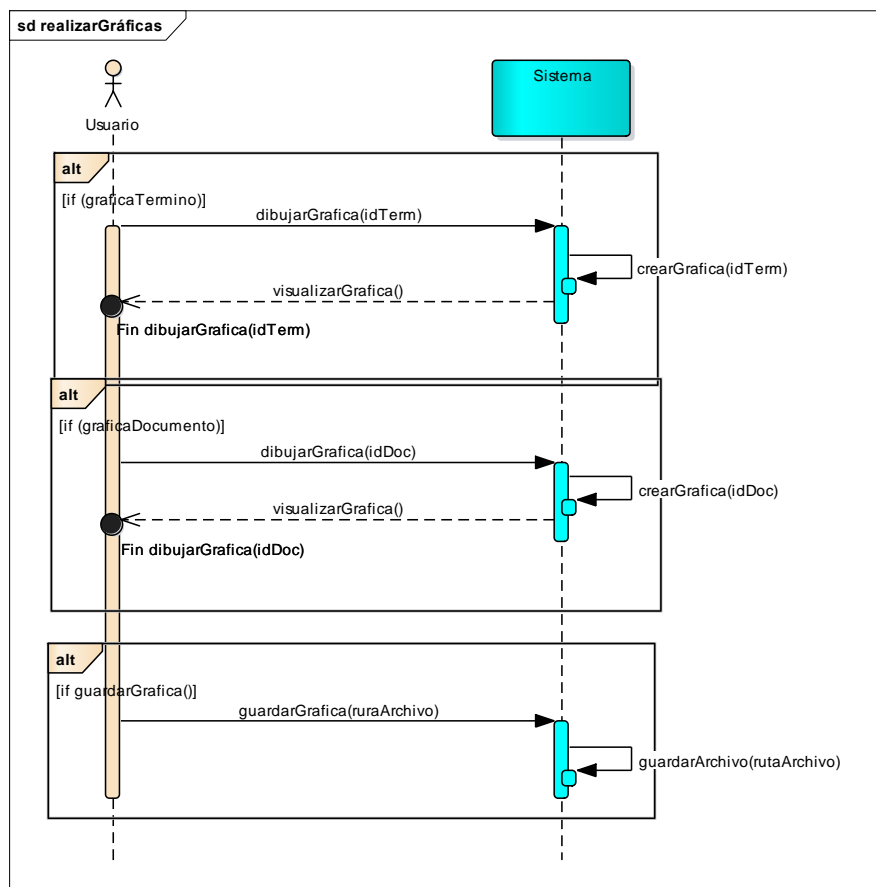


Ilustración 18: Diagrama de secuencia: Realizar Gráficas

## Diagrama de secuencia CU – 11, Procesamiento de consultas

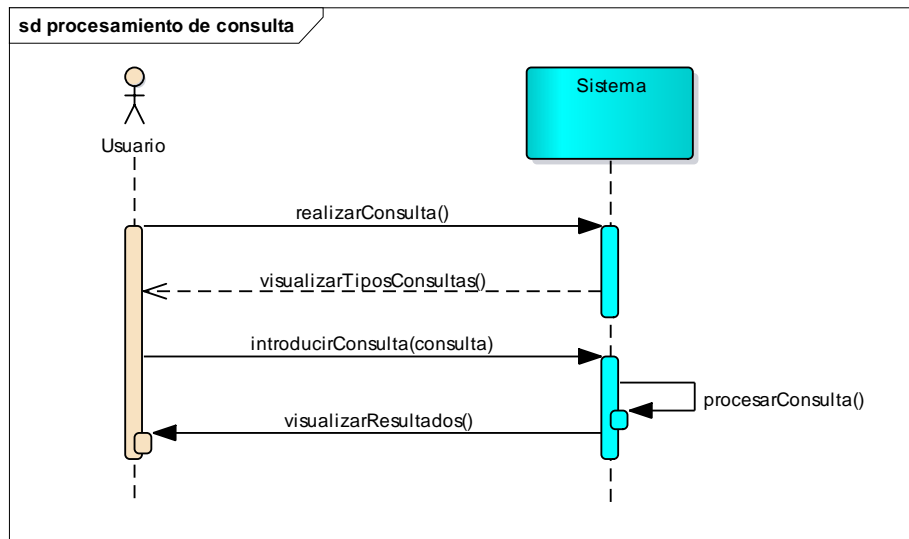


Ilustración 19: Diagrama de secuencia: Procesamiento de consulta

## Diagrama de secuencia CU – 12, Consulta vectorial

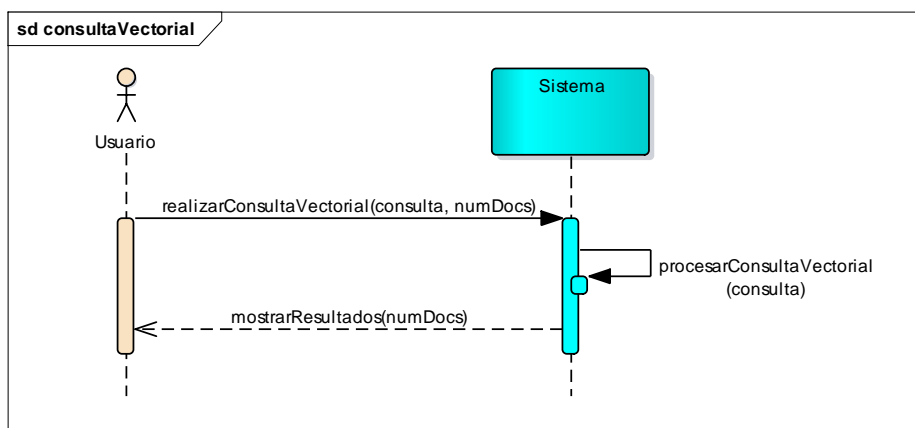


Ilustración 20: Diagrama de secuencia: Consulta Vectorial

### Diagrama de secuencia CU – 13, Consulta booleana

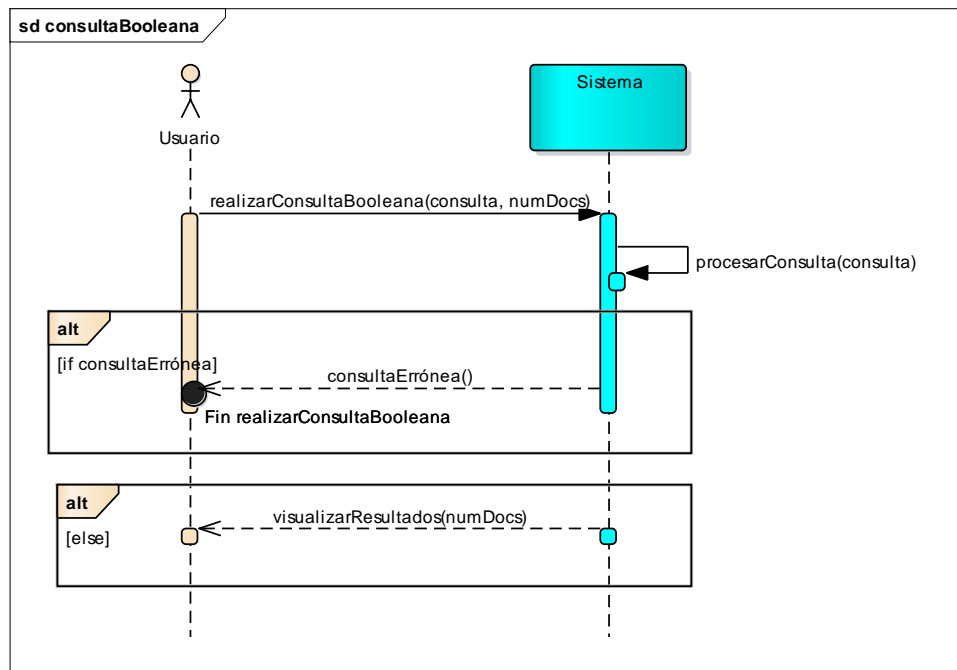


Ilustración 21: Diagrama de secuencia: Consulta Booleana

### Diagrama de secuencia CU – 14, Consulta Probabilística

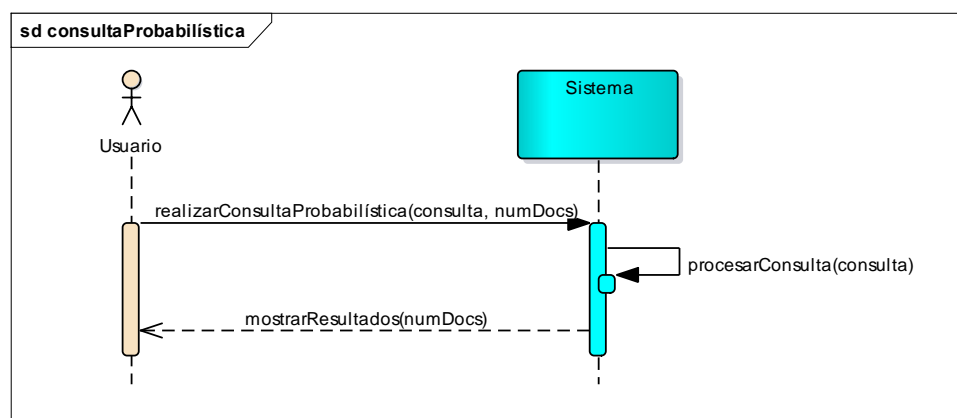


Ilustración 22: Diagrama de secuencia: Consulta Probabilística



## 5. Diseño

En este apartado se dará información sobre la arquitectura utilizada en el proyecto para hacer posible su implementación.

### 5.1. Diagrama de clases

Empezaremos con el diagrama de clases ([Ilustración 23](#)), que son los tipos de datos utilizados para hacer la implementación más sencilla gracias a las ventajas que nos proporciona como es la reusabilidad de módulos, la mantenibilidad, su modificabilidad y la fiabilidad. En este diagrama tenemos:

Una clase denominada ArchivoColección, donde tendremos la información que vamos a almacenar de los archivos de la colección (archivos tokenizados, archivos sin palabras vacías, archivos originales y archivos lematizados).

Una clase denominada colección, que tendrán la información sobre la colección que estamos procesando. Esta información es:

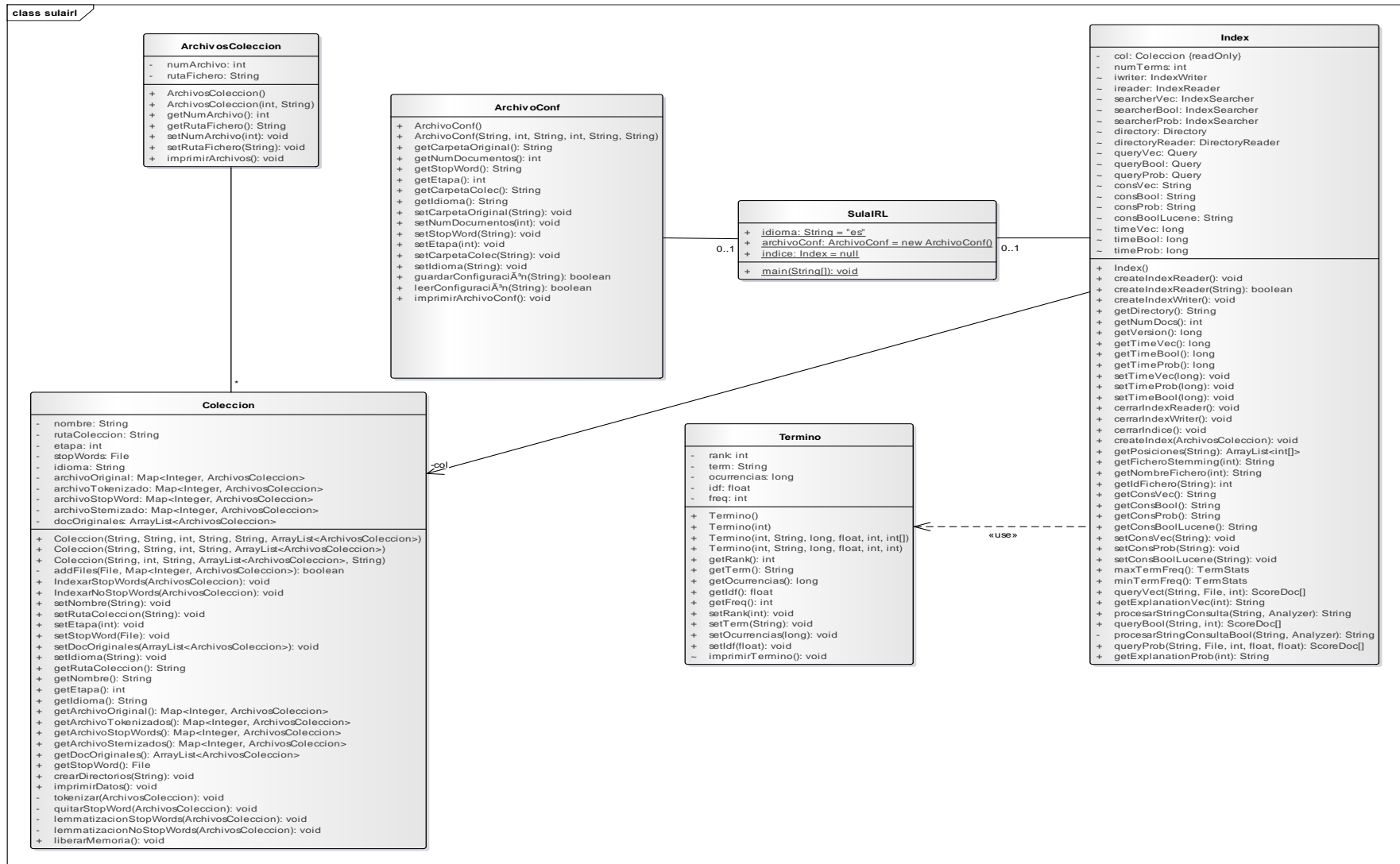
- Arrays con los archivos de palabras vacías, los archivos tokenizados, archivos originales y archivos lematizados.
- El idioma en el que está procesada la colección.
- La ruta de la carpeta de la colección.
- El archivo que contiene las palabras vacías y que será nulo si en la creación se indica que no se va a realizar este proceso.
- La etapa actual en la que se encuentra la colección.
- El nombre de la colección.
- Un array con los documentos que están en la carpeta original de la colección.

Una clase llamada Termino, en la que tendremos la información sobre un término de la colección. Esta clase será usada en algunos métodos de la clase Index.

Una clase Index que nos permitirá crear el índice y recuperar la información el contiene. También nos almacenará información útil sobre las consultas que hemos realizado.

Una clase llamada SulaIRL, que es la que contiene el main de la aplicación y que almacenará información como el índice que se va a utilizar en la aplicación, el archivo de configuración que nos cargará la colección que se desee y el idioma en el que va a estar la interfaz gráfica durante todo el tiempo que esté abierta.

Una clase denominada ArchivoConf que almacenará la información de una colección almacenada en el sistema y que nos permitirá actualizar la etapa conforme se va realizando el proceso de creación del sistema de recuperación de información y de cargar la configuración de una colección.



Ilustraci3n 23: Diagrama de clases

## 5.2. Diseño de la interfaz

La interfaz de usuario de la aplicación va a estar construida de la siguiente manera donde tenemos lo siguiente ([Ilustración 24](#)):

### Principal:

Será la pantalla de bienvenida a la aplicación. Desde ahí se podrá definir el idioma de la interfaz del programa.

### Configuración:

Esta pantalla será la que nos permita crear colecciones. Para ello da la posibilidad de cargar la carpeta donde se encuentra la colección, darle un nombre y dar la opción de utilizar un archivo de palabras vacías o no y cargar el archivo en caso afirmativo.

También da la posibilidad de cargar una colección que ya exista en la aplicación. Esta permite cargarla desde, como máximo, la última etapa completa que se completó en la construcción del sistema de recuperación.

