Recuperación de Información (2016-2017)
Grado en Ingeniería Informática
Universidad de Granada

Memoria Práctica Lucene

Pablo Martínez Ruano

28 de diciembre de 2016

Índice

Índice de figuras

Índice de tablas

1. Introducción

A continuación se encuentra la documentación asociada a la práctica desarrollada en la asignatura RI, en la cual vamos a llevar a cabo el desarrollo de un sistema de recuperación de información.Nuestro proyecto consiste en desarrollar una aplicación de recuperación de información usando Lucene. A lo largo de este proceso obtendremos las siguientes competencias:

1. Adquirir las destrezas, conocimientos y técnicas básicas para buscar información textual.
2. Entender el concepto de modelo de recuperación de información
3. Adquirir una visión general del proceso de recuperación de información.
4. Conocer los diferente componentes de un sistema de recuperación de información, su funcionamiento y relaciones entre ellos.
5. Comprender las peculiaridades de la recuperación de información XML y las similitudes y diferencias con la recuperación de información de la informática clásica.
6. Identificar los elementos que conforman la Web, así como conocer la estructura.
7. Conocer las técnicas específicas para la recuperación de información en la Web.
8. Asumir la importancia de la recuperación de información en e diseño y desarrollo de sistemas de información.
9. Analizar problemas de acceso de información en el marco de los sistemas de información y diseñar un sistema de recuperación de información que les de solución.
10. Ser capaz de integrar un sistema de recuperación de información en un sistema de información.
11. Conocer las instituciones responsables de la legislación vigente en el ámbito de los sistemas de información y ser conscientes de la normativa aplicable en cada momento. Así como poder evaluar la adecuación de un sistema de información a la normativa y legislación vigente.

2. Nuestro Proyecto

Nuestro proyecto va a ser una aplicación de recuperación de información para twitter. La idea es sencilla, nuestra aplicación estara formada por 2 ventanas. En una ventana estara compuesta por por un buscador. En la misma ventana y para usar el buscador tendremos a disposición un menu de opciones de aquello que deseamos filtrar en la busqueda. También como extra disponemos de un boton para imprimir el contenido en un fichero de texto.

