



Gestión de Información en la Web

Desarrollo de un Sistema de Recuperación de Información con la biblioteca Lucene

Irene Béjar Maldonado

76066018G

irenebejar@correo.ugr.es

9/3/2020

Índice

Colección usada	3
Estructura del código	4
Indexador	4
Constructor	5
Close	5
CreateDocument	5
IndexDocuments	5
Motor de búsqueda	6
Constructor	6
SearchByTitle	6
getTitles	6
SearchByAbstract	6
getAbstracts	7
Interfaz	8
Gráfica	8
Por consola	11
Uso de los jars	12
Referencias	13

1. Colección usada

Cómo colección para usar el Sistema de recuperación de Información se ha usado una de las proporcionadas en Prado, concretamente *NSF Research Awards Abstracts 1990-2003*.

Esta colección contiene 129.000 documentos con resúmenes sobre los *NSF Awards* entre los años 1990 y 2003. Para la realización de la práctica se han seleccionado los **cien primeros** documentos de la primera parte de la colección.

La organización de los documentos está hecha mediante títulos y tabulaciones, cada título corresponde a un campo concreto. Entre los disponibles están:

- Título del documento
- Tipo
- Fecha
- Nombre del archivo
- Número del premio
- Sponsor
- Dinero recibido por el premio
- Resumen

Para simplificar la práctica se va a usar solo el **título** y el **resumen**, aunque el uso del resto de campos también es factible.

2. Estructura del código

El código de la práctica está estructurado en tres archivos:

- `Indexer.java` → Contiene el indexador del sistema.
- `SearchEngine.java` → Contiene el motor de búsqueda.
- `App.java` → Contiene las interfaces de usuario.

Para el control de las dependencias del proyecto se ha usado **Maven**, por lo que es necesario tenerlo instalado en la máquina si se quiere ejecutar el código fuente. Junto con el código se entregan **2 jars** correspondientes a la interfaz gráfica y la interfaz por consola de la práctica.

2.1. Indexador

Para implementar el indexador se ha usado la siguiente referencia: ("Lucene Tutorial - Index and Search Examples - HowToDoInJava" n.d.)

Los elementos utilizados para programar el indexador han sido:

- **EnglishAnalyzer** → Analyzer que ofrece la biblioteca Lucene para trabajar con textos en inglés. Ofrece una lista de las StopWords más comunes en inglés y aplica las transformaciones básicas a los textos (*stemming*, pasar a minúscula, etc), a parte de algunas extra como reconocimiento de palabras clave. ("What's the Difference between Lucene StandardAnalyzer and EnglishAnalyzer?" n.d.).
- **Directory** → Clase de Lucene que nos permite indicar el directorio donde queremos crear el índice.
- **Document** → Clase con la que se forman los documentos que se quieren almacenar para su posterior recuperación.
- **IndexWriter** → Clase que crea el índice a partir de los documentos que le pasemos.

La clase del indexador contiene 4 métodos. A continuación se detallará el uso de cada uno.

2.1.1. Constructor

Crea el índice en el directorio que se pasa como argumento y lo configura para que se use el EnglishAnalyzer.

2.1.2. Close

Permite cerrar el indexWriter una vez se haya terminado de construir el índice.

2.1.3. CreateDocument

Crea un documento a partir del título y un texto (en este caso el campo *resumen* de los documentos) que se pasan como argumentos. A los dos argumentos se les pasa el EnglishAnalyzer, ninguno se almacena como texto plano.

2.1.4. IndexDocuments

Abre el directorio donde se encuentra la colección de documentos y lee de cada uno de ellos el campo título y el campo resumen, de cada uno de ellos. Después se crea el documento que se insertará en el índice.

2.2. Motor de búsqueda

Para la implementación del motor de búsqueda se ha usado la siguiente referencia: ("Lucene Tutorial - Index and Search Examples - HowToDoInJava" n.d.)

Para programar el motor de búsqueda se han usado los siguientes elementos:

- **IndexReader** → Permite al buscador leer los documentos del índice.
- **IndexSearcher** → Nos permite buscar documentos en el índice que se le proporcione.
- **TopDocs** → Estructura de datos que contiene los documentos recuperados por el buscador.
- **EnglisAnalyzer** → Para procesar el texto que se va a guardar.
- **Directory** → Para indicar dónde se encuentra el índice
- **Document** → Para poder recuperar los documentos que se encuentran indexados.

Los métodos implementados para esta clase son los siguientes:

2.2.1. Constructor

Crear el buscador a partir del índice que se encuentre en el directorio que se pasa como argumento.