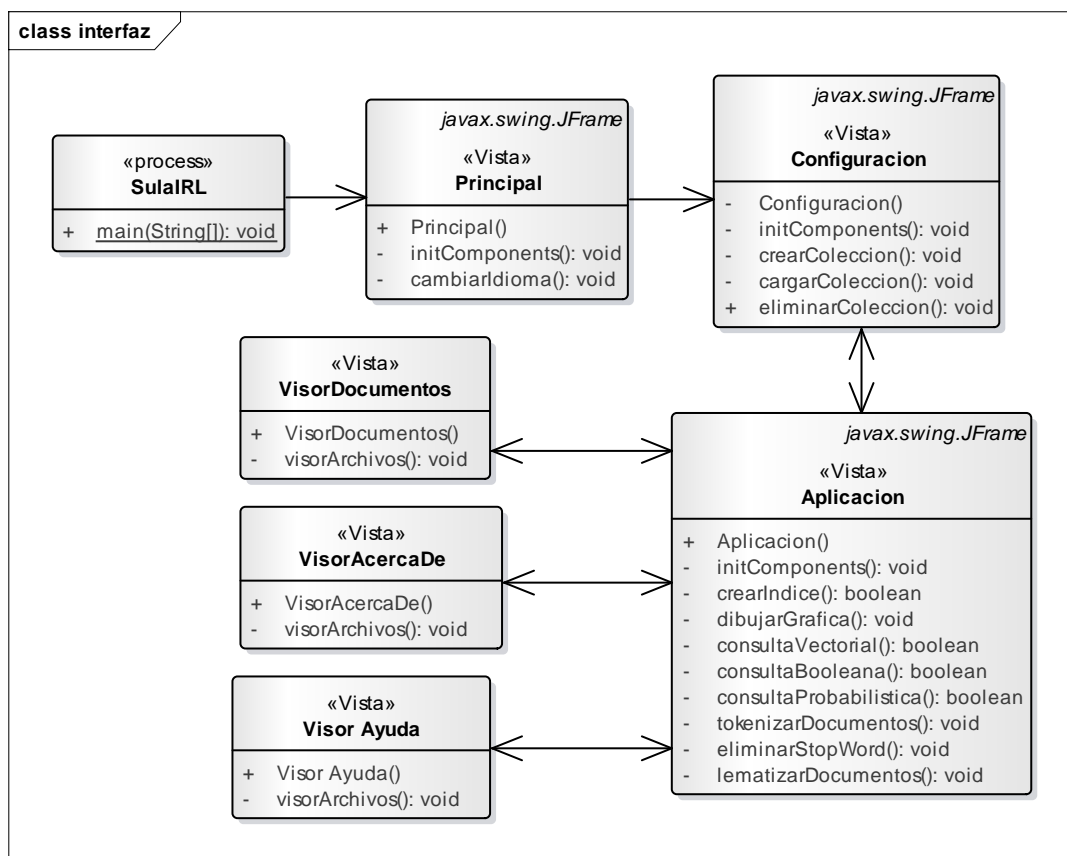


Ilustración 24: Diagrama de interfaz de usuario

### **Aplicación:**

La ventana de aplicación permite visualizar la creación del sistema de recuperación de información. En ella se podrá ver como se realiza el procesamiento de los textos, desde que se realiza la tokenización y se termina con la lematización.

También permite ver cómo queda el índice después de haberse creado, con información sobre los términos y el propio índice. Además, se verán los términos en sus respectivas posiciones en los documentos y se podrá crear unas gráficas donde se vea la distribución de los términos en la colección y las frecuencias de los términos en cada documento, de forma similar a la ley de Zipf.

Por último, se verá el sistema de consultas con el que se podrán hacer las tres diferentes búsquedas, éstas son vectorial, booleana y probabilística. En estas pantallas se mostrarán los resultados de las mismas y los detalles de cada una de ellas.

### **VisorAyuda:**

Pantalla de ayuda sobre los distintos contenidos de la aplicación.

### **VisorAcercaDe:**

Se mostrará la información sobre el autor de la aplicación, el tutor y un enlace a la página web de la aplicación.

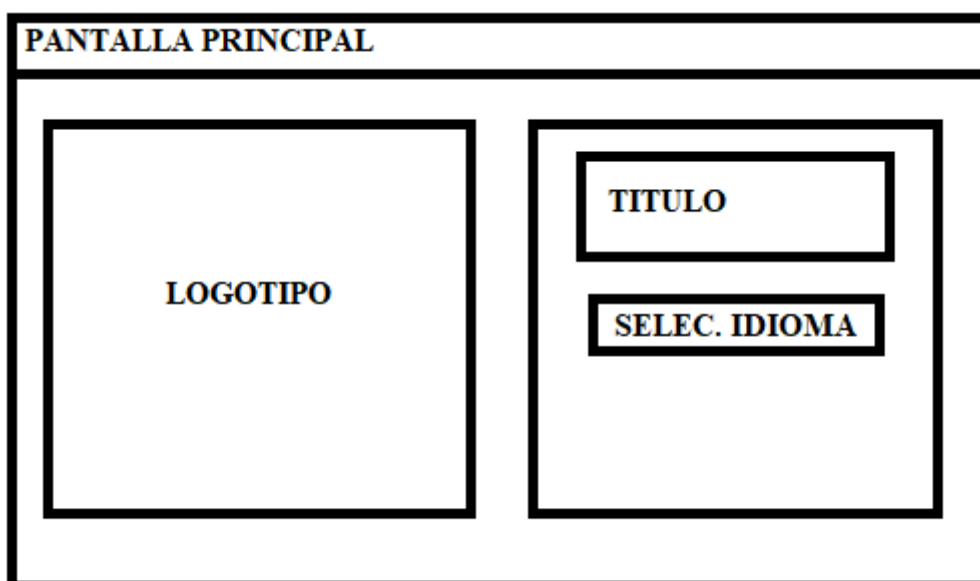
### **VisorDocumentos:**

Esta utilidad de la aplicación permitirá al usuario poder visualizar los documentos originales usados para crear el sistema de recuperación.

La navegación entre las diferentes pantallas se hace de la misma manera que indican las flechas del diagrama mostrado en la [ilustración 24](#).

## **5.3. Prototipos de pantalla**

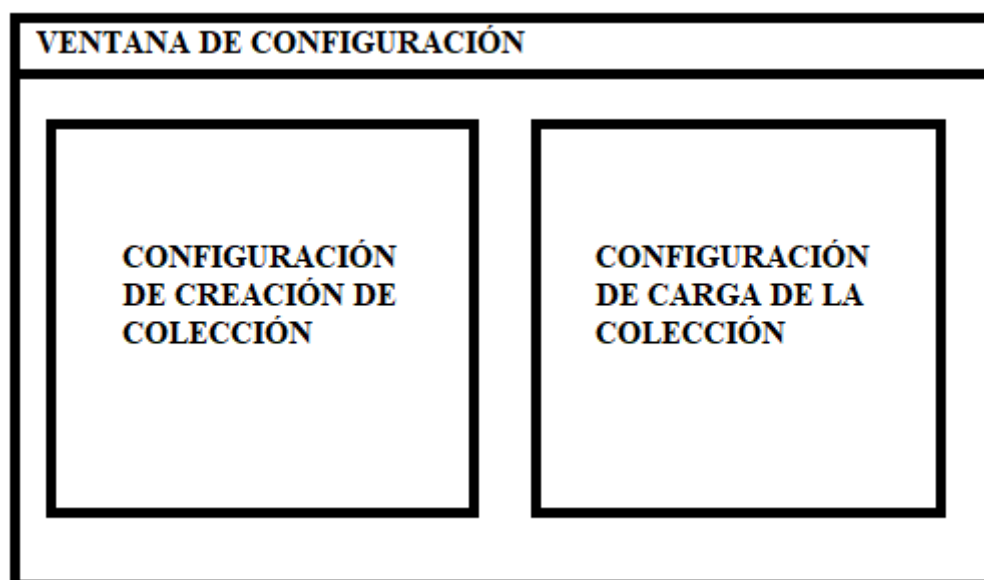
En la pantalla principal tendremos un apartado para el logotipo del programa, el título y la selección del idioma que tendremos durante todo el uso del programa ([ilustración 25](#)).



*Ilustración 25: Pantalla principal de la aplicación*

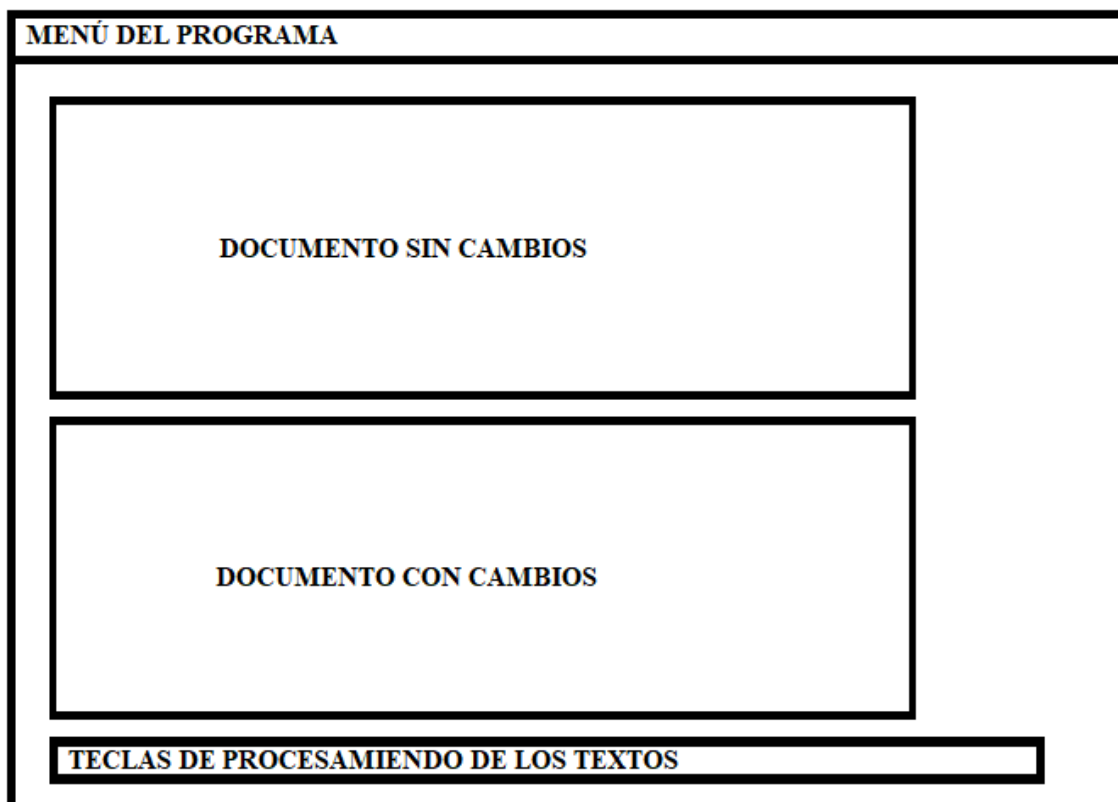
En la pantalla de configuración ([Ilustración 26](#)) se permitirá cargar la carpeta de la colección a utilizar en el proceso de creación del sistema de recuperación de información, establecer su nombre y elegir si se va a utilizar o no un archivo de palabras vacías. Si es afirmativo también se podrá elegir la carpeta donde se encuentra dicho archivo.

Esta pantalla también nos permite cargar colecciones que tengamos ya creadas en la aplicación. Además, nos permite seleccionar la última etapa que se completó para así poder seguir creando el sistema de recuperación de información.



*Ilustración 26: Pantalla de configuración y carga de una colección*

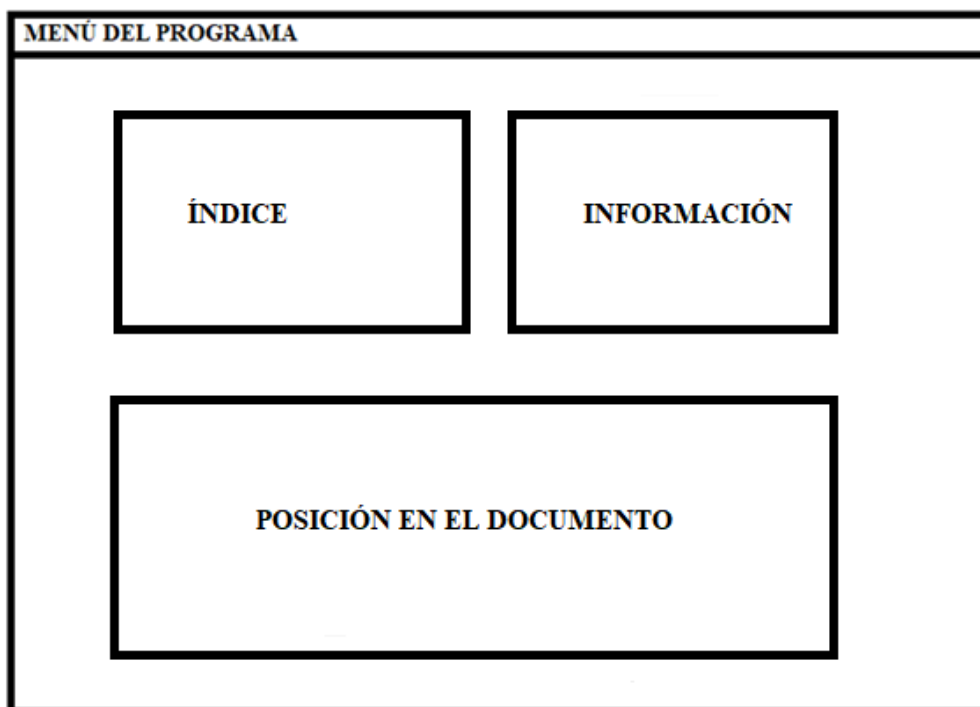
La pantalla de procesamiento de texto será muy similar para la tokenización, la eliminación de stopwords y la lematización. En la parte de arriba se mostrará el documento que se esté procesando en ese instante mientras en la parte de abajo se mostrará cómo va quedando el texto después del procesamiento, viéndose que términos cambian y cuáles no.



*Ilustración 27: Ventana de procesamiento de textos*

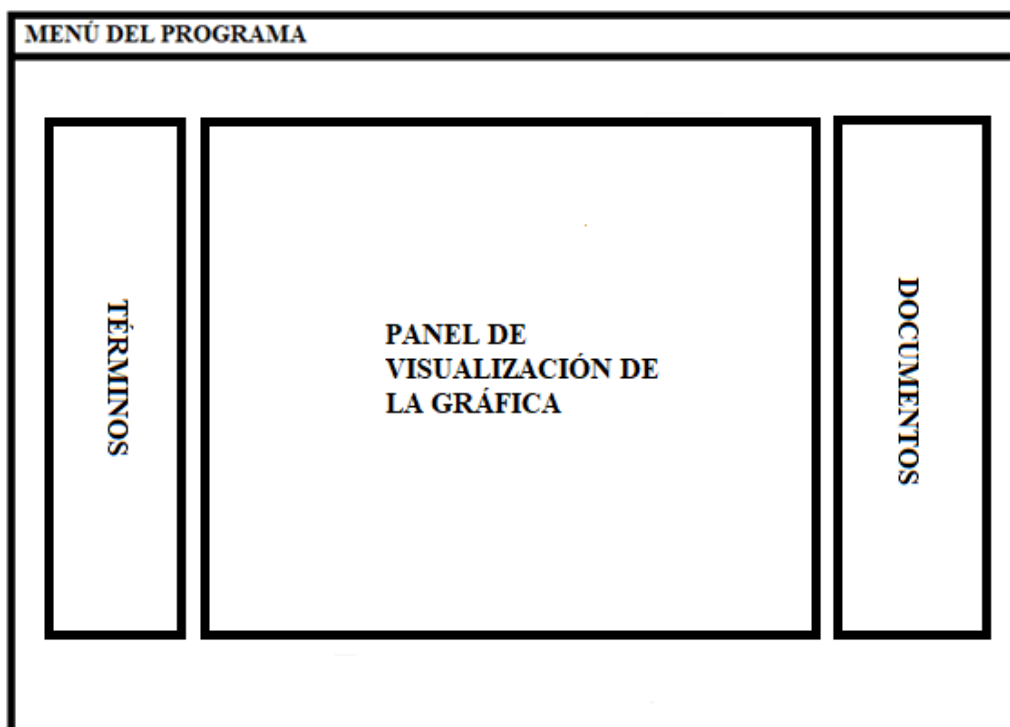
Abajo se ubicarán los botones que cargarán los documentos y nos permitirán ir realizando los respectivos procesos término a término o a documentos completos. También nos permitirá pasar a la siguiente etapa del proceso de creación del sistema de recuperación.

En la parte superior izquierda de la siguiente pantalla se podrá ver el índice generado por Lucene con información referente a cada uno de los términos que lo compone. A su derecha se visualizará información sobre el término seleccionado en la tabla del índice de la izquierda, como es la posición y el documento en el que aparece y también alguna información sobre el índice. Por último, en la parte de abajo se podrá ver la posición del término en el documento que se haya seleccionado en la parte superior derecha.