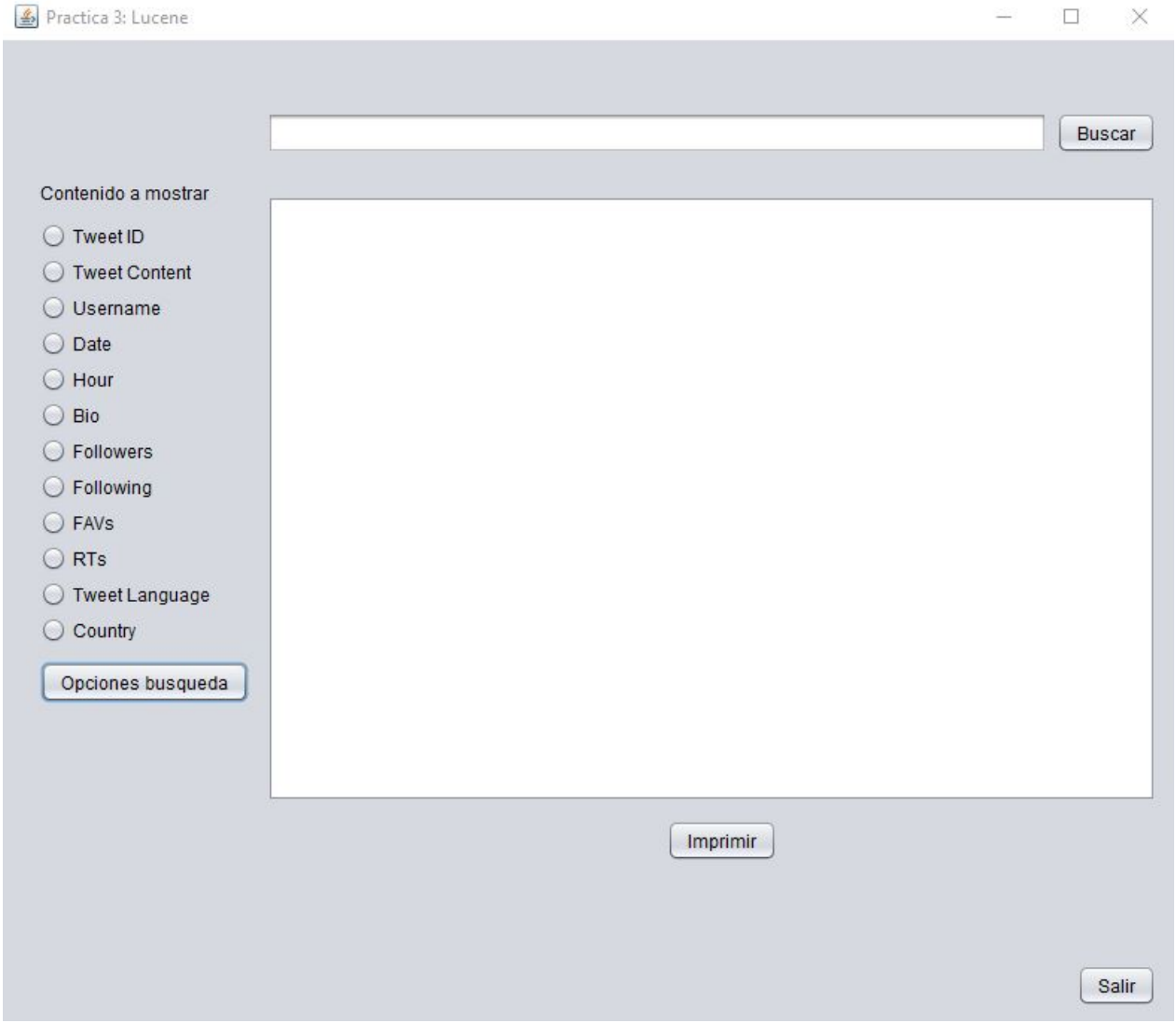

Figura 2.1: Muestra de la interfaz gráfica principal.

También disponemos de una segunda ventana, esta será emergente nos mostrara un menú de aquello que desemos buscar.

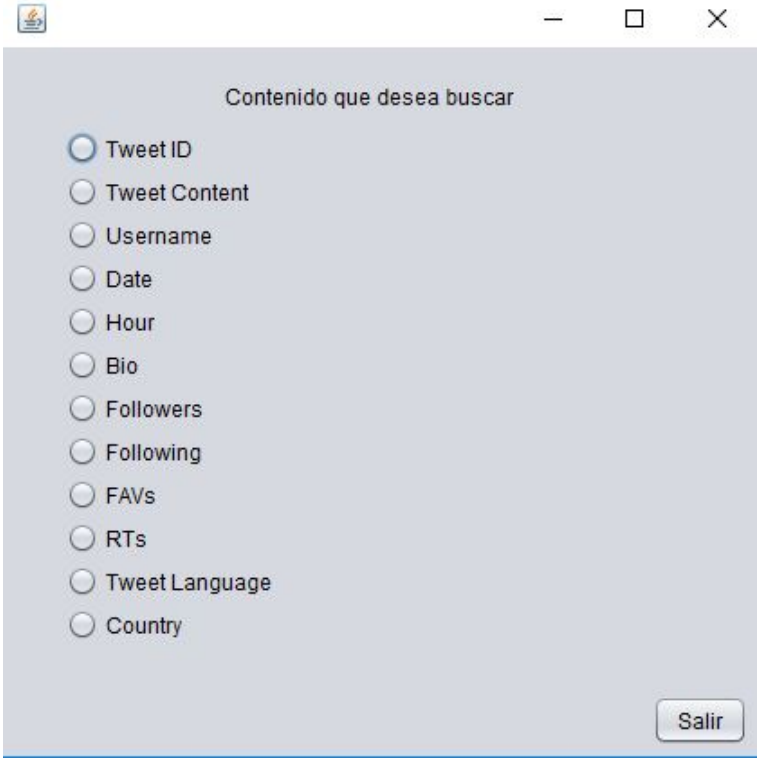

Figura 2.2: Ventana emergente de nuestra interfaz gráfica

El modo de uso es bastante sencillo, vamos explicarlo con un ejemplo. Imaginemos que insertamos en el buscador "2000z en opciones de busqueda marcamos Tweet Content. Es hara buscar tweet que contenga el numero 2000. Esto es solamente para buscar, para mostrar la información en opciones vamos a suponer que marcamos Nickname, Tweet Content, FAVs y RTs. Esto hara que buscara los contenido del tweet que tenga el numero 2000, pero mostrara el nickname del autor del tweet, el contenido del tweet, sus favs y rts.

3. Análisis previo de requisitos

A continuación vamos a comentar el análisis previo de requisitos que hemos realizado para el proyecto, para ello vamos hablar de los elementos funciones y de los elementos no funcionales.

3.1. Funcionales

RF1.**BuscarTUsername**El usuario usa el sistema de recuperación de información para buscar el Username

RF2**BuscarNickname.**El usuario usa el sistema de recuperación de información para buscar el Nickname

RF3.**BuscarTweetContent.**El usuario usa el sistema de recuperación de información para buscar el contenido del tweet

RF4.**BuscarDate.**El usuario usa el sistema de recuperación de información para buscar la fecha del tweet

RF5.**BuscarBio.**El usuario usa el sistema de recuperación de información para buscar las biograficas de los usuarios.

RF6.**BuscarCountry.**El usuario usa el sistema de recuperación de información para buscar el pais del usuario.