2.2.2. SearchByTitle

Busca en el campo *título* de los documentos. Devuelve los 5 primeros que satisfagan el criterio de búsqueda. El texto que se quiere buscar se pasa como argumento.

2.2.3. getTitles

Devuelve un ArrayList con los Strings de los títulos que se encuentren en el resultado de una búsqueda.

2.2.4. SearchByAbstract

Busca en el campo *resumen* de los documentos. También devuelve los 5 primeros resultados. El texto que se quiere buscar se pasa como argumento.

2.2.5. getAbstracts

Devuelve un ArrayList con los Strings de los resúmenes que se encuentren en el resultado de una búsqueda.

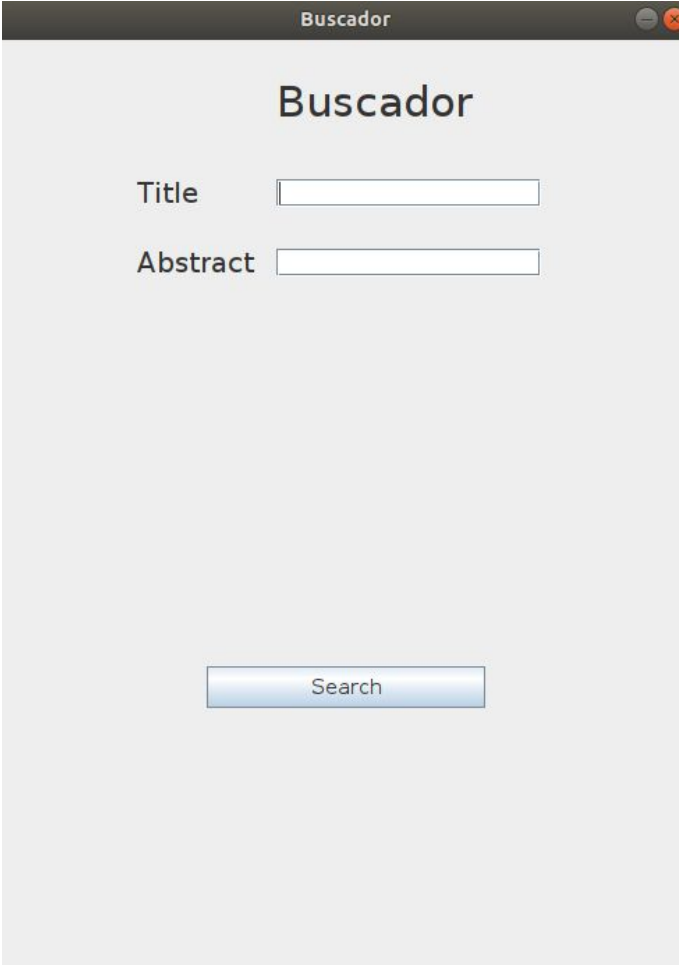
2.3. Interfaz

Para la interfaz del sistema se proporcionan dos interfaces: la primera una interfaz gráfica programada con la biblioteca JavaSwing, la segunda una interfaz por consola que realiza la misma funcionalidad.

2.3.1. Gráfica

La interfaz gráfica que se ha implementado consta de tres ventanas: una para la búsqueda y dos para mostrar resultados.

La primera ventana que nos encontramos al iniciar la aplicación es el **formulario de búsqueda** (“Java Swing | Simple User Registration Form - GeeksforGeeks” 2019):

A screenshot of a Java Swing window titled "Buscador". The window has a light gray background and a dark gray title bar with standard window controls. The title "Buscador" is centered at the top. Below the title, there are two text input fields. The first field is labeled "Title" and the second is labeled "Abstract". Both labels are positioned to the left of their respective input boxes. At the bottom center of the window, there is a blue button with the text "Search".

Formulario de búsqueda

En el formulario nos encontramos dos campos que nos permiten buscar por título o por resumen. Solo hay que rellenar el campo por el que se quiera realizar la búsqueda y pulsar el botón de “Search”. Después de realizar esta acción iremos a la pantalla de resultados.

Si se quiere probar la interfaz se pueden usar las siguientes cadenas como prueba:

- Para buscar por título “Genetic Diversity”.
- Para buscar por resumen “Commercial exploitation”.

Después de realizar la búsqueda pasamos a la **pantalla de resultados** (“A Simple JScrollPane for a JList Component : Scrollpane « Swing JFC « Java” n.d.):



Pantalla de resultados

Aquí se muestran los títulos de los documentos que satisfacen el criterio de búsqueda. Si hacemos doble click sobre algunos de los títulos o pulsamos enter, se nos muestra en una nueva ventana el texto del documento.