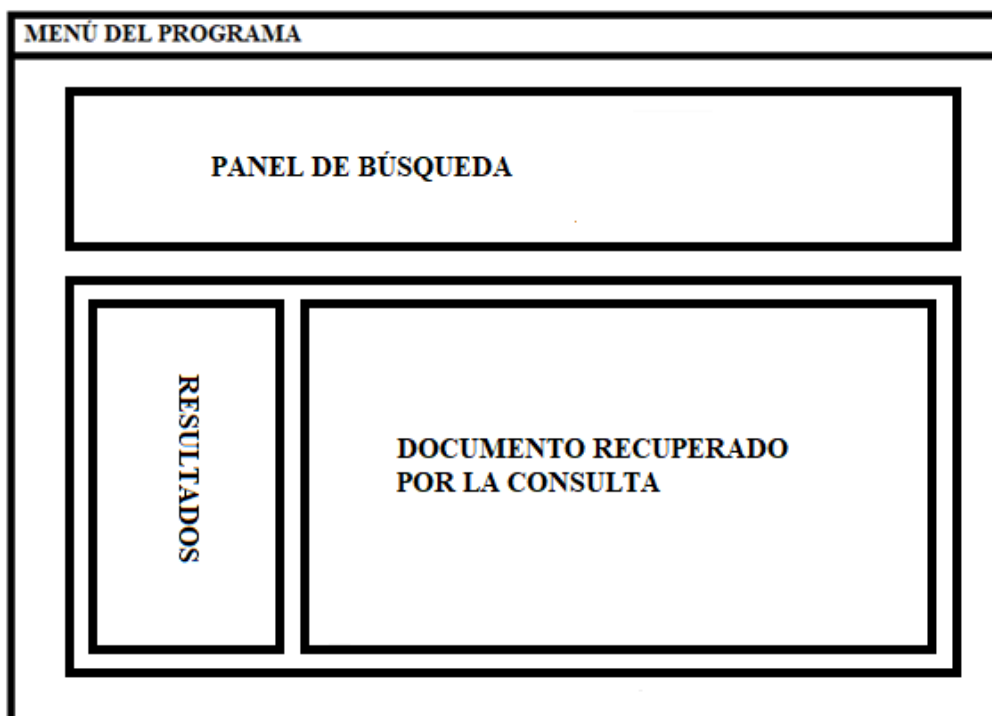


*Ilustración 28: Pantalla de información del índice*

En la siguiente pantalla se podrá ver las gráficas de frecuencias de un término o de un documento. Para cada término se dibujará una gráfica en la que se visualizará la frecuencia de dicho término en cada uno de los documentos en los que aparece. También se podrá ver una gráfica por cada documento. En este caso se podrá ver cómo es la distribución de los términos en cada uno de los documentos, al igual que en la ley de Zipf. Las gráficas se verán en el panel de en medio.



*Ilustración 29: Pantalla de visualización del índice*



*Ilustración 30: Pantalla de realización de consultas y visualización de resultados*

La [ilustración 30](#) será igual para los tres tipos de consultas. En la parte superior de esta pantalla tendremos el cuadro de búsqueda en el que se podrá realizar la consulta y en el cuadro inferior de abajo tendremos los resultados de la misma. En su parte izquierda tendremos los documentos ordenados por su relevancia y en el de la derecha se visualizará el documento con los términos que coinciden con los de la consulta.

El visor de documentos nos permitirá ver los documentos originales que se han utilizado para crear el sistema de recuperación de información. A la derecha se podrá seleccionar el archivo que se desea visualizar y a la derecha se verá el contenido de dicho archivo.



*Ilustración 31: Pantalla de visualización de los archivos originales*



La pantalla de visualización de ayuda estará dividida en dos partes. La parte de la izquierda permitirá seleccionar el contenido del cual se desea obtener información. En la parte derecha tendremos la ayuda referente al contenido seleccionado en la parte de la izquierda. Esta pantalla será de la siguiente manera:



*Ilustración 32: Pantalla de visualización de ayuda*

## 6. Desarrollo

### 6.1. Herramientas utilizadas

En esta sección se dará a conocer todas las herramientas utilizadas para crear el software, así como el lenguaje de programación utilizado y las bibliotecas utilizadas.

#### 6.1.1. Lenguaje utilizado

Para el desarrollo de la aplicación se ha optado por el lenguaje de programación JAVA. Este lenguaje fue desarrollado por Sun Microsystems en 1995 y adquirida por Oracle en 2010. Con él se pretendía crear un lenguaje que fuese parecido a C++ pero que fuese más seguro y utilizara menos recursos.

La gran ventaja de utilizar Java es la independencia del hardware para su uso ya que se hace a través de una máquina virtual que es compatible con sistemas tan diversos como dispositivos móviles, sistemas empujados, en sistemas operativos diferentes como son Windows, Linux o Mac OS. Otras propiedades de Java son las siguientes:

- Es un lenguaje orientado a objetos, por lo que nos permite beneficiarnos de todas las ventajas que esto nos ofrece.
- Permite crear aplicaciones multihebra, por lo que nos permite aprovechar el paralelismo de los procesadores multicore.
- No existen los punteros, por lo que el más fácil realizar una programación sin errores a la hora de utilizar la memoria en algunos tipos de datos.
- Permite crear fácilmente interfaces gráficas, realización de gráficos, programas en modo texto, conexión a bases de datos.
- Posee mecanismos para controlar las excepciones de posibles fallos inesperados.
- El código solo hay que escribirlo una sola vez ya que el compilador crea un archivo .class que es ejecutado en la máquina virtual.
- Existe mucha información para consultas en su documentación.

#### 6.1.2. Entorno de programación

Netbeans es un intérprete que nos permite desarrollar un amplio rango de tecnologías como son Java, PHP, HTML, C y C++, ... siendo compatible con varios sistemas operativos como Windows, Linux o Mac OS.

Es un buen editor de código, multilenguaje el cual se va interpretando conforme se va escribiendo, obteniéndose así los posibles fallos en el código. También nos permite añadir componentes al código tan solo con arrastrar y soltar por lo que nos permite la creación de interfaces gráficas de una manera muy sencilla. Además, también integra un depurador de errores con el que ver el valor de las variables en cada sentencia del programa, controlar los diferentes errores...

Hace más fácil la administración de grandes proyectos gracias a las múltiples vistas y visualización de mismo de manera ordenada según los archivos y carpetas que lo compone.

Otra de las ventajas que ofrece Netbeans es la integración de los módulos. Los módulos contienen clases de Java que se agrupan en API's. Una vez añadida la API a

Netbeans, este lee todas las clases que la compone y hace disponible su uso.

### **6.1.3. Apache Tika [1]**

Apache Tika es una librería software que permite la lectura y extracción de información de archivos de diferentes formatos. Tika nos posibilita la extracción de metadatos de los ficheros y el texto contenido en ellos. En esta librería existen parsers de documentos y herramientas de detección de documentos con los que podemos extraer la información.

La API Tika es muy usada en el desarrollo de motores de búsqueda para indexar el texto de los documentos, análisis de documentos, gestión de archivos digitales, y análisis de contenido.

El parser es la clave para analizar documentos en Tika. Esta interfaz muestra el texto y los metadatos extraídos de un documento y los pone a disposición para su uso. Sus características son las siguientes:

- Uso bajo de memoria: Usa pocos recursos de memoria, lo que hace que sea adecuado para aplicaciones Java o en plataformas de dispositivos móviles.
- Detección de idiomas: Tika incluye utilidades para identificar automáticamente el idioma en el que está escrito el archivo.
- Rápido procesamiento: Realiza la extracción y detección de una forma muy eficiente.
- Tika contiene múltiples bibliotecas dentro de una sola interfaz, por tanto, no tenemos que examinar que biblioteca hay que utilizar para cada uno de los tipos de archivos que se va a utilizar. Añadiendo el archivo de Tika las tenemos todas.

### **6.1.4. JFreeChart [2]**

JFreeChart es una librería que nos permite crear y visualizar fácilmente gráficas de gran calidad en aplicaciones Java. Hoy en día es la biblioteca de gráficos para Java más usada y es mantenida por una diversa comunidad de desarrolladores. Es una API que contiene numerosos tipos de gráficos, cada uno con su documentación y permite la creación de archivos PNG, JPEG, PDF, EPS y SVG.

### **6.1.5. Apache Lucene [3] [4]**

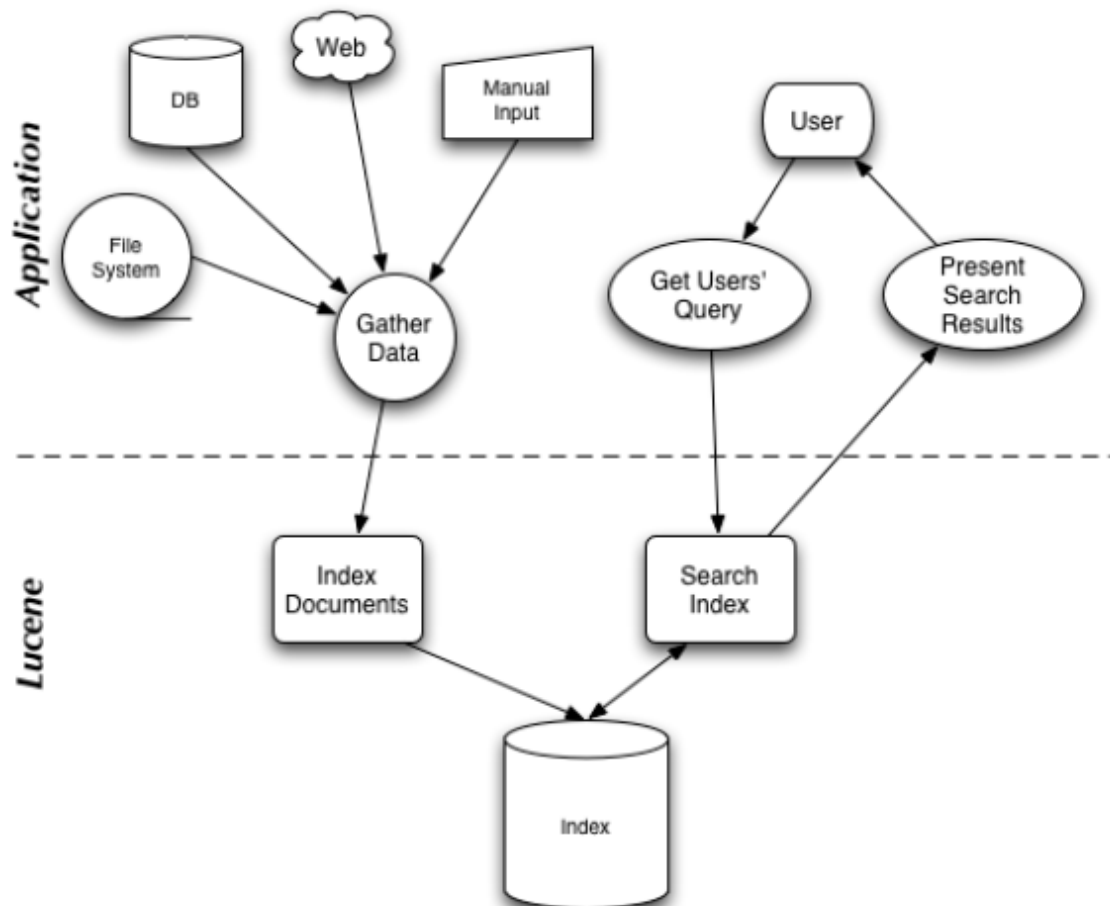
Lucene es una biblioteca de alto rendimiento de recuperación de información. Permite añadir motores de búsqueda e indexación en nuestra aplicación. Soporta diferentes lenguajes de programación como son Delphi, Perl, C++, PHP, Ruby, Python, C#, .Net y Java.

Lucene es software de código abierto con licencia open source desarrollada en Java y mantenida por la fundación Apache. Estas características lo hacen uno de las bibliotecas más populares en la recuperación de información. Proporciona una potente API no requiere un gran conocimiento sobre la indexación y la búsqueda.

Con Lucene se puede implementar las funciones de indexación y de búsqueda en la aplicación. Con el podemos indexar y buscar casi cualquier contenido que pueda expresarse de forma textual.

El funcionamiento de Lucene es simple: se obtienen los documentos ya sea de la web, de una base de datos, de archivos en el disco duro... y crea un índice a partir de ellos para que así el usuario pueda hacer las consultas, estas sean procesadas por Lucene y se ejecuten sobre el índice recuperándose los documentos que coincidan con la consulta.

La integración de Lucene con la aplicación quedaría como muestra la siguiente imagen:



*Ilustración 33: Integración de Lucene en una aplicación. Imagen tomada de [4, p. 8]*

Donde se puede ver que la recogida de los documentos a indexar, la consulta del usuario y la visualización de los resultados es realizada por parte de la aplicación, mientras que la indexación, el almacenamiento del índice y la búsqueda en el índice es ejecutada por Lucene.

El índice de Lucene se compone de una serie de clases fundamentales para su creación. Son las siguientes:

- **IndexWriter:** Es el componente principal en el proceso de indexación. Crea un índice y le añade documentos.
- **Directory:** Representa la ubicación en la que se encuentra el índice de Lucene. Hay dos métodos para almacenar el índice, una a través de `FSDirectory` que crea el índice un directorio del disco duro, y otra a través de `RAMDirectory`, que crea el índice en memoria. El primer método puede ser más robusto pero el segundo caso es más rápido
- **Analyzer:** Se pasa por parámetro al constructor del índice y su misión es la de extraer el texto de los documentos. Lucene implementa numerosos

analyzers para los diferentes idiomas de las colecciones. En algunos de ellos se puede agregar incluso archivos de palabras vacías y también son capaces de realizar la fase de stemming y de convertir los términos de mayúsculas a minúsculas.

- **Document:** Un documento es un conjunto de campos. Este campo representa el documento o los metadatos que posee el documento. En el índice se creará un tipo Document por cada uno de los archivos a indexar.
- **Field:** Un documento puede tener uno o más campos. Cada campo representa una parte del documento de la cual queremos guardar información para poder recuperarla cuando se efectúe una consulta. Lucene ofrece diferentes tipos de *fields* dependiendo de cómo se quiera almacenar la información en el índice (si es literal al dato, si es palabra por palabra, ...).

La interfaz de búsqueda también está compuesta por otras clases que también son fundamentales para recuperar la información del índice. Son las siguientes:

- **IndexSearcher:** Es la clase fundamental para recuperar la información del índice. Esta clase abre el índice en modo solo lectura y ofrece numerosos métodos de búsqueda.
- **Term:** Es la unidad más pequeña por la que se puede buscar. Estos términos están almacenados en los fields de los documentos. Cuando se realiza la búsqueda se debe especificar en qué campo estamos buscando dichos términos.
- **Query:** Es la consulta que se realiza para recuperar los documentos. Hay diferentes tipos de consultas siendo las utilizadas en este proyecto la QueryParser y la BooleanQuery.
- **TopDocs:** Almacena los documentos recuperados por la consulta y sus ponderaciones.

La versión de Lucene utilizada para este proyecto ha sido la 6.3.0.

#### 6.1.6. Luke [5]

Luke es una aplicación de código abierto que nos permite acceder al índice que hayamos creado para comprobar su estructura y funcionamiento y nos permite modificar y visualizar su contenido de las siguientes maneras:

- Buscar información por número de documento o por término.
- Ver documentos y copiarlos al portapapeles.
- Recuperar los términos con un ranking de pesos, determinado por defecto por la similitud coseno.
- Ejecutar búsquedas y ver los resultados.
- Analizar los resultados de las búsquedas.
- Borrar, añadir y modificar documentos del índice.
- Optimizar el índice.

## 6.2. Implementación

### 6.2.1. Carga de archivos

En una primera parte se diseñó la interfaz de creación de la colección en la cual se permite seleccionar la colección a cargar, el nombre de la colección y la opción de utilizar un archivo con palabras vacías o no y, en caso afirmativo, cargarlo.

Una vez diseñada se implementó la carga de los ficheros, que crea la jerarquía de carpetas para cada una de las colecciones que se agregan a la aplicación y las estructuras de datos necesarias para que esto sea posible. La jerarquía de carpetas es la siguiente.