RF7.**BuscarPlace.**El usuario usa el sistema de recuperación de información para buscar el lugar donde se escribio el tweet

RF8.**BuscarLanguage.**El usuario usa el sistema de recuperación de información para buscar el lenguaje

RF9.**BuscarFAVs.**El usuario usa el sistema de recuperación de información para buscar los Favs

RF10.**BuscarRTs.**El usuario usa el sistema de recuperación de información para buscar los RTs

RF11.**FiltrarTUsername**El usuario usa el sistema de recuperación de información para filtrar el Username

RF12**FiltrarNickname.**El usuario usa el sistema de recuperación de información para filtrar el Nickname

RF13.**FiltrarTweetContent.**El usuario usa el sistema de recuperación de información para filtrar el contenido del tweet

RF14.**FiltrarDate.**El usuario usa el sistema de recuperación de información para filtrar la fecha

RF15.**FiltrarBio.**El usuario usa el sistema de recuperación de información para filtrar la biografia del usuario.

RF16.**FiltrarCountry.**El usuario usa el sistema de recuperación de información para filtrar el pais

RF17.**FiltrarPlace.**El usuario usa el sistema de recuperación de información para filtrar el lugar

RF18.**FiltrarLanguage.**El usuario usa el sistema de recuperación de información para filtrar el idioma

RF19.**FiltrarFAVs.**El usuario usa el sistema de recuperación de información para filtrar los FAVs

RF20.**FiltrarRTs.**El usuario usa el sistema de recuperación de información para filtrar los RTs

3.2. No funcionales

RNF1. El sistema no deberá mostrar nada si no se marcado en los menus ninguna opción. **RNF2.**Las búsquedas simultaneas solo se podran realizar las que se hayan descrito en el manual. **RNF3.**El cuadro de texto se debera mover tanto largo como ancho para mostrar el contenido completo. **RNF4.**Se debe incluir una funcion para imprimir los resultados y de esa manera guardarlos. **RNF5.**Los ficheros que indexaremos seran formato .csv.

4. Diseño de la solución

Ahora vamos comentar el diseño que vamos enfocar para dar solución a nuestro problema. Para ello vamos comentar la arquitectura de nuestro problema, los posibles escenarios, la vista de procesos y la vista de desarrollo.

4.1. Arquitectura de nuestro programa

Para llegar a cabo este proyecto no necesitamos un hardware costoso, nos vale con nuestro ordenador personal. El sistema operativo a utilizar también es indiferente, nos sirve tanto Windows, MAC OS o algún Linux ya que vamos utilizar la máquina virtual de Java para ejecutar la aplicación, el unico requisito es que nuestro SO soporte Java. Lo que si necesitamos es tener las biblitecas de Lucene y Tika , que son las bibliotecas que vamos utilizar para llevar acabo el proyecto.

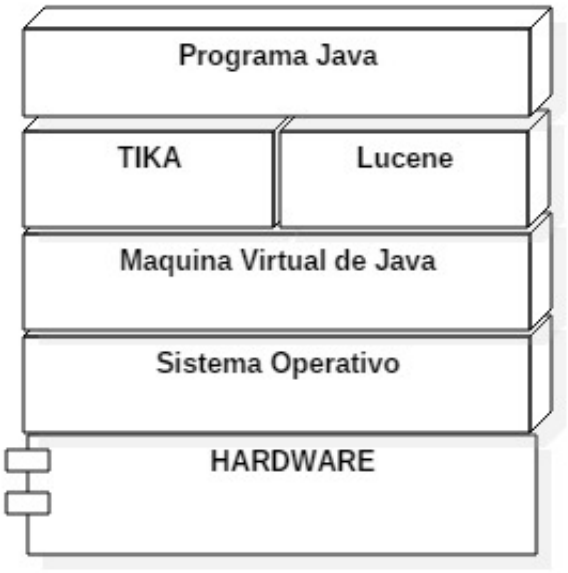

Figura 4.1: Muestra de la interfaz gráfica principal.

4.2. Escenario

A continuación hemos preparado un esquema sobre el escenario posible de interacción de los usuarios con el software.