Si queremos volver al formulario de búsqueda sólo tenemos que pulsar el botón “Back”.

Por último tenemos la ventana donde se nos muestra el **texto del documento**:



Texto del documento

Aquí se muestra el contenido del documento. Tiene un botón "Back" para volver a la lista de resultados de la búsqueda.

2.3.2. Por consola

La interfaz por consola es muy simple y ha sido utilizada mayormente para realizar pruebas rápidas. A continuación se muestra un ejemplo de uso. Se han resaltado las entradas que debe realizar el usuario.

Para probar la búsqueda se puede utilizar los mismos ejemplos que con la interfaz gráfica:

- Para buscar por título “Genetic Diversity”.
- Para buscar por resumen “Commercial exploitation”.

```
Índice creado
```

```
-----  
-----
```

```
BUSCADOR CON LUCENE
```

```
-----  
-----
```

```
Selecciones para buscar
```

- ```
1- Buscar por title
2- Buscar por abstract
```

```
1
```

```
Introduzca la búsqueda:
```

```
Genetic Diversity
```

```
Indique el número para mostrar el contenido
```

```
0 Genetic Diversity of Endangered Populations of Mysticete Whales:
```

```
Mitochondrial DNA and Historical Demography
```

```
1 Patterns as a Measure of Diversity in Small Populations
```

```
2 Genetic Variation and Estimates of Population Viability for a Rare
Perennial Plant
```

```
3 Plant Longevity and Life-form Diversity in Reconstructed Tropical
Ecosystems
```

```
4 Effects of Landscape Management and Deer Abundance on Plant Community
and Species Diversity
```

```
1
```

```
Studies of chickens have provided serological and nucleic acid probes
useful in defining the major histocompatibility complex (MHC) in other
avian species. Methods used in detecting genetic diversity at loci
within the MHC of chickens and mammals will be applied to determining
the extent of MHC polymorphism within small populations of ring-necked
pheasants, wild turkeys, cranes, Andean condors and other species. The
knowledge and expertise gained from working with the MHC of the chicken
should make for rapid progress in defining the polymorphism of the MHC
in these species and in detecting the polymorphism of MHC gene pool
within small wild and captive populations of these birds. Genes within
the major histocompatibility complex (MHC) are known to encode molecules
```

that provide the context **for** recognition of foreign antigens by the immune system. Whether a given animal is able to mount an immune response to the challenge of a pathogen is determined, **in** part, by the allelic makeup of its MHC. In many species, an unusually high degree of polymorphism is maintained at multiple loci within the MHC **in** freely breeding populations. The allelic pool within a population presumably provides diversity upon **which** to draw **in** the face of environmental challenge. The objective of the proposed research is to extend ongoing studies of the MHC of domesticated fowl to include avian species experiencing severe reduction **in** population size. Knowledge of the MHC gene pool within populations and of the haplotypes of individual animals may be useful **in** the husbandry of species requiring intervention **for** their preservation.

### 3. Uso de los jars

Para ejecutar los jars hay que situarse en el directorio **InformationRetrieval**. No es necesario tener instalado Maven, solo si se quiere volver a compilar o ejecutar el código fuente directamente.

Después ejecute:

1. Para la interfaz gráfica

```
java -jar GraphicInformationRetrieval.jar
```

2. Para la interfaz por consola

```
java - jar ConsoleInformationRetrieval.jar
```

El índice se creará en el mismo directorio.

# Referencias

- “A Simple JScrollPane for a JList Component : Scrollpane « Swing JFC « Java.” n.d. Accessed March 10, 2020.  
<http://www.java2s.com/Code/Java/Swing-JFC/AsimpleJScrollPaneforaJListcomponent.htm>.
- “Java Swing | Simple User Registration Form - GeeksforGeeks.” 2019. GeeksforGeeks. August 28, 2019.  
<https://www.geeksforgeeks.org/java-swing-simple-user-registration-form/>.
- “Lucene Tutorial - Index and Search Examples - HowToDoInJava.” n.d. HowToDoInJava. Accessed March 10, 2020.  
<https://howtodoinjava.com/lucene/lucene-index-search-examples/>.
- “What’s the Difference between Lucene StandardAnalyzer and EnglishAnalyzer?” n.d. Stack Overflow. Accessed March 10, 2020.  
<https://stackoverflow.com/questions/17011854/whats-the-difference-between-lucene-standardanalyzer-and-englishanalyzer>.