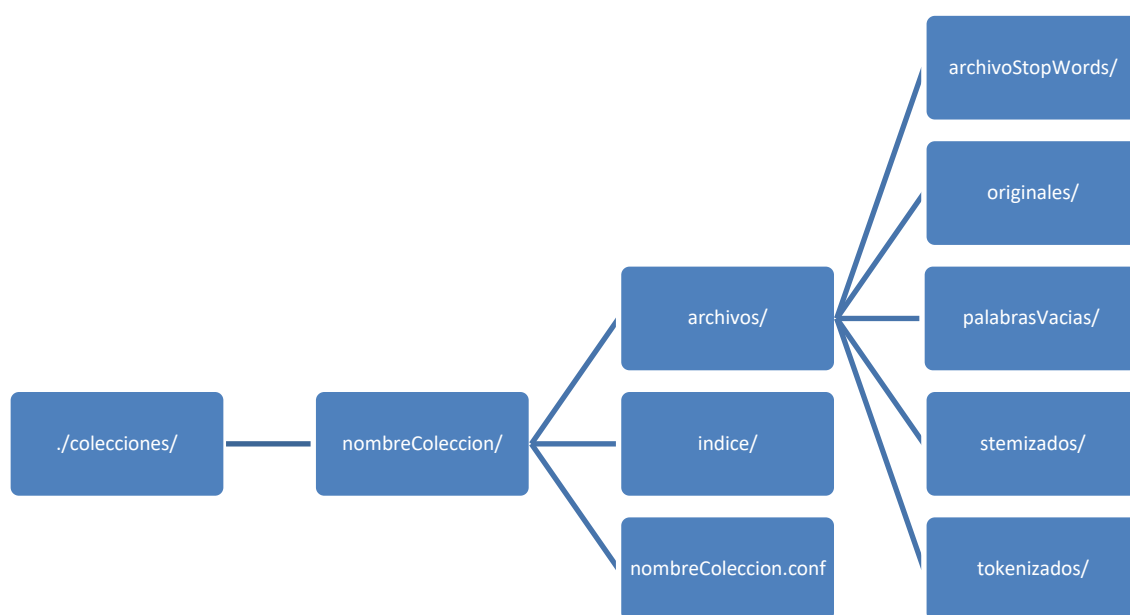


Ilustración 34: Jerarquía de carpetas de las colecciones

Donde:

- *./colecciones/*: Carpeta que se crea en la misma carpeta en la que se está ejecutando la aplicación y que almacena los datos de las distintas colecciones.
  - *nombreColeccion/*: es el nombre que se le da a la colección desde la aplicación.
    - *archivos/*: es la carpeta que guardará todos los archivos de la colección.
      - *archivoStopWords/*: archivo de palabras vacías utilizado.
      - *originales/*: almacena los archivos originales de la colección.
      - *palabrasVacias/*: almacena los archivos a los que se le han eliminado las palabras vacías.
      - *stemizados/*: almacena los archivos con el texto lematizado.
      - *tokenizados/*: almacena los archivos resultantes de realizar la tokenización.

- Índice/: es la carpeta donde se guarda los archivos del índice de Lucene
- nombreColeccion.conf: archivo que guarda información necesaria para cargar una colección en la aplicación

### 6.2.2. Procesamiento de documentos

Una vez cargados todos los documentos en el sistema se da paso al siguiente paso, el procesamiento de los documentos. Este proceso será realizado por cuatro hebras que se repartirán el trabajo, asignándole a cada hebra un conjunto de todos los documentos de la colección. Se ha optado por cuatro hebras ya que casi todos los ordenadores modernos poseen más de un núcleo de procesamiento, siendo la mayoría capaz de controlar cuatro hebras ejecutándose concurrentemente.

Primero se realiza la tokenización de los documentos originales de la colección, después se realiza la eliminación de las palabras vacías, siempre que el usuario haya seleccionado la opción y la lematización de los documentos. Todos estos pasos serán descritos a continuación.

#### *Tokenización*

Recordemos que era el proceso de tokenización. Este proceso consiste en el reconocimiento del texto en los archivos y su estructura. Aquí se leen cada una de las palabras y se eliminan algunas que no tienen sentido de ser almacenadas en el índice, al igual que se hace con los signos de puntuación, se cambian las palabras de mayúscula a minúscula...

En esta aplicación se ha optado por lo siguiente:

**Eliminación de los números:** La decisión de eliminar los números de los documentos es debido a que, en los sistemas de recuperación generales, como son los navegadores web, no se suele buscar por números y por lo tanto no son relevantes.

**Acentos:** En este paso también ha sido eliminados ya que el índice almacenaba la misma palabra tanto si la tenía como si no. Entonces se llegó a la conclusión de que no era necesario tener dos palabras iguales diferenciadas por un acento.

**Transformación de las palabras de mayúscula a minúscula:** La transformación de mayúsculas a minúsculas. Con esto, junto con el paso anterior, permitimos al usuario escribir las palabras en mayúsculas o minúsculas, con tildes o sin tildes, tolerando así algunos errores del usuario al hacer la consulta.

**Caracteres especiales:** Los caracteres especiales como @, #, ©, <sup>2</sup>, <sup>3</sup>, †, /, ..., √, °, ®, =, <sup>a</sup> y otros más encontrados en las colecciones también han sido eliminados ya que carecen de utilidad.

**Signos de puntuación:** Se eliminan los signos de puntuación como guiones, puntos, comas... también han sido eliminados. Las palabras que son separadas por un guion se les elimina el guion y se separan las palabras.

Para realizar la tokenización se ha utilizado el analyzer `StandardAnalyzer()` de `org.apache.lucene.analysis`. Este analyzer permite eliminar los signos de puntuación, así como pasar de mayúsculas a minúsculas el texto. Hay que decir que este analizador no quita todos los elementos descritos anteriormente, por lo que se ha optado a utilizar expresiones regulares para ello.

El resultado de aplicar el analyzer se guarda en un fichero, del mismo nombre que el original, en la carpeta de tokenizados, descrito en la sección anterior. Esto se hace con cada uno de los documentos de la colección.

### *Eliminación de palabras vacías*

La eliminación de palabras vacías es el proceso por el que se eliminan palabras que no tienen significado en ciertos idiomas. Eliminando dichas palabras se consigue reducir considerablemente el tamaño del índice, ya que son palabras que se repiten bastante todos los documentos.

Para la eliminación de este tipo de palabras se parte del archivo generado anteriormente en el paso anterior de tokenización. Para ello hay que utilizar el analyzer de Lucene ubicado en `org.apache.lucene.analysis` y llamar al constructor `StandardAnalyzer()`, al que se le pasa por parámetro el archivo con las palabras vacías que queremos eliminar.

En nuestro caso los archivos de palabras vacías utilizados para el castellano y el inglés son los siguientes:

### *Castellano*

a	estamos	fuésemos	hubiesen	seas	teníamos
al	estamos	fuesen	hubieses	sed	tenían
algo	están	fueses	hubimos	ser	tenías
algunas	están	fui	hubiste	será	tenida
algunos	estando	fuimos	hubisteis	serán	tenidas
ante	estar	fuiste	hubo	serás	tenido
antes	estará	fuisteis	la	seré	tenidos
como	estarán	ha	las	seréis	teniendo
con	estarás	ha	le	seremos	ti
contra	estaré	habéis	les	sería	tiene
cual	estaréis	haber	lo	seríais	tiene
cuando	estaremos	había	los	seríamos	tienen
de	estaría	había	más	serían	tienes
del	estaríais	habíais	me	serías	todo
desde	estaríamos	habíamos	mi	sí	todos
donde	estarían	habían	mí	sido	tu
durante	estarías	habías	mía	siendo	tú
e	estas	habida	mías	sin	tus
el	estás	habidas	mío	sobre	tuve
él	este	habido	míos	sois	tuviera
ella	esté	habidos	mis	somos	tuvierais
ellas	estéis	habiendo	mucho	son	tuviéramos
ellos	estemos	habrá	muchos	son	tuvieran
en	estén	habrán	muy	soy	tuvieras
entre	estés	habrás	nada	su	tuvieron
era	esto	habré	ni	sus	tuviese
erais	estos	habréis	no	suya	tuvieseis



éramos	estoy	habremos	nos	suyas	tuviésemos
eran	estuve	habría	nosotras	suyo	tuviesen
eras	estuviera	habríaís	nosotros	suyos	tuvieses
eres	estuvierais	habríamos	nuestra	también	tuvimos
es	estuviéramos	habrían	nuestras	tanto	tuviste
es	estuvieran	habrías	nuestro	te	tuvisteis
esa	estuvieras	han	nuestros	tendrá	tuvo
esas	estuvieron	han	o	tendrán	tuya
ese	estuviese	has	os	tendrás	tuyas
eso	estuvieseis	hasta	otra	tendré	tuyo
esos	estuviésemos	hay	otras	tendréis	tuyos
esta	estuviesen	haya	otro	tendremos	un
está	estuvieses	hayáis	otros	tendría	una
está	estuvimos	hayamos	para	tendríaís	uno
estaba	estuviste	hayan	pero	tendríamos	unos
estaba	estuvisteis	hayas	poco	tendrían	vosotras
estabais	estuvo	he	por	tendrías	vosotros
estábamos	fue	hemos	porque	tened	vuestra
estaban	fue	hube	que	tenéis	vuestras
estabas	fuera	hubiera	qué	tenemos	vuestro
estad	fuerais	hubierais	quien	tenga	vuestros
estada	fuéramos	hubiéramos	quienes	tengáis	y
estadas	fueran	hubieran	se	tengamos	ya
estado	fueras	hubieras	sea	tengan	yo
estado	fueron	hubieron	sea	tengas	
estados	fueron	hubiese	seáis	tengo	
estados	fuese	hubieseis	seamos	tenía	
estáis	fueseis	hubiésemos	sean	teníaís	

Este archivo ha sido descargado de la página web de Snowball<sup>5</sup>, que es una API para el manejo de cadenas de caracteres en diferentes idiomas.

## Inglés

a	couldnt	he'd	must	she'd	two
about	couldn't	he'll	mustn't	she'll	un
above	cry	hence	my	she's	under
across	daren't	her	myself	should	until
after	de	here	name	shouldn't	up
afterwards	describe	hereafter	namely	show	upon
again	detail	hereby	needn't	side	us
against	did	herein	neither	since	very
all	didn't	here's	never	sincere	via
almost	do	hereupon	nevertheless	six	was
alone	does	hers	new	sixty	wasn't

<sup>5</sup> Página web de Snowball - <http://snowball.tartarus.org/>

along	doesn't	herself	next	so	way
already	doing	he's	nine	some	we
also	done	high	no	somehow	we'd
although	don't	him	nobody	someone	well
always	down	himself	none	something	we'll
am	due	his	noone	sometime	were
among	during	how	nor	sometimes	we're
amongst	each	however	not	somewhere	weren't
amongst	eg	how's	nothing	still	we've
amount	eight	hundred	now	such	what
an	either	i	nowhere	system	whatever
and	eleven	i'd	of	take	what's
another	else	ie	off	ten	when
any	elsewhere	if	often	than	whence
anyhow	empty	i'll	old	that	whenever
anyone	enough	i'm	on	that's	when's
anything	etc	in	once	the	where
anyway	even	inc	one	their	whereafter
anywhere	ever	indeed	only	theirs	whereas
are	every	interest	onto	them	whereby
aren't	everyone	into	or	themselves	wherein
around	everything	is	other	then	where's
as	everywhere	isn't	others	thence	whereupon
at	except	it	otherwise	there	wherever
back	few	its	ought	thereafter	whether
be	fifteen	it's	oughtn't	thereby	which
became	fify	itself	our	therefore	while
because	fill	i've	ours	therein	whither
become	find	just	ourselves	there's	who
becomes	fire	keep	out	thereupon	whoever
becoming	first	last	over	these	whole
been	five	latter	own	they	whom
before	for	latterly	part	they'd	who's
beforehand	former	least	per	they'll	whose
behind	formerly	less	perhaps	they're	why
being	forty	let's	please	they've	why's
below	found	like	put	thick	will
beside	four	long	rather	thin	with
besides	from	ltd	re	third	within
between	front	made	said	this	without
beyond	full	make	same	those	won't
bill	further	many	say	though	would
both	get	may	says	three	wouldn't
bottom	give	me	second	through	yet
but	go	meanwhile	see	throughout	you
by	goes	might	seem	thru	you'd

call	had	mightn't	seemed	thus	you'll
can	hadn't	mill	seeming	to	your
cannot	has	mine	seems	together	you're
cant	hasnt	more	seen	too	yours
can't	hasn't	moreover	serious	top	yourself
co	have	most	several	toward	yourselves
computer	haven't	mostly	shall	towards	you've
con	having	move	shan't	twelve	
could	he	much	she	twenty	

Este archivo es la unión de un archivo de palabras vacías que ya poseía y otro descargado de la página web de Snowball. Se ha realizado la unión ya que existían términos en uno que no existían en el otro.

El resultado de aplicar el analyzer anterior se guardará en la carpeta palabrasVacías descrito al principio de la sección de [carga de archivos](#). Este procedimiento se hace con cada uno de los documentos de la colección.

### *Lematización de documentos*

Recordemos que la lematización del texto de un documento consistía en extraer la raíz que contiene el significado de las palabras para así agruparlas y hacer más efectiva la búsqueda además de hacer el tamaño del índice más pequeño.

Esta fase es dependiente del idioma y por tanto se le debe especificar a la aplicación cuando esta se va a cargar. Para ello, y sabiendo que la aplicación solamente estará preparada para colecciones en castellano o en inglés, se llamará al analyzer de Lucene que nos interpretará dicho idioma. Estos analyzer son los siguientes:

- EnglishAnalyzer() para el inglés y que está ubicado en org.apache.lucene.analysis.en
- SpanishAnalyzer() para el castellano y que está ubicado en org.apache.lucene.analysis.es

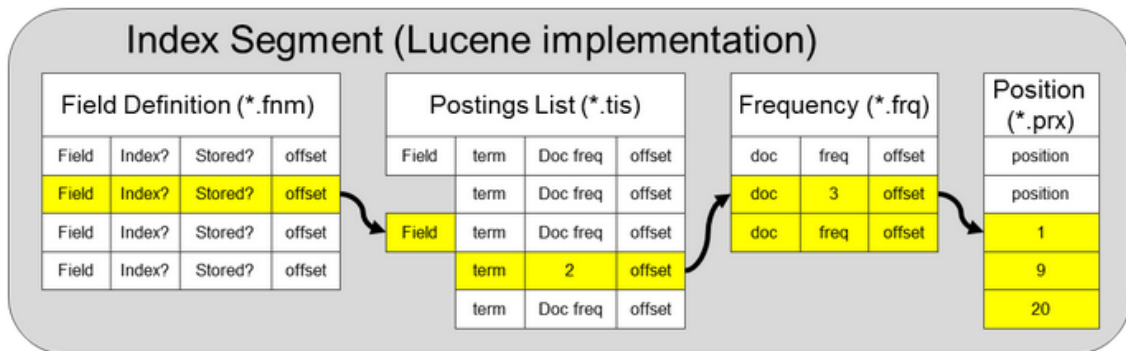
Estos dos constructores también dan la posibilidad de pasar por parámetro el archivo de palabras vacías para realizar su eliminación al mismo tiempo, pero esto no ha sido considerado en el apartado anterior ya que se producía la lematización al mismo tiempo.

Cada uno de los archivos lematizados serán guardados en la carpeta de stemizados descrito en la sección de [carga de archivos](#) descrito en el apartado anterior.

Este proceso se debe de realizar con cada uno de los ficheros que no estén lematizados, ya sean los tokenizados o los archivos a los que se le han eliminado las palabras vacías, dependiendo si se han procesado esto últimos o no.

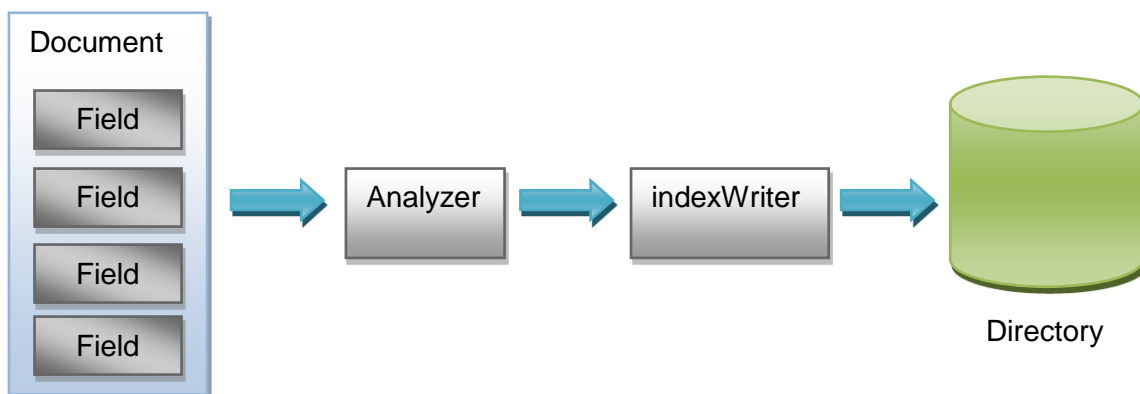
### 6.2.3. Creación del índice

Una vez realizada la etapa de procesamiento del texto ya podemos crear nuestro índice con Lucene. El índice de Lucene se organiza en segmentos y se formarán diferentes segmentos dependiendo del tamaño de los elementos indexados.