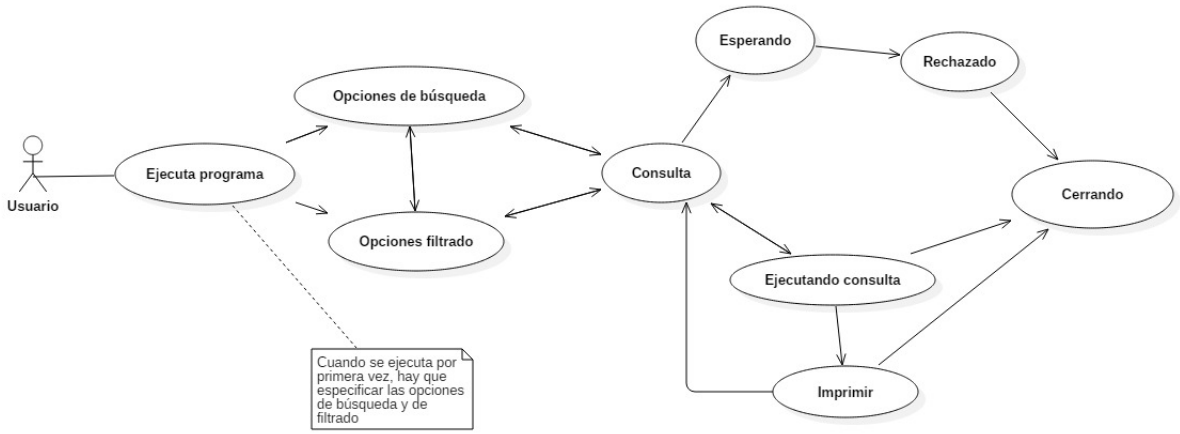

Figura 4.2: Muestra de la interfaz gráfica principal.

Como se puede ver no es muy complejo, una vez se haya iniciado el programa el usuario

por primera vez tendra que marcar lo que desea buscar y filtrar, una vez realizado esto el usuario interacciona con el buscador. Pueden darse 2 casos: Que la buscado no de resultado lo cual provoca el cierre el programa en caso de error, o que sastifaga su busqueda. Si satisface su busqueda se le da la opcion de imprimir para guardar su búsqueda y después el usuario puede realizar otra busqueda o usar el programa.

4.3. Vista de los procesos

Ahora vamos a mostrar una vista de los procesos, en cual de forma sencilla vamos a mostrar de manera gráfica todo el abanico de opciones que proporciona nuestro software, como ya antes habiamos especificados en nuestros requisitos funcionales

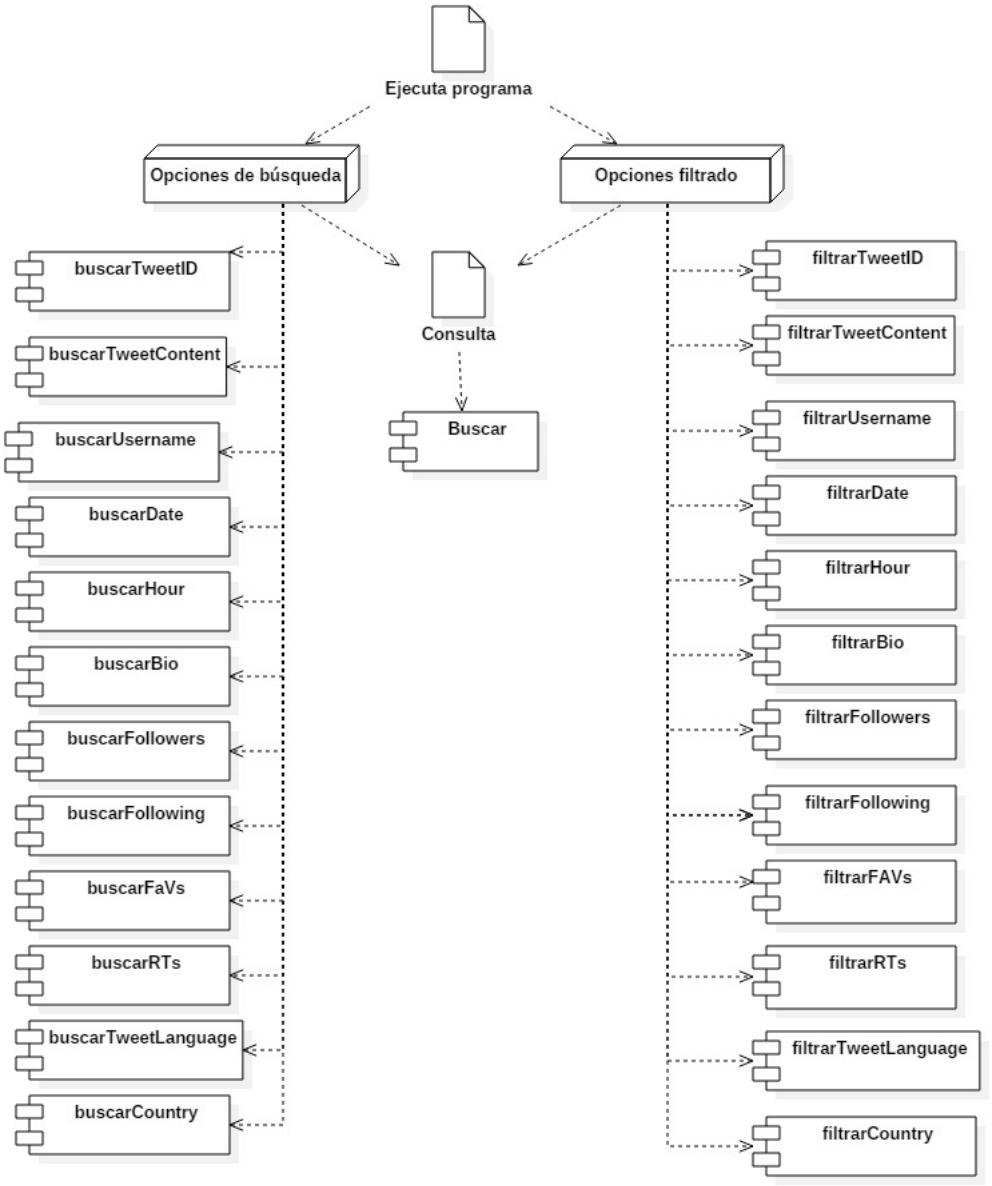

Figura 4.3: Muestra de la interfaz gráfica principal.

4.4. Vista del desarrollo

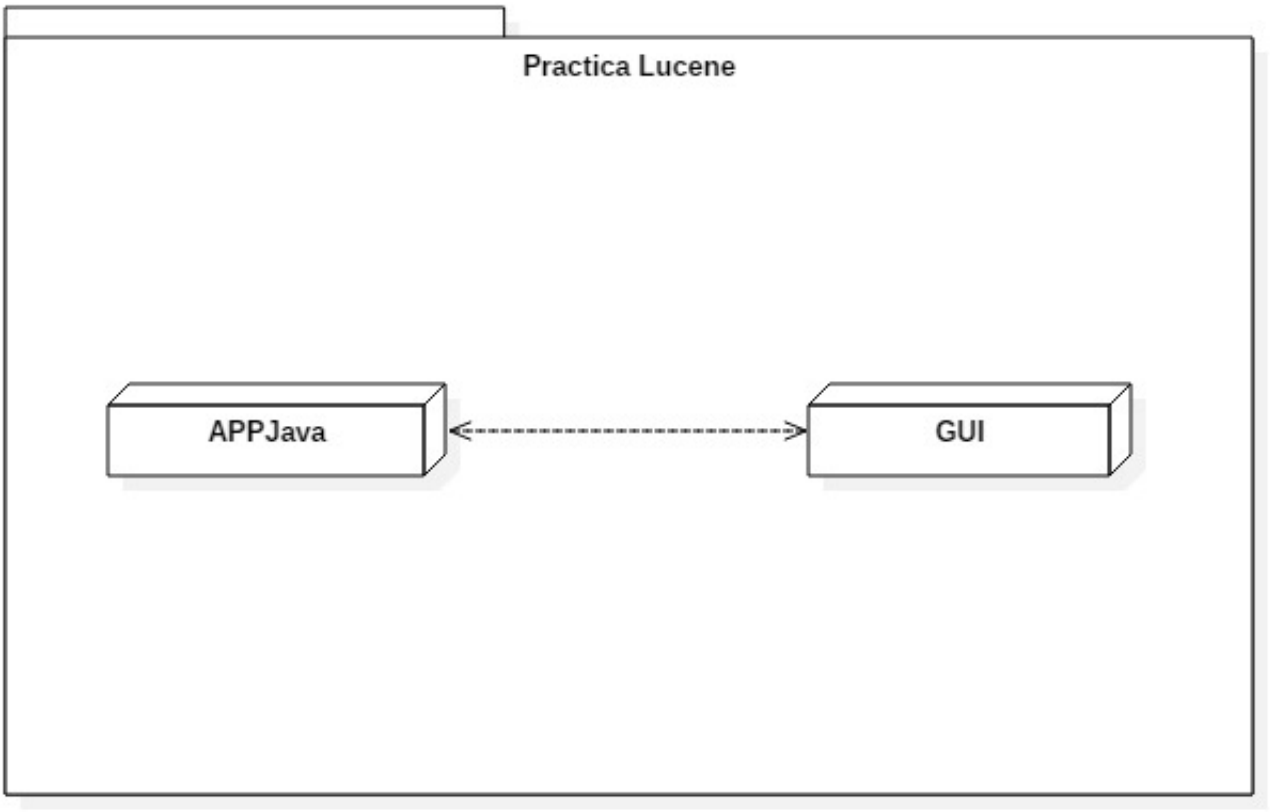

Figura 4.4: Muestra de la interfaz gráfica principal.