*Ilustración 35: Estructura de un segmento de Lucene<sup>6</sup>*

La indexación en Lucene es una funcionalidad básica y fundamental. Para ello debemos de realizar el proceso descrito en la [ilustración 36](#) en la que se crea un objeto de tipo Document que estará compuesto por uno o más objetos del tipo Field. Cada uno de esos Field es un campo que va a guardar información que será recuperable a través de una consulta. A continuación, se crea un objeto del tipo Analyzer para que Lucene lo analice pasándolo a un elemento IndexWriter, que será el que lo escriba en el índice y lo guarde en el Directory. El elemento IndexWriter, como su nombre indica, no sirve para leer del índice, solamente sirve para escribir en él o actualizarlo.



*Ilustración 36: Proceso de indexación*

A continuación, veremos el proceso de indexación utilizado para la aplicación. Primero se mostrará el código para crear el objeto IndexWriter para escribir el índice en el disco y después se verá cómo se añaden los objetos del tipo Document con sus respectivos fields (campos) al índice.

<sup>6</sup> <https://blog.parse.ly/post/1691/lucene/>

```

public void createIndexWriter() throws IOException{
    Path p = Paths.get(col.getRutaColeccion() + "indice/"); // Indicamos la carpeta del índice
    directory = FSDirectory.open(p); // Almacenamos el índice en el directorio
    Analyzer analyzer = new StandardAnalyzer();
    IndexWriterConfig config = new IndexWriterConfig(analyzer);
    iwriter = new IndexWriter(directory, config);
    iwriter.deleteAll();
}

```

Como podemos observar, el método anterior establece el Directory en la carpeta “índice” situado en la ruta de la colección. Acto después creamos el objeto analyzer, y se le pasa por parámetro al objeto IndexWriterConfig, que será el que guarde la configuración que será usada por el objeto IndexWriter.

```

public void createIndex(ArchivosColeccion archivo) throws IOException, SAXException,
TikaException{
    Document doc;
    Metadata metadata = new Metadata();
    ParseContext parseContext = new ParseContext();
    AutoDetectParser parser = new AutoDetectParser();
    InputStream input;
    ContentHandler ch;
    FieldType ft = new FieldType(TextField.TYPE_STORED);
    ft.setStoreTermVectors(true);
    ft.setStoreTermVectorOffsets(true);
    ft.setStoreTermVectorPayloads(true);
    ft.setStoreTermVectorPositions(true);
    ft.setStored(true);

    ft.setIndexOptions(IndexOptions.DOCS_AND_FREQS_AND_POSITIONS_AND_OFFSETS);
    ch = new BodyContentHandler(-1);
    input = new FileInputStream(archivo.getRutaFichero());
    parser.parse(input, ch, metadata, parseContext);
    doc = new Document(); //Creamos un documento
    doc.add(new StringField("rutaFicheroOriginal", archivo.getRutaFichero(),
Field.Store.YES));
    doc.add(new StringField("ficheroStemming", archivo.getRutaFichero(), Field.Store.YES));
    doc.add(new StringField("nombreFichero",
        new File(archivo.getRutaFichero()).getName() , Field.Store.YES));
    doc.add(new Field("text", ch.toString(), ft));
    iwriter.addDocument(doc);
}

```

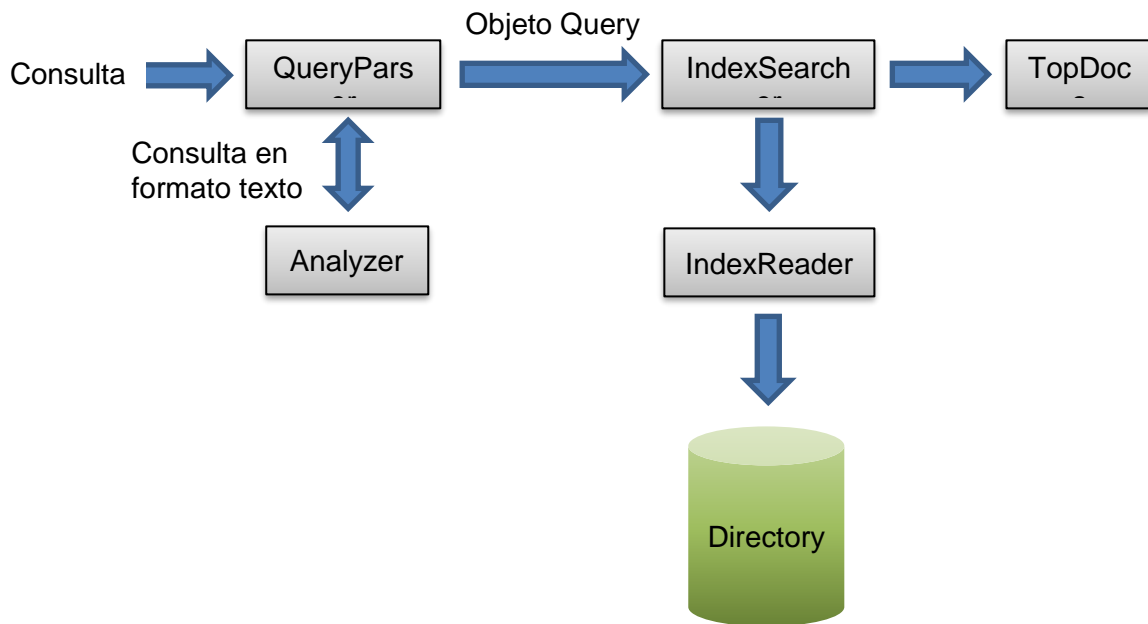
En el código anterior podemos ver como se añaden los documentos a índice. Para ello se crea un objeto de tipo Document y se le añaden los elementos de tipo Field o StringField. En este caso se crea un Document con cuatro Fields llamados rutaFicheroOriginal, ficheroStemming, nombreFichero y text.

Como se puede ver hay unos Fields que son StringField y otro que es FieldType. La diferencia entre ambos es que el primero se almacena en el índice en su totalidad, tal y como es; y el segundo se almacena en el índice como un vector de términos. Por último, vemos cómo se añade el documento al índice mediante la función addDocument().

### 6.2.3. Tratamiento de consultas

Después de crear el índice y de haberlo probado en Luke ya estamos en disposición para utilizarlo. Para ello hemos creado tres tipos de consultas que nos devolverán los resultados mejor puntuados.

El proceso de búsqueda es, junto a la creación del índice, una de las funciones principales de Lucene. El proceso de búsqueda sigue el siguiente esquema:



*Ilustración 37: Procesamiento de consultas*

A continuación, se muestra el código que se ha utilizado en este proyecto para realizar la búsqueda vectorial, basada en el modelo de similitud coseno, y que será explicada a más abajo

```
public ScoreDoc[] queryVect(String q, File stopWords, int numDocumentos) throws
IOException, ParseException{
    createIndexReader();
    Analyzer analyzer = new StandardAnalyzer();
    consVec = q;
    if (stopWords != null){
        analyzer = new StandardAnalyzer(new FileReader(stopWords));
    }
    consVec = procesarStringConsulta(consVec, analyzer);
    if (col.getIdioma().equals("en")){
        analyzer = new EnglishAnalyzer(null);
    }else{
        analyzer = new SpanishAnalyzer(null);
    }
    consVec = procesarStringConsulta(consVec, analyzer);
    searcherVec = new IndexSearcher(ireader);
    searcherVec.setSimilarity(new ClassicSimilarity());
    long startTime = System.nanoTime();
    queryVec = new QueryParser("text", new StandardAnalyzer()).parse(consVec);
    long endTime = System.nanoTime();
    setTimeVec((endTime - startTime)/1000);
```

```

TopDocs hits = searcherVec.search(queryVec, numDocumentos);
return hits.scoreDocs;
}

```

1. El usuario introduce la consulta desde un lugar reservado para ello en la interfaz y es pasada por parámetro a la función anterior.
2. Esa consulta es analizada y sufre las mismas modificaciones que los documentos en el momento del procesamiento. Con ella se crea un objeto del tipo QueryParser que contiene el campo por el que se va a buscar y el analizador de la consulta y que es asignado a un objeto de tipo Query que se utilizará más tarde para buscar documentos en el índice.
3. Después se crea el objeto searcherVec, que es de tipo IndexSearcher, al cual se le ha pasado el objeto ireader (del tipo IndexReader) y se establece a searcherVec el modelo de similitud a utilizar. SearchVec será el que nos busque los documentos en el índice.
4. Se ejecuta el searchVec mediante el método search pasándole por parámetro el objeto del tipo Query y el número de documentos a recuperar. Este último es opcional y si no se introduce ningún valor nos devolverá el máximo número de documentos que cumplan la consulta.
5. La ejecución del objeto del tipo IndexSearch junto al método search devuelve un array del tipo TopDocs que contiene cada uno de los resultados obtenidos por la consulta.

Para la consulta probabilística se actúa de la misma manera, pero cambiando el modelo de similitud utilizado de ClassicSimilarity a BM25Similarity.

Para las consultas booleanas también es similar a la anterior, solamente que no tenemos que establecer similitud, ya que las consultas booleanas devuelven los documentos que la cumplen. También se cambia el objeto del tipo Query por uno del tipo BooleanQuery.

## 7. Manual de uso de la aplicación

Recordemos que SulaIR-L nace por la necesidad de tener una aplicación que muestre de manera visual todo el proceso de creación de un sistema de recuperación de información. Ese proceso de creación debe de comprender desde la toma de documentos hasta la realización de consultas y visualización de resultados. Para ello debe de pasar por todas las etapas intermedias como son el procesamiento de los documentos, la creación del índice y el procesamiento de las consultas.

En este apartado se realizará el manual de uso de la aplicación, detallando cada uno de las pantallas disponibles en la aplicación.

### 7.1. Pantalla inicial del programa

En la pantalla inicial del programa nos encontramos la bienvenida al sistema con el logotipo. En ella podemos seleccionar el idioma que quiere utilizar a lo largo de la aplicación. Una vez elegido se pulsará en entrar para comenzar a crear el sistema de recuperación de información.



*Ilustración 38: Pantalla de bienvenida a la aplicación de SulaIR-L*

### 7.2. Pantalla de configuración de SulaIR-L

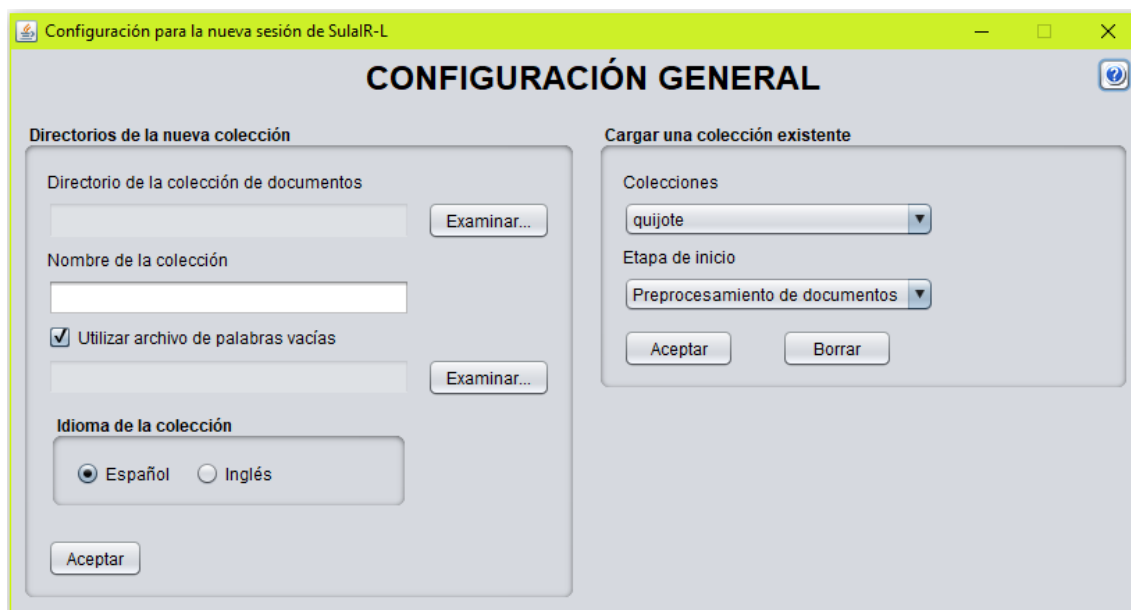
En la pantalla de configuración ([Ilustración 39](#)) tenemos la oportunidad de crear una nueva colección en el sistema o utilizar una ya existente.

#### Creación de una nueva colección

Para crear una colección debemos de seleccionar en el botón de **examinar**, en el dónde pone **Directorio de la colección de documentos**. Ahí seleccionaremos la carpeta donde se encuentran dichos archivos.

Acto seguido pondremos un **nombre** a nuestra colección. Si ese nombre existe la aplicación nos da la oportunidad de eliminarla y crear otra.





*Ilustración 39: Pantalla de configuración de la colección de SulaIR-L*

El siguiente paso es seleccionar si queremos utilizar un archivo para eliminar las palabras vacías. Primero debemos de seleccionar el **checkbox** habilitado para ello y después seleccionar en **examinar**. Se nos abrirá una ventana para buscar el archivo y cargarlo. Este archivo debe estar en **formato txt**.

Por último, para crear una colección nueva tendremos que seleccionar el **idioma de la colección**. Esto es muy importante ya que si no seleccionamos el idioma correcto el analizador no funcionará correctamente y los términos que se indexaran pueden ser erróneos.

### **Carga de una colección existente**

Para la carga de una colección existente tendremos que tener **colecciones ya creadas** en el sistema. Esta colección debe de tener los documentos originales en la carpeta que se utilizó para crear la colección.

Para seleccionar una colección debemos de ir al panel de la derecha y **seleccionar** la que queremos cargar. Después seleccionaremos la **etapa** a partir de la cual queremos realizar el proceso de carga. Hay que tener en cuenta que solo podremos cargar una colección en la última etapa del proceso que hayamos completado. Por ejemplo, si solamente se ha realizado el proceso de procesamiento de textos la aplicación no nos permitirá cargar la etapa de indexación o de consultas.

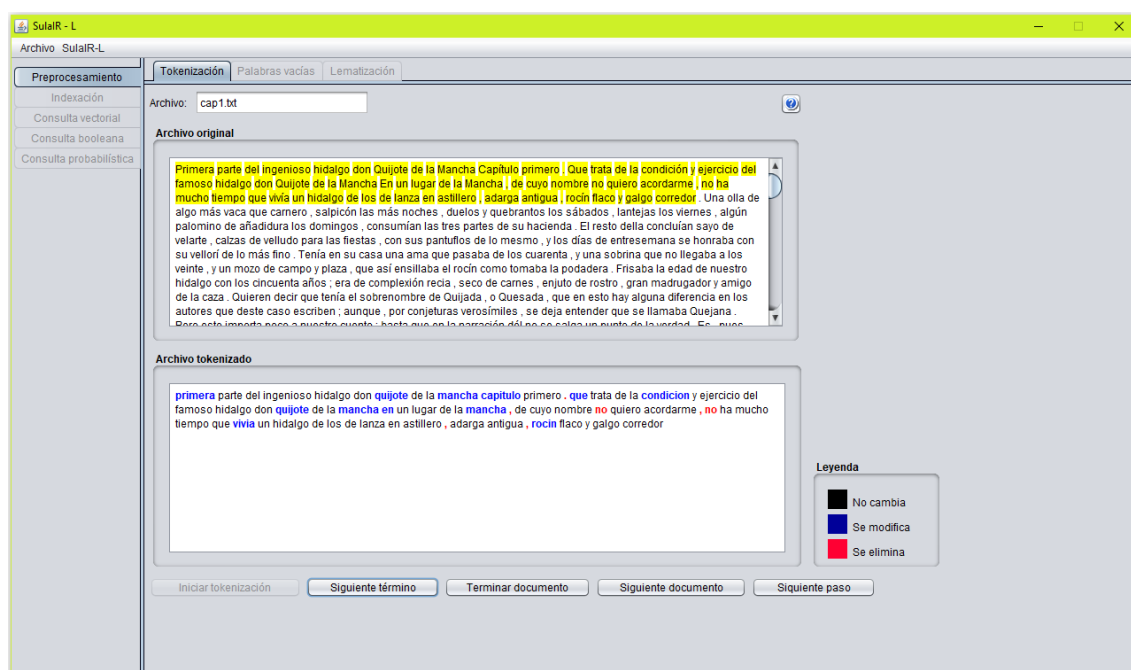
### **Eliminar una colección del sistema**

Si lo que queremos es eliminar una colección del sistema tan solo tendremos que seleccionarla en el panel de la derecha, pulsar sobre el botón de **borrar** y confirmar la acción.