5. Codigo

5.1. Indexar

```
/*
 * To change this license header, choose License Headers in
    Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package indexacion;

import java.io.BufferedReader;
```

```java
import java.io.FileReader;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.lucene.document.TextField;
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Field;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.RAMDirectory;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.IntPoint;
import org.apache.lucene.document.NumericDocValuesField;
import org.apache.lucene.document.StoredField;
import org.apache.lucene.document.StringField;
import org.apache.lucene.store.FSDirectory;


/**
 *
 * @author salvador
 */
public class Indexacion {

    /**
     * @param args the command line arguments
     */


    public static void main(String[] args) {
        /*Date UserName Nickname Bio TweetContent Favs RTs
            Country
        Place (as appears on Bio) Followers Following Language*/
        final String DATE = "Date";
        final String USERNAME = "UserName";
        final String NICKNAME = "Nickname";
        final String BIO = "Bio";
        final String TWEET = "TweetContent";
        final String FAVS = "Favs";
        final String RTS = "RTs";
        final String COUNTRY = "Country";
```

```
final String PLACE = "Place";
final String FOLLOWERS = "Followers";
final String FOLLOWING = "Following";
final String LANGUAGE = "Language";




try {
    Analyzer analyzer = new StandardAnalyzer();
    // Store the index in memory:
    //Directory directory = new RAMDirectory();
    // en disco ...
    Directory directory = FSDirectory.open(Paths.get("/
       home/salvador/Escritorio/UGR/RI/practica-final/
       index"));
    IndexWriterConfig config = new IndexWriterConfig(
       analyzer);
    IndexWriter iwriter = new IndexWriter(directory,
       config);

    BufferedReader bf = null;
    try {
        bf = new BufferedReader(new FileReader(args[0]))
           ;
        String bfRead = bf.readLine();
        while ((bfRead = bf.readLine()) != null) {
            String[] text = bfRead.split("\t");
            ArrayList<String> listaTokens = new
               ArrayList<>();
            for (String s : text){
                if (s.isEmpty())
                    listaTokens.add("null");
                else
                    listaTokens.add(s);
            }
            Document doc = new Document();
            doc.add(new TextField(DATE, listaTokens.get
               (0), Field.Store.YES));
            doc.add(new TextField(USERNAME, listaTokens.
               get(1), Field.Store.YES));
            doc.add(new TextField(NICKNAME, listaTokens.
               get(2), Field.Store.YES));
```

```
                    doc.add(new TextField(BIO, listaTokens.get
                        (3), Field.Store.YES));
                    doc.add(new StringField(TWEET, listaTokens.
                        get(4), Field.Store.YES));
                    if (!listaTokens.get(5).equals("null"))
                        doc.add(new IntPoint(FAVS, Integer.
                            parseInt(listaTokens.get(5))));
                    if (!listaTokens.get(6).equals("null"))
                        doc.add(new IntPoint(RTS, Integer.
                            parseInt(listaTokens.get(6))));
                    doc.add(new TextField(COUNTRY, listaTokens.
                        get(7), Field.Store.YES));
                    doc.add(new TextField(PLACE, listaTokens.get
                        (8), Field.Store.YES));
                    /*if (!listaTokens.get(9).equals("null"))
                        doc.add(new IntPoint(FOLLOWERS, Integer.
                            parseInt(listaTokens.get(9))));
                    if (!listaTokens.get(10).equals("null"))
                        doc.add(new IntPoint(FOLLOWING, Integer.
                            parseInt(listaTokens.get(10))));*/
                    doc.add(new TextField(LANGUAGE, listaTokens.
                        get(11), Field.Store.YES));



                    iwriter.addDocument(doc);
                }
            } catch (Exception ex) {
                Logger.getLogger(Indexacion.class.getName()).log
                    (Level.SEVERE, null, ex);
            }
            iwriter.close();
        } catch (Exception ex) {
            Logger.getLogger(Indexacion.class.getName()).log(
                Level.SEVERE, null, ex);
        }

    }

}
```