## **7.3. Pantalla del proceso de tokenización**

Una vez cargada y procesada la colección se nos abre la pantalla de tokenización. Esta pantalla nos muestra cómo se realiza el proceso término a término, visualizando en diferentes colores qué términos son los que cambian, cuales se

mantienen y los que se eliminan.



*Ilustración 40: Pantalla del proceso de tokenización de SulaIR-L*

Para iniciar el proceso debemos pulsar sobre el botón de **Iniciar tokenización**. Automáticamente se nos cargará el texto del primer documento a tokenizar en la parte superior de la pantalla. También se nos habilitará los botones inferiores para poder ver los cambios término a término o para mostrar el documento completo. Mediante el botón **Siguiente documento** podremos pasar al siguiente documento de la colección a tokenizar.

Una vez visualizado todos los documentos se puede volver a realizar el proceso desde el primer archivo pulsando en el botón de **iniciar tokenización**.

En **Siguiente paso** podremos pasar a la etapa de eliminación de palabras vacías, en el caso de que hayamos seleccionado en la pantalla de configuración de la colección dicho checkbox. En caso contrario iremos directamente a la fase de lematización.

#### 7.4. Pantalla de palabras vacías

En esta pantalla se nos mostrará cómo se van eliminando las palabras vacías del documento tokenizado. Esta pantalla visualizará que términos de los documentos se eliminan y cuáles no.

Para iniciar el proceso primero debemos presionar sobre el botón Empezar palabras vacías y automáticamente se cargará el primer documento en el cuadro superior y en la parte de abajo se irá viendo el resultado del proceso.

Al igual que en la sección de la tokenización, se puede ir viendo el resultado de la eliminación de las palabras vacías en el cuadro inferior, visualizándose en todo momento que palabras se eliminan y cuáles no en diferentes colores. Esta operación se puede ir viendo término a término o documento a documento, ya pulsemos en el botón de **siguiente término** o **terminar documento**.

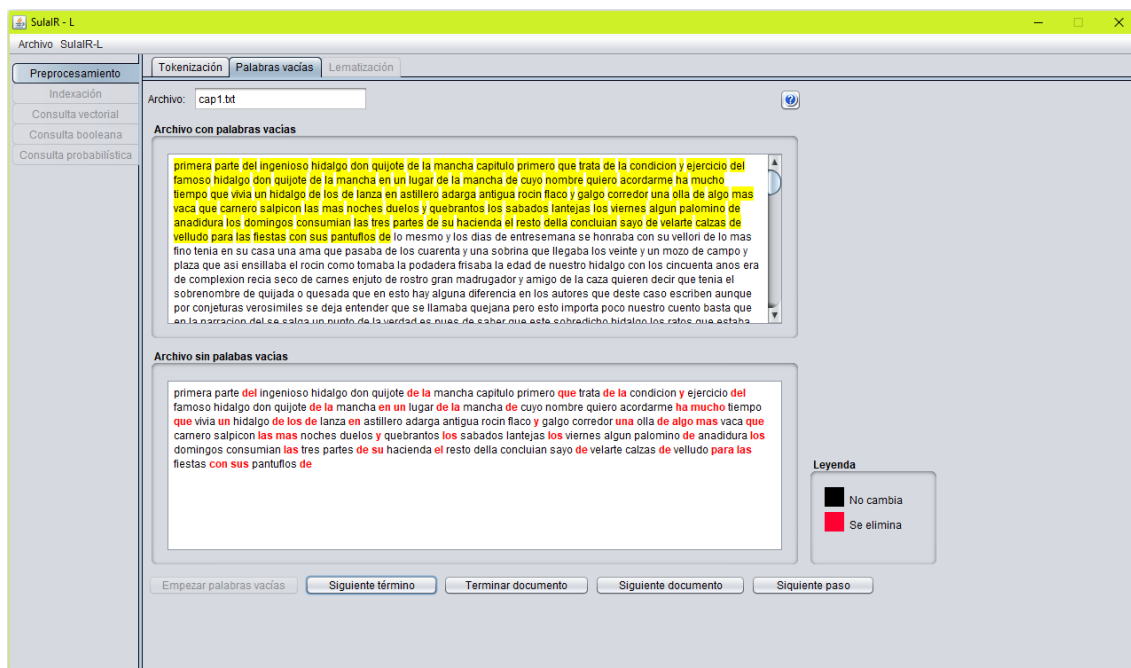


Ilustración 41: Pantalla del proceso de palabras vacías de SulaIR-L

Si queremos pasar al siguiente paso, el de la lematización, tan solo tenemos que pulsar en **siguiente paso** y se nos cargará automáticamente la pantalla del siguiente proceso.

## 7.5. Pantalla de lematización

En la pantalla de lematización se puede diferenciar dos zonas como en la tokenización y la eliminación de palabras vacías. En la parte de arriba se muestra el documento sin haber sido lematizado y abajo, el documento con la lematización realizada.

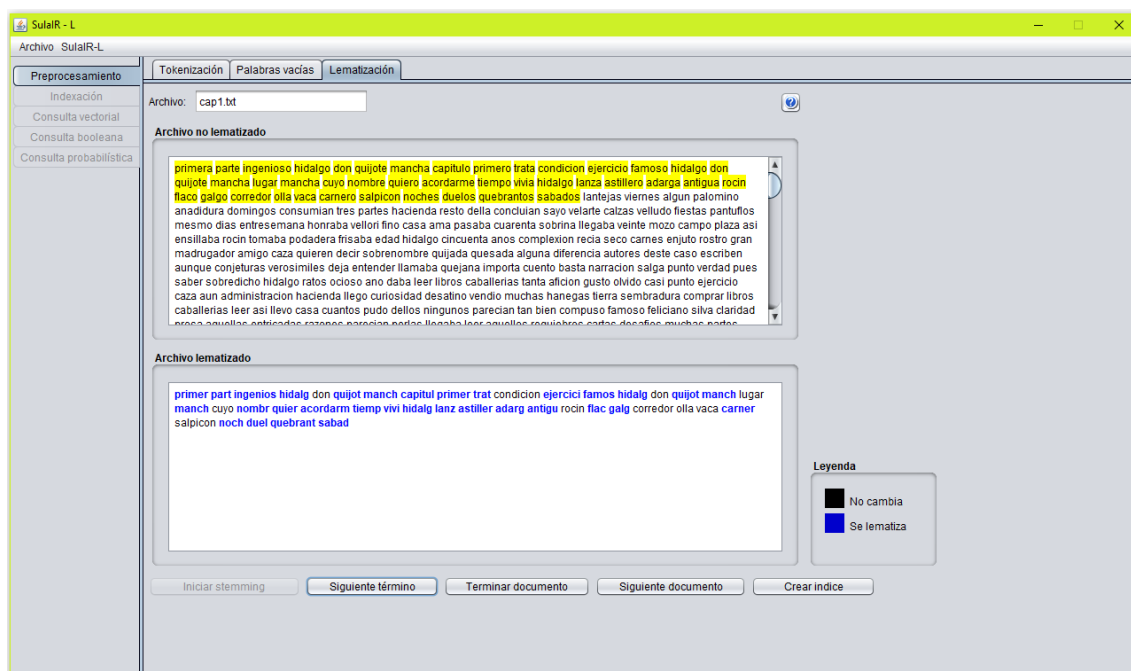


Ilustración 42: Pantalla del proceso de lematización de SulaIR-L

Hay que tener en cuenta qué si al crear la colección se utilizó la opción de utilizar un archivo de palabras vacías o no, ya que, si optó por quitar las palabras vacías, el archivo no lematizado será el resultante de ese procesamiento mientras que, en caso contrario, el archivo no lematizado será el archivo resultante de la operación de tokenización.

Para iniciar la lematización debemos de pulsar sobre el botón **iniciar stemming**. Automáticamente se nos cargará el primer documento de la colección en el cuadro superior. A continuación, podemos ir viendo cómo se lematiza cada uno de los términos pulsando en **siguiente término** o ver el documento completo pulsando en **terminar documento**. Si queremos cargar el siguiente documento tan solo habrá que pulsar sobre el botón **siguiente documento**.

A diferencia que en la pantalla de tokenización y eliminación de palabras vacías esta pantalla tiene un botón llamado **crear índice**. Este botón será el que procese todos los documentos lematizados y agregue al índice de Lucene.

## 7.6. Pantalla del índice

Una vez creado el índice de Lucene se nos abre automáticamente una pantalla con todos los términos que contiene el índice, el número de documentos en los que aparece el término, la frecuencia del término en la colección y su idf.

En la parte derecha del índice se encuentra una tabla que contiene los documentos en los que aparece el término, la posición en el documento, la posición de inicio de la cadena y su posición final.

Justo debajo de esta tabla hay información referente al índice como es la ruta de donde se encuentra, el número de documentos totales de la colección, el número total de términos y la versión del índice.

**Visualización del índice completo**

Rank	Término	Num. Docu...	Frecuencia e...	idf
1	abad	6	6	3,914
2	abadej	1	2	5,167
3	abades	2	2	4,761
4	abai	32	45	2,363
5	abajan	1	1	5,167
6	abajars	2	2	4,761
7	abajen	1	1	5,167
8	abalanzas	1	1	5,167
9	abandonarm	1	1	5,167
10	abarraganad	2	2	4,761
11	abalanar	1	1	5,167
12	abalen	1	1	5,167
13	abald	2	2	4,761
14	abaliend	1	1	5,167
15	abatier	1	1	5,167

**Información del término**

Doc ID	Pos	Start offset	End offset
4	113	782	786
5	780	5126	5130
5	1080	7027	7031
14	561	3650	3654
15	364	2404	2408

**Información del índice**

Ruta del índice: ./colecciones/quijote/indice/  
 Número de documentos: 128  
 Numero de términos: 16119  
 Versión del índice: 5

**Información del término en el documento**

Archivo: cap103.txt Tfx idf 3.3422

senor gobernad...  
 senor atreven famos fuller usar tret pues vici jueg vuelt ejerci comun mejor juego cas principal algun oficial cogen desdichad medi noch abaj desuellan  
 vivo agor escriban dijo sanch decir ileg corchet trai asid mozo dijo senor gobernador manceb veni haci asi columbr justici volvi espald comenz correr  
 gano senal debe algun delinquent parti tras tropez cayo alcanzar jam hui hombr pregunt sanch mozo respondi senor escusar responder much pregunt  
 justici hacen ofici tejedor tej hierri lanz licend buen merced graciosic chocarrer picais bien adond ibad ahor senor tomar aire adond toma aire insul adond  
 sopi buen respondeis proposi discret manceb haced cuent aire sopi popa encamin carcel asid hola llevadi hare duerm alli aire noch par dios dijo mozo  
 asi haga merced dormir carcel hacerm rey pues hare dormir carcel respondi sanch poder prendert sollart cada quisier poder merced dijo mozo bastant  
 hacerm dormir carcel replic sanch llevadi luego vera ojos desengan aunqu alcald quier usar interesal liberalidad pondr pena dos mil ducad deja salir paso  
 carcel cosa risa respondi mozo caso haran dormir carcel cuant hoy vimen dime demoni dijo sanch algun angel saqu quit grill piens mandar echar ahor  
 senor gobernador respondi mozo buen donair razon vengam punt prosupong merced mand llevar carcel echan grill caden meten calaboz ponen alcald  
 grav pen deja salir cumpl mand quier dormir estarm despiert toda noch pegar pestan merced bastant poder hacerm dormir quier ciert dijo secretari  
 hombr salid intencion modo dijo sanch dejareis dormir cosa voluntad contravenir senor dijo mozo piens pues andad dios dijo sanch idos dormir casa  
 dios buen suen quier quitaros! aconsejo aqui adelant burleis justici topareis algun burl casc mozo gobernador prosigui rond alli vinieron dos corchet  
 traian hombr asid dijeron senor gobernador parec hombr sino mujer fea vien vestid habit hombr llegaroni ojos dos tres lantern cuy luz descubrieron rostr  
 mujer parecer diez seis poc anos recogid cabell redeçill oro seda verd hermos mil peiti miraroni amb **80a** vieron venti unas medi seda encamad lig

Ilustración 43: Pantalla de visualización del índice de Lucene en SulaIR-L

En la mitad inferior de la pantalla se encuentra el documento en el cual aparece el término seleccionado y en la posición seleccionada. También se encuentra el valor del  $tf \cdot idf$  de ese término en ese documento. Hay que comentar que para calcular el **idf** se ha utilizado la función en Lucene **idf(...)**<sup>7</sup> de la clase **ClassicSimilarity** cuya fórmula es:

$$idf = 1 + \ln\left(\frac{1 + numDoc}{1 + numDocTerm}\right)$$

Donde:

- *numDoc*: es el número de documentos de la colección
- *numDocTerm*: es el número de documentos en los que aparece el término.
- *ln*: es el logaritmo neperiano.

Para hacer uso de la pantalla hay que **pinchar** en uno de los términos que aparece en la **tabla del índice**. Automáticamente se **cargarán** todos los documentos en los que **aparece el término** en la tabla de la derecha junto con sus **posiciones**, su **start offset** y **end offset**. Al hacer **click** en una de estas **posiciones** aparecerá en el cuadro de abajo el documento con el término seleccionado en la tabla índice, en la posición seleccionada. Además, en la parte superior del documento aparece el  $tf \cdot idf$  del término en el documento. Este valor está calculado usando la clase **ClassicSimilarity** de Lucene y es de la siguiente manera:

$$tf \cdot idf = \sqrt{freqDoc} \cdot idf$$

Donde:

- *freqDoc*: es la frecuencia del término en el documento.
- *idf*: Se calcula de la forma explicada anteriormente.

Para pasar a las consultas solamente tendremos que pulsar sobre el botón siguiente etapa que se encuentra en la parte inferior, a la derecha.

## 7.7. Pantalla de gráficas

Esta pantalla viene a representar como es la distribución de los términos en la colección. Se compone de dos listas, una de términos y otra de documentos.

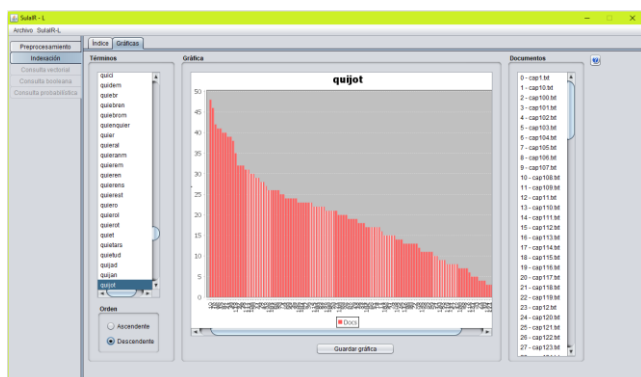


Ilustración 44: Pantalla de visualización de la gráfica de un término de SulaIR-L

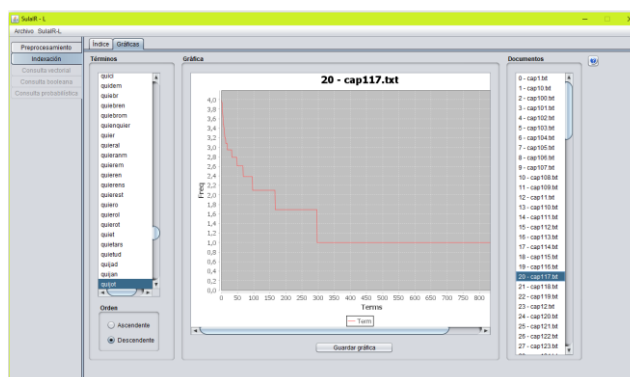


Ilustración 45: Pantalla de visualización de la gráfica de un documento en SulaIR-L

<sup>7</sup> Para ver la información sobre el cálculo del idf ir a [6]

Para dibujar una gráfica tan solo tenemos que hacer click en cualquier término o documento de una lista y aparecerá entre media de las dos listas.

Para las gráficas de términos tenemos la opción de realizarla en orden ascendente o descendente y nos visualiza el número de veces que se repite un término en un determinado documento.