5.2. Buscar

```
/*
```

```
 * To change this license header, choose License Headers in
    Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package indexacion;

import java.io.File;
import java.io.IOException;
import java.nio.file.Paths;
import java.text.DateFormat;
import java.util.StringTokenizer;
import javax.swing.event.DocumentListener;
import javax.swing.event.UndoableEditListener;
import javax.swing.text.AttributeSet;
import javax.swing.text.BadLocationException;

import javax.swing.text.Element;
import javax.swing.text.Position;
import javax.swing.text.Segment;
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.standard.StandardAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;
import static org.apache.lucene.document.IntPoint.newRangeQuery;
import org.apache.lucene.document.TextField;
import org.apache.lucene.index.DirectoryReader;
import org.apache.lucene.index.IndexReader;

import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.index.IndexWriterConfig;
import org.apache.lucene.index.Term;
import org.apache.lucene.queryparser.classic.ParseException;
import org.apache.lucene.queryparser.classic.QueryParser;
import org.apache.lucene.search.BooleanClause;
import org.apache.lucene.search.BooleanQuery;
import org.apache.lucene.search.IndexSearcher;
import org.apache.lucene.search.MatchAllDocsQuery;
import org.apache.lucene.search.PhraseQuery;
import org.apache.lucene.search.Query;
import org.apache.lucene.search.ScoreDoc;
import org.apache.lucene.search.Sort;
import org.apache.lucene.search.SortField;
import org.apache.lucene.search.TermQuery;
```

```
import org.apache.lucene.search.TermRangeQuery;
import org.apache.lucene.search.TopDocs;
import org.apache.lucene.search.TopFieldDocs;
import org.apache.lucene.store.Directory;
import org.apache.lucene.store.FSDirectory;
import org.apache.lucene.store.RAMDirectory;
/**
 *
 * @author felix
 */
public class Busqueda {
        //1-busca usuarios
        //2-busca usuarios y los tweets de esos usuarios
        //3-busca usuairos y su nickname
        //4-buscar en tweets y mostrar fecha
        //5-busca en biografia y muestra el usuario
        //6-busca usuarios y muestra su nacionalidad
        //7-busca en tweets y muesta el lugar donde fue escrito
        //8-busca en tweets y muestra el lenguaje
        //9-busca los tweets retweeteados entre dos valores
        //10-busca los favoritos retweeteados entre dos valores



        final String USERNAME = "UserName";
        final String NICKNAME = "Nickname";
        final String TWEET = "TweetContent";
        final String DATE = "Date";
        final String BIO = "Bio";
        final String COUNTRY = "Country";
        final String PLACE = "Place";
        final String LANGUAGE = "Language";
        final String FAVS = "Favs";
        final String RTS = "RTs";

        /*final String FOLLOWERS = "Followers";
        final String FOLLOWING = "Following";*/


    //Para cuando tengo la opcion de nombre de usuarios solo

    public void buscarUsuarios(String ruta,String busqueda)
       throws IOException, ParseException, Exception{
       Directory d=FSDirectory.open(Paths.get(ruta));
```

```
        IndexReader ireader=DirectoryReader.open(d);
        IndexSearcher isearcher =new IndexSearcher(ireader);
        BooleanQuery.Builder Constructor = new BooleanQuery.
            Builder();
        StringTokenizer st =new StringTokenizer(busqueda);
        while(st.hasMoreTokens()){
            TermQuery query = new TermQuery(new Term(USERNAME, st
                .nextElement().toString()));
            Constructor.add(query, BooleanClause.Occur.MUST);
        }
        BooleanQuery q = Constructor.build();
        TopDocs top=isearcher.search(q,1000);
        System.out.println("
            ------------------------------------------------------
            ");
        for(int i=0;i<top.scoreDocs.length;i++){
            int docId =top.scoreDocs[i].doc;
            Document foundDocument = isearcher.doc(docId);
            System.out.println(foundDocument.get(USERNAME));
        }
        System.out.println("Encontrado " + top.scoreDocs.length
            + "resultados");
        System.out.println("
            ------------------------------------------------------
            ");
        d.close();
    }


    //Para cuando tengo la opcion de nombre de usuarios con la
        de tweets, busca por nombre de usuario y muestra sus
        tweets

     public void buscarUsuariosYTweets(String ruta,String
        busqueda) throws IOException, ParseException, Exception
        {
        Directory d=FSDirectory.open(Paths.get(ruta));
        IndexReader ireader=DirectoryReader.open(d);
        IndexSearcher isearcher =new IndexSearcher(ireader);
        BooleanQuery.Builder Constructor = new BooleanQuery.
            Builder();
        StringTokenizer st =new StringTokenizer(busqueda);
        while(st.hasMoreTokens()){
```

```
            TermQuery query = new TermQuery(new Term(USERNAME, st
                .nextElement().toString()));
            Constructor.add(query, BooleanClause.Occur.MUST);
        }
        BooleanQuery q = Constructor.build();
        TopDocs top=isearcher.search(q,1000);
        System.out.println("
            ------------------------------------------------------
            ");
        for(int i=0;i<top.scoreDocs.length;i++){
            int docId =top.scoreDocs[i].doc;
            Document foundDocument = isearcher.doc(docId);
            System.out.println(foundDocument.get(USERNAME) + "  
                " + foundDocument.get(TWEET));
        }
        System.out.println("Encontrado " + top.scoreDocs.length
            + "resultados");
        System.out.println("
            ------------------------------------------------------
            ");
        d.close();
    }

    //busca por nombre de usuario y muestra su nickname

    public void buscarUsuariosYNickname(String ruta,String