Para la gráfica de documentos tenemos, para cada término, su frecuencia en el documento. Aquí se hace una gráfica como la de la ley de Zipf, donde los términos que más se repiten aparecen primero y los que menos se repiten salen al final. Hay que decir que esta gráfica se encuentra en una escala logarítmica de las frecuencias de los términos.

Esta pantalla también nos da la posibilidad de generar un archivo PNG sobre la gráfica que se esté visualizando en este momento. Al hacer click en guardar gráfica se abrirá un cuadro de dialogo para indicar el lugar donde guardar la gráfica.

## 7.8. Pantalla de búsqueda vectorial

En la pantalla de búsqueda vectorial se nos permite realizar consultas utilizando el modelo de recuperación de información de similitud coseno. Esta medida viene explicada en [6].

Para realizar una consulta debemos de escribirla en el lugar reservado para ello y pulsar **enter** o sobre el botón **buscar**. Además, también podemos establecer el número de documentos como máximo que queremos que Lucene nos devuelva.

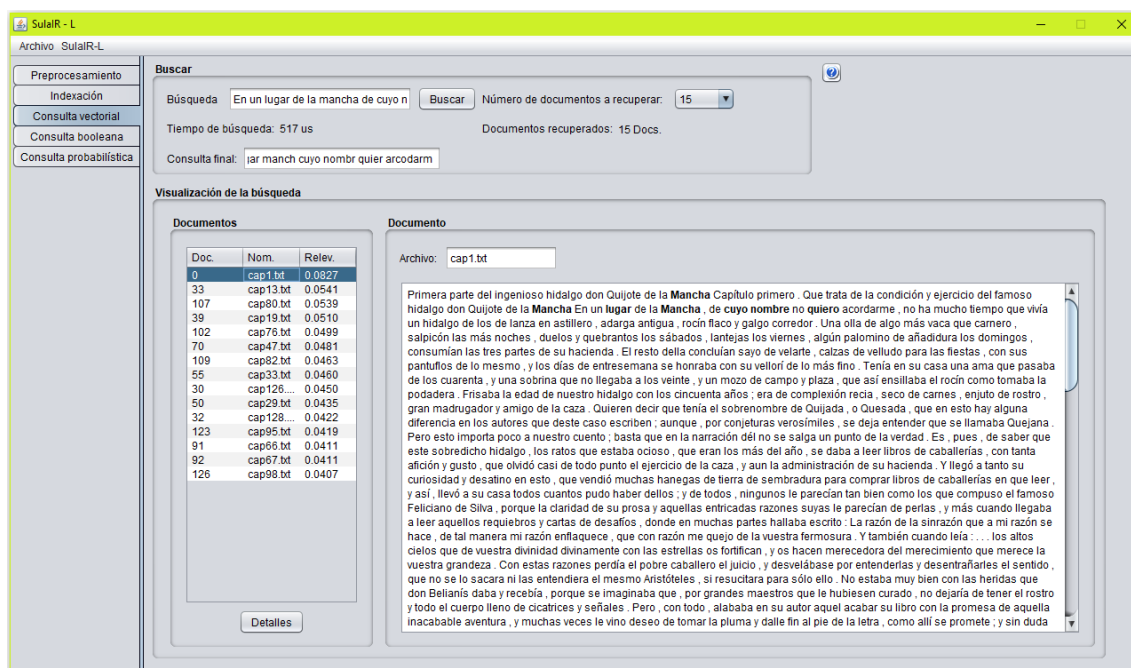


Ilustración 46: Pantalla de búsqueda vectorial de SulaIR-L

Una vez realizada la consulta Lucene nos devuelve el resultado de la misma, el tiempo de ejecución, el número de documentos recuperados y la transformación final de la consulta realizada.



En la tabla de la izquierda aparecen los  $n$ -documentos más relevantes para esa consulta, donde  $n$  es el número de documentos que hayamos fijado o, en caso de ser mayor a los recuperados, los que Lucene ha calculado que son relevantes.

Si seleccionamos alguno de estos documentos se nos cargará a la derecha dicho documento resaltando los términos de la consulta que están en ese documento. También podemos pulsar sobre el botón **detalles** para ver cómo ha calculado Lucene la relevancia para dicho documento. Tendremos una ventana como la siguiente:

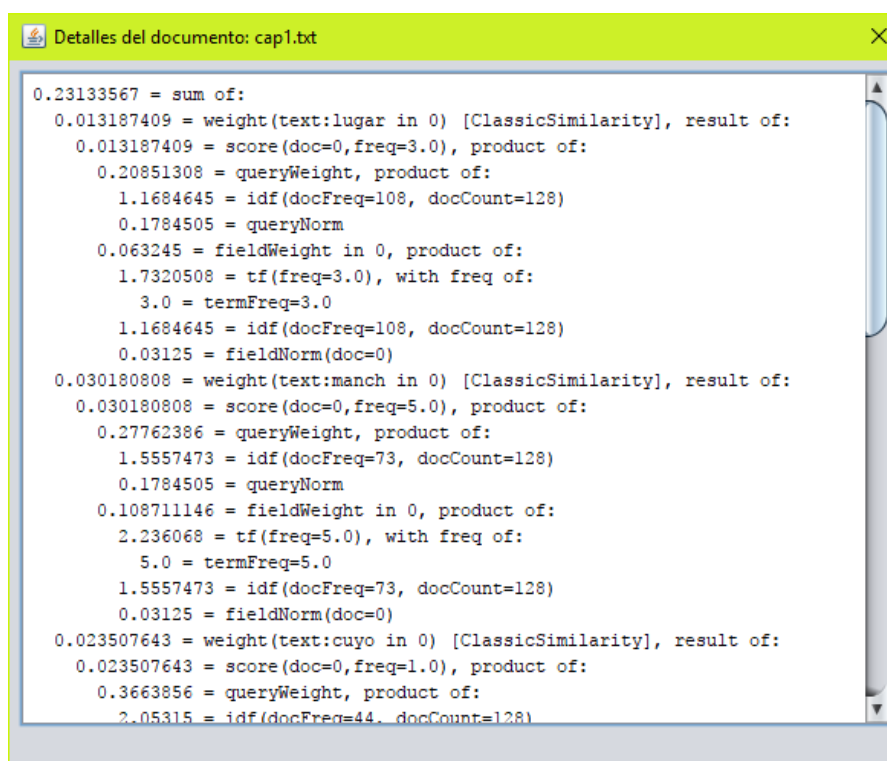


Ilustración 47: Pantalla de detalles de la consulta vectorial en SulaIR-L

## 7.9. Pantalla de búsqueda booleana

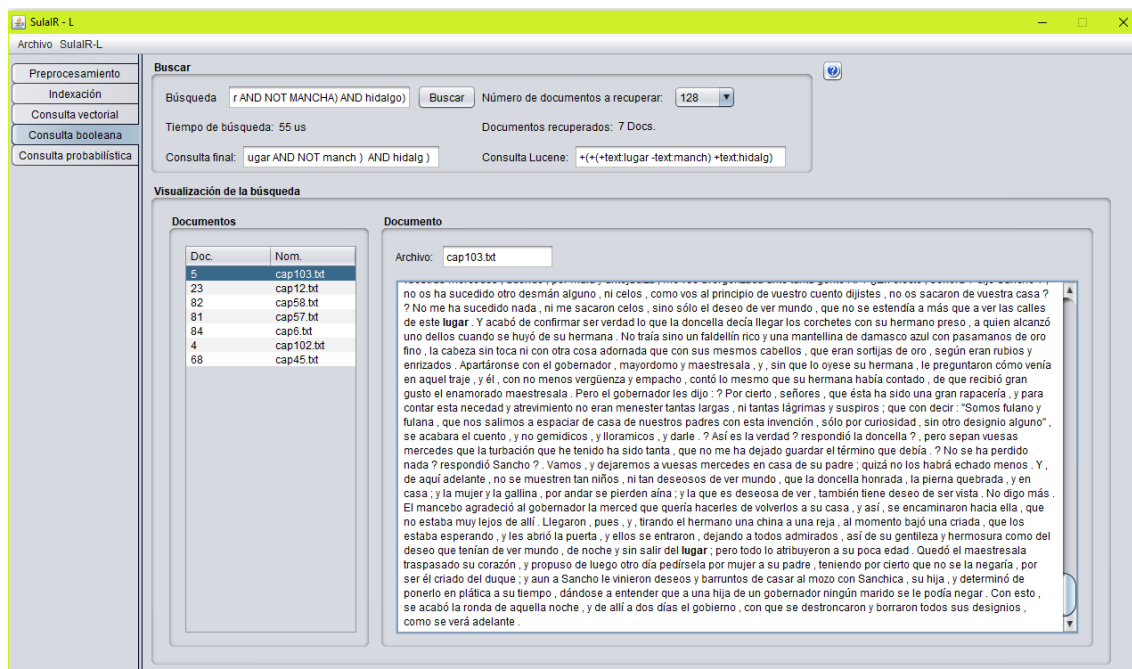
En esta pantalla podremos realizar consultas del tipo booleano en Lucene. Para ello debemos de introducir una consulta del tipo booleano en el lugar reservado para ello y pulsar **enter** o en el botón **buscar**. Esta consulta deberá utilizar correctamente los operadores lógicos **AND**, **OR**, **NOT** y los **paréntesis**.

Al realizar una consulta de este tipo solamente obtendremos los documentos que la cumplan ya que en las consultas booleanas se cumple la condición o no se cumple.

Además de la consulta también podemos seleccionar el número máximo de documentos que deseamos recuperar.

Una vez realizada la consulta obtendremos, además de los documentos que la cumplen, el tiempo de ejecución, los documentos totales recuperados, la consulta ejecutada al realizarse el procesamiento y la consulta que ejecuta Lucene donde utiliza los operadores +, - y los paréntesis.

En la tabla de la izquierda aparecen los  $n$ -documentos que cumplen la consulta, donde  $n$  es el número de documentos que hayamos fijado, pudiendo ser menos si no hay suficientes documentos que cumplan la consulta.



*Ilustración 48: Pantalla de consulta booleana en SulaIR-L*

Una vez obtenida la lista de los documentos que cumplen con la consulta lógica podremos seleccionar uno y cargarlo en el cuadro de la derecha, remarcando el término correspondiente en la consulta.

## 7.10. Pantalla de búsqueda probabilística

La pantalla de búsqueda probabilística es similar a la de la consulta vectorial, pero en vez de utilizar el modelo de similitud coseno se utilizar el modelo probabilístico BM25.

Como en ese caso también tenemos un cuadro en el cuál introducir la consulta. Para ejecutarla solamente tendremos que pulsar en el botón **buscar** o pulsar en **enter**. En este caso también tenemos dos parámetros de los cuales dependen el ranking del modelo probabilístico como es el grado de saturación y de normalización.

La normalización determina como la componente  $tf$  del término cambia cuando se incrementa la frecuencia del término en el documento. Experimentalmente se ha observado que en 1.2 se obtienen mejores resultados. La normalización es el grado en que se normaliza la longitud de los documentos en la colección, comportándose mejor cuando vale 0.75.

En esta pantalla también podemos seleccionar el número de documentos que queremos que la consulta recupere como máximo.

En la tabla de la izquierda aparecen los  $n$ -documentos más relevantes para esa consulta, donde  $n$  es el número de documentos que hayamos fijado o, en caso de ser mayor a los recuperados, los que Lucene ha calculado que son relevantes.



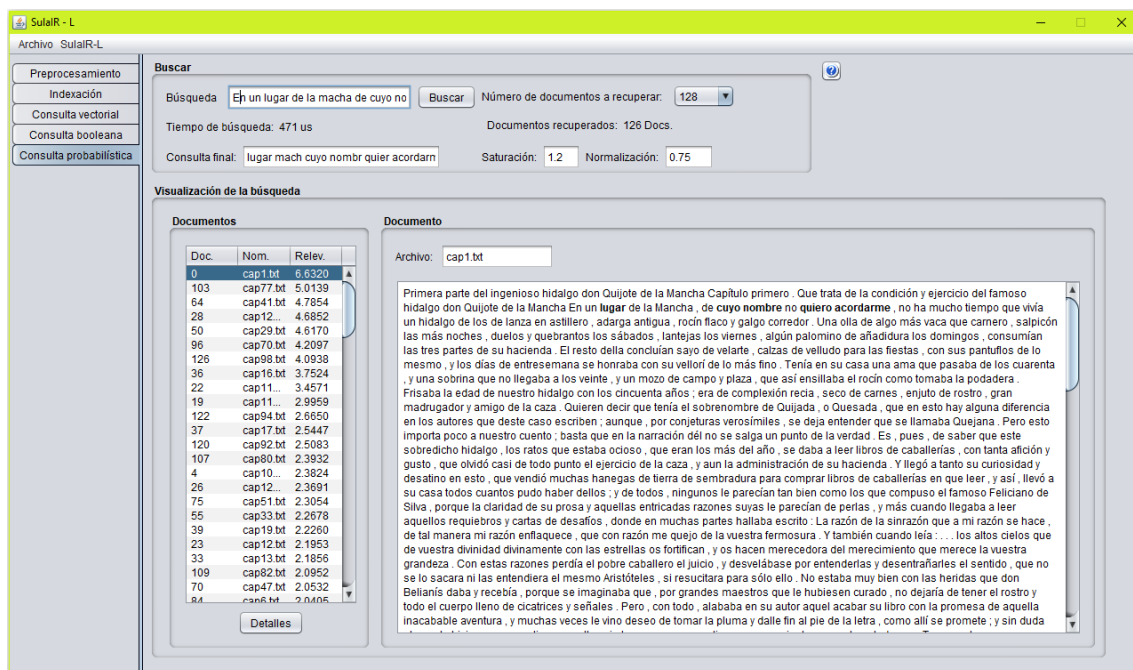


Ilustración 49: Pantalla de consulta probabilística en SulaIR-L

Si seleccionamos un documento de la tabla automáticamente se cargará dicho documento en el cuadro de la derecha resaltando los términos que coinciden con los de la consulta.

También podremos ver las operaciones que ha llevado a Lucene a recuperar esos documentos haciendo click en el botón de **detalles**. Automáticamente se nos abrirá una ventana con la descripción de las estimaciones como la siguiente:

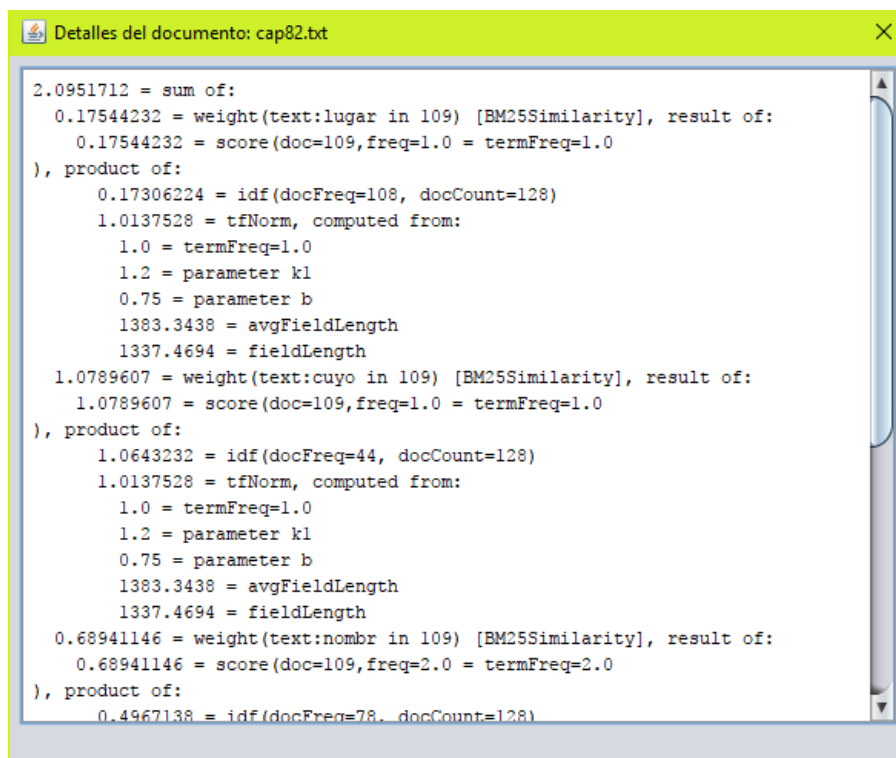


Ilustración 50: Pantalla de detalles de la consulta probabilística en SulaIR-L

## 7.11. Pantalla de visor de documentos

La pantalla del visor de documentos muestra todos los documentos utilizados para crear la colección.

A la izquierda podemos ver la lista de todos los documentos de la colección y otra más abajo que nos muestra el archivo de palabras vacías, si lo hemos utilizado. Una vez seleccionemos uno de estos archivos se cargará en la derecha, siempre que estos sean HTML, htm, txt o xml. Para los archivos pdf se abrirán con la aplicación predeterminada en el sistema operativo.

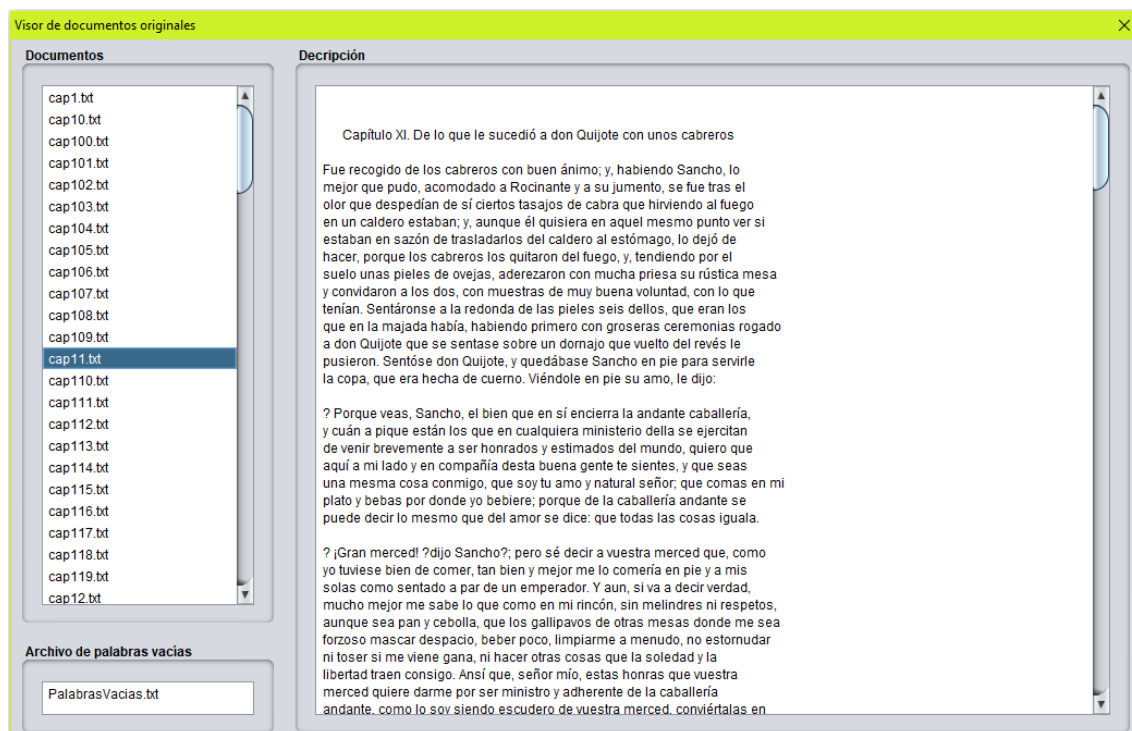


Ilustración 51: Pantalla de visor de documentos en SulaIR-L

## 7.12. Pantalla de ayuda

La pantalla de ayuda hace una pequeña explicación sobre en qué consiste cada una de las pantallas que nos podemos encontrar en la aplicación. Para hacerla visible solamente debemos de ir a la barra del menú superior y pulsa sobre SulaIR-L y hace click en ayuda.

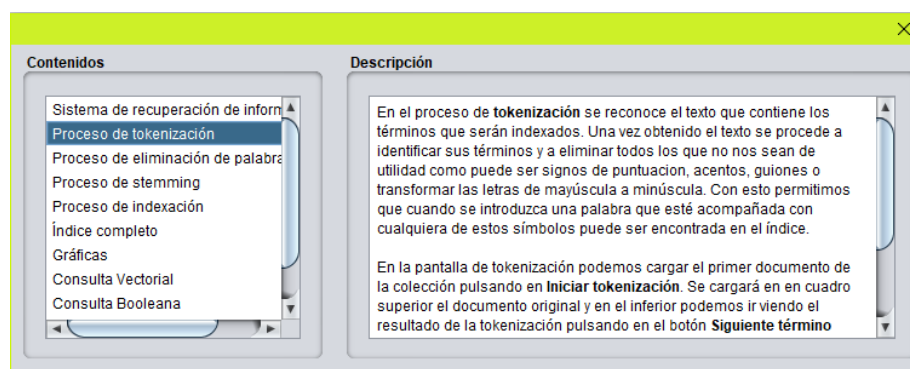



Ilustración 52: Pantalla de ayuda en SulaIR-L

Acto seguido se nos abrirá la ventana anterior ([ilustración 52](#)) en la que podemos seleccionar el tema de ayuda que deseemos y será explicado en el panel de la derecha. Además de ayudar a comprender de qué trata el contenido seleccionado también nos explica cómo utilizar la pantalla correspondiente a ese contenido.

Además de abrir la ayuda desde el menú superior también podemos abrirla todas y cada una de las pantallas en la que nos encontremos en ese momento haciendo click en el icono .

### 7.13. Pantalla de acerca de...

En la pantalla de acerca de... se encuentra la información sobre el autor de esta aplicación y del tutor que ha seguido este trabajo fin de carrera. Además, también encontramos un botón que nos lleva directamente a una web donde podemos encontrar esta aplicación para ser descargada y este manual de uso para el usuario. Esa web es <http://bahia.ugr.es/~sulairl>

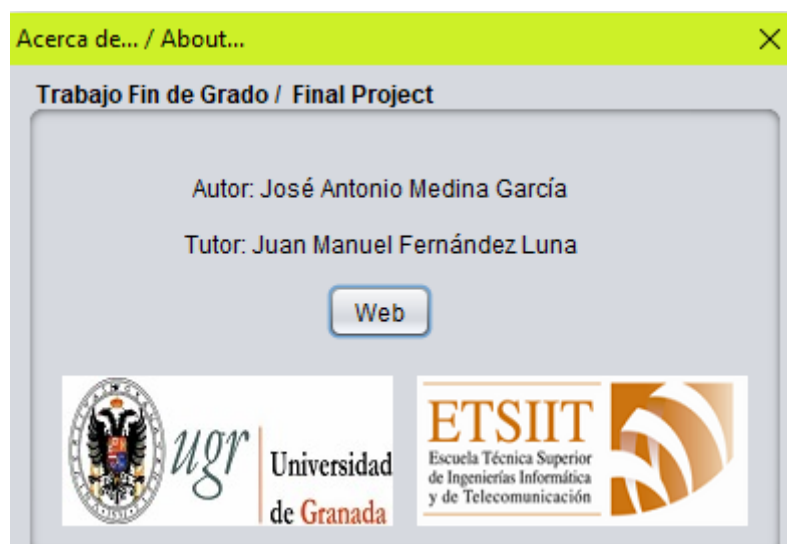


Ilustración 53: Pantalla de acerca de... en Sulair-L

## 8. Conclusiones finales y trabajo futuro

### 8.1. Desarrollo temporal real

En la [ilustración 54](#) se podrá observar el tiempo real empleado en realizar la aplicación. Como se puede observar se ha empleado más tiempo de lo que se creía debido a los numerosos problemas en la implementación, sobretodo en la codificación de la etapa de procesamiento de textos. El problema es que el proceso de tokenización en Lucene no elimina símbolos, caracteres especiales ni pertenecientes a otros idiomas como el chino. Por esto, había que ir buscándolos documento por documento, eliminando los que más se repetían y los que daban algún tipo de error usando expresiones regulares.

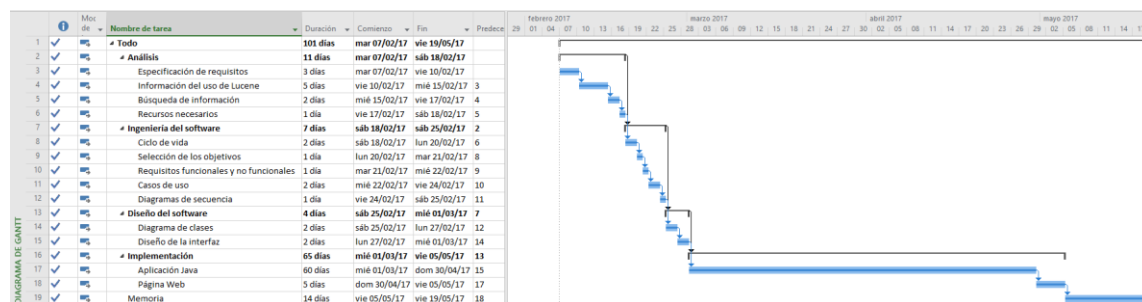


Ilustración 54: Diagrama de Gantt real

Por último, también se decidió realizar una página web donde establecer la ayuda en línea del programa y desde la que se pudiese descargar el archivo .jar de la aplicación.

### 8.2. Conclusiones

El objetivo principal de esta aplicación era el de construir una herramienta que ayudara al aprendizaje del proceso de creación de un sistema de recuperación de información utilizando la API de Lucene. Habiendo terminado el proyecto se puede decir que:

- Se han alcanzado los objetivos que se planteaban. El usuario que utilice esta aplicación comprenderá cuales son los procesos a seguir para la construcción de un sistema de recuperación de información, desde el procesamiento de textos, pasando por la estructura del índice y hasta el procesamiento de las consultas.
- Todo este proceso se ha creado en una interfaz fácil de utilizar, siendo totalmente clara la forma en la que se realiza cada uno de los pasos a seguir, disponiendo en todo caso un menú de ayuda al usuario en cada una de las pantallas del estado del proceso.
- El sistema es portable a cualquier otro sistema operativo que tenga soporte a una Máquina Virtual Java (JVM), por tanto, el objetivo de que se pudiese ejecutar en el mayor número de sistemas operativos sin tener que reescribir código se ha conseguido.

- Se ha cumplido el objetivo de que la colección pudiese ser en español o en inglés gracias a la utilización de los analizadores de Lucene.
- Se consigue que la aplicación pueda tratar los archivos de texto más utilizados hoy en día, como son los archivos en formato txt, pdf, HTML, htm y doc.
- Se pueden realizar consultas de tres tipos diferentes de modelos. Estos modelos son los más utilizados hoy en día por los sistemas de recuperación de información y son el vectorial, cuya medida de similitud es el coseno, el booleano, que mediante una consulta booleana recupera los documentos que la cumplen, y la BM25, que utiliza procedimientos probabilísticos para recuperar los documentos más relevantes.
- Se ha optimizado el procesamiento de los textos todo lo posible utilizando para ello las ventajas de la programación paralela. En este caso se ha establecido a cuatro el número de hebras ya que la mayoría de los procesadores de hoy en día son capaces de soportarlo.
- Se ha conseguido la internacionalización de la interfaz ya que se podrá utilizar tanto en español como en inglés.

### 8.3. Trabajo futuro

Cómo trabajos futuros se plantean los siguientes:

- Un sistema que fuese capaz de tratar más tipos de documentos. Para ello habría que interpretar la estructura de cada uno de ellos. Esos documentos pueden ser imágenes, archivos de audio, video...
- También se podría hacer una versión web SulaIR-L si se dispone de los medios adecuados, como es una buena conexión de banda ancha a la que se pueda subir colecciones de archivo de gran tamaño, un servidor capaz de procesar todos los documentos de un gran número de usuarios, almacenar para cada uno de esos usuarios las colecciones que crean en la aplicación...
- Habilitar al programa de una función que compruebe cuando un archivo es añadido a la carpeta de la colección y que automáticamente lo procesara y lo añadiera al índice. Al igual que cuando se eliminase un documento, la aplicación detectaría el documento que se ha eliminado y lo quitaría del índice y de la colección de documentos a tener en cuenta en la etapa de procesamiento de textos.
- Implementar la retroalimentación por relevancia. El usuario haría una consulta y se examinaría que elementos de los recuperados son relevantes, se seleccionaría esos documentos y se reformularía la consulta con el objetivo de recuperar otros documentos que se adapten mejor a la consulta del usuario.
- Tener una fase de evaluación de los rankings que computase al menos alguna estadística básica (del tipo MAP, P@K, etc) usando colecciones clásicas en la recuperación de información como son CISI, CACM o MED.

#### 8.4. Valoración personal

La elaboración de este proyecto me ha permitido aprender a organizarme para realizar un proyecto de tamaño considerable y a mejorar el modo de programar en proyectos futuros.

También me ha permitido aprender mucho sobre la programación en JAVA que, aunque ya tenía conocimientos previos, me ha hecho mejorar en la estructuración de un proyecto, en el conocimiento de otras API's que facilitan la programación en este lenguaje, el uso de la memoria y la eficiencia de una aplicación y profundizar más en el funcionamiento y uso de Lucene.

Finalizar este proyecto me ha hecho ver que puedo realizar proyectos de gran tamaño y que en un futuro puedo, incluso, afrontar otros mayores, que funcionen de manera más eficiente gracias a la experiencia que he obtenido con este.

Pero sin duda, la satisfacción más grande para mi es saber que este proyecto puede ser usado para la enseñanza de las asignaturas de recuperación de información y que podrá mostrar, de manera más intuitiva y fácil, los distintos procesos de los que se compone la creación de un sistema de recuperación de información.

## 9. Anexo

### 9.1. Código fuente de la aplicación

El código fuente de esta aplicación puede ser consultado y descargado del siguiente repositorio en GitHub:

- <http://github.com/joanmega/SulalR-L>

### 9.2. Descarga de la aplicación y ayuda

El archivo .jar de la aplicación se encuentra en la misma página en la que se puede encontrar la ayuda en español e inglés, y es la siguiente:

- <http://bahia.ugr.es/~sulair/>

Para ejecutar dicha aplicación podemos ejecutar directamente o escribiendo en la consola lo siguiente:

```
java -jar SulalR-L.jar
```

## 10. Bibliografía

- [1] Apache Software Foundation, «Apache Tika,» 2017. [En línea]. Available: <https://tika.apache.org/>.
- [2] Object Refinery Limited, «JFreeChart,» [En línea]. Available: <http://www.jfree.org/jfreechart/>.
- [3] Apache Lucene, «Lucene,» [En línea]. Available: <https://lucene.apache.org/core/>.
- [4] O. Gospodnetic y E. Hatcher, Lucene in action, Manning Publications Co., 2005.
- [5] Luke, «Luke - Lucene Index Toolbox,» [En línea]. Available: <https://code.google.com/archive/p/luke/>.
- [6] Apache Lucene, «TFIDFSimilarity,» 2017. [En línea]. Available: [https://lucene.apache.org/core/6\\_3\\_0/core/index.html?org/apache/lucene/search/similarities/ClassicSimilarity.html](https://lucene.apache.org/core/6_3_0/core/index.html?org/apache/lucene/search/similarities/ClassicSimilarity.html).
- [7] W. B. Frakes y R. Baeza-Yates, Information Retrieval: Data Structures & Algorithms., Prentice Hall, 1992.
- [8] J. F. Huete Guadix, J. M. Fernández Luna y F. Cachela Seijo, Recuperación de información: Un enfoque práctico y multidisciplinar., Ra-Ma, 2011.
- [9] Wikipedia, «Netbeans - Wikipedia,» 2017. [En línea]. Available: <https://en.wikipedia.org/wiki/NetBeans>.
- [10] Sun Microsystems, «Netbeans,» 2017. [En línea]. Available: <https://netbeans.org/>.
- [11] R. Baeza-Yates y B. Ribeiro-Neto, Modern Information Retrieval, Pearson, 1999.
- [12] W. B. Croft, D. Metzler y T. Strohman, Search engines: Information retrieval in practice, Pearson, 2010.
- [13] Wikipedia, «Ley de Zipf - Wikipedia,» 2017. [En línea]. Available: [https://es.wikipedia.org/wiki/Ley\\_de\\_Zipf](https://es.wikipedia.org/wiki/Ley_de_Zipf).
- [14] C. Larman, UML y patrones: Introducción al análisis y diseño orientado a objetos, Pearson, 2002.
- [15] J. A. M. García, «Sulair-L,» 2017. [En línea]. Available: [bahia.ugr.es/~sulairl](http://bahia.ugr.es/~sulairl).
- [16] w3ii, «Lucene Tutorial,» 2017. [En línea]. Available: <http://www.w3ii.com/es/lucene/default.html>.
- [17] Apache Lucene, «Apache Lucene 6.3.0 Documentation,» 2017. [En línea]. Available: [https://lucene.apache.org/core/6\\_3\\_0/](https://lucene.apache.org/core/6_3_0/).