        busqueda) throws IOException, ParseException{
        Directory d=FSDirectory.open(Paths.get(ruta));
        IndexReader ireader=DirectoryReader.open(d);
        IndexSearcher isearcher =new IndexSearcher(ireader);
        BooleanQuery.Builder Constructor = new BooleanQuery.
            Builder();
        StringTokenizer st =new StringTokenizer(busqueda);
        while(st.hasMoreTokens()){
            TermQuery query = new TermQuery(new Term(USERNAME, st
                .nextElement().toString()));
            Constructor.add(query, BooleanClause.Occur.MUST);
        }
        BooleanQuery q = Constructor.build();
        TopDocs top=isearcher.search(q,1000);
        System.out.println("
            ------------------------------------------------------
            ");
        for(int i=0;i<top.scoreDocs.length;i++){
```

```
            int docId =top.scoreDocs[i].doc;
            Document foundDocument = isearcher.doc(docId);
            System.out.println(foundDocument.get(USERNAME) + "␣
                @" + foundDocument.get(NICKNAME));
        }
        System.out.println("Encontrado␣" + top.scoreDocs.length
            + "resultados");
        System.out.println("
            ----------------------------------------------------
            ");
        d.close();
    }

    //busca en el contenido de los tweets y muestra el tweet y
        su fecha de publicacion, para cuando tenga
    //la opcion de buscar el tweet y la de fecha

    public void buscarEnTweetsYFecha(String ruta ,Analyzer a,
        String busqueda) throws IOException, ParseException{
        Directory d=FSDirectory.open(Paths.get(ruta));
        IndexReader ireader=DirectoryReader.open(d);
        IndexSearcher isearcher =new IndexSearcher(ireader);
        BooleanQuery.Builder Constructor = new BooleanQuery.
            Builder();
        StringTokenizer st =new StringTokenizer(busqueda);
        while(st.hasMoreTokens()){
            TermQuery query = new TermQuery(new Term(TWEET, st.
                nextElement().toString()));
            Constructor.add(query, BooleanClause.Occur.MUST);
        }
        BooleanQuery q = Constructor.build();
        TopDocs top=isearcher.search(q,1000);
        System.out.println("
            ----------------------------------------------------
            ");
        for(int i=0;i<top.scoreDocs.length;i++){
            int docId =top.scoreDocs[i].doc;
            Document foundDocument = isearcher.doc(docId);
            System.out.println(foundDocument.get(TWEET) +
                foundDocument.get(DATE));
        }
        System.out.println("Encontrado␣" + top.scoreDocs.length
            + "resultados");
```

```
        System.out.println("
            ------------------------------------------------------------
            ");
        d.close();
    }

    //busca en el contenido de la biografia y muestra la
        biografia completa y el nombre del usuario que le
        pertenece

     public void buscarBioYUsuario(String ruta,Analyzer a,String
          busqueda) throws IOException, ParseException{
        Directory d=FSDirectory.open(Paths.get(ruta));
        IndexReader ireader=DirectoryReader.open(d);
        IndexSearcher isearcher =new IndexSearcher(ireader);
        QueryParser parser = new QueryParser(BIO,a);
        Query query = parser.parse(busqueda);
        ScoreDoc[] hits =isearcher.search(query,1000).scoreDocs;
        for(int i=0;i<hits.length;i++){
            Document hitDoc=isearcher.doc(hits[i].doc);
            System.out.println(hitDoc.get(BIO).toString() + "  "
                + hitDoc.get(USERNAME).toString());
        }
        ireader.close();
        d.close();
    }

    //busca por el nombre de usuario y muestra el nombre del
        usuario y su nacionalidad

    public void buscarUsuarioYNacionalidad(String ruta,Analyzer
        a,String busqueda) throws IOException, ParseException{
        Directory d=FSDirectory.open(Paths.get(ruta));
        IndexReader ireader=DirectoryReader.open(d);
        IndexSearcher isearcher =new IndexSearcher(ireader);
        BooleanQuery.Builder Constructor = new BooleanQuery.
            Builder();
        StringTokenizer st =new StringTokenizer(busqueda);
        while(st.hasMoreTokens()){
            TermQuery query = new TermQuery(new Term(USERNAME, st
                .nextElement().toString()));
            Constructor.add(query, BooleanClause.Occur.MUST);
        }
        BooleanQuery q = Constructor.build();
```

```
        TopDocs top=isearcher.search(q,1000);
        System.out.println("
            ----------------------------------------------------------
            ");
        for(int i=0;i<top.scoreDocs.length;i++){
            int docId =top.scoreDocs[i].doc;
            Document foundDocument = isearcher.doc(docId);
            System.out.println(foundDocument.get(USERNAME) + "  "
                " + foundDocument.get(COUNTRY));
        }
        System.out.println("Encontrado " + top.scoreDocs.length
            + "resultados");
        System.out.println("
            ----------------------------------------------------------
            ");
        d.close();
    }

    //busca en el contenido de los tweets y miestra el tweet
        completo y el lugar donde se publico

    public void buscarEnTweetsYLugar(String ruta ,Analyzer a,
        String busqueda) throws IOException, ParseException{
        Directory d=FSDirectory.open(Paths.get(ruta));
        IndexReader ireader=DirectoryReader.open(d);
        IndexSearcher isearcher =new IndexSearcher(ireader);
        BooleanQuery.Builder Constructor = new BooleanQuery.
            Builder();
        StringTokenizer st =new StringTokenizer(busqueda);
        while(st.hasMoreTokens()){
            TermQuery query = new TermQuery(new Term(TWEET, st.
                nextElement().toString()));
            Constructor.add(query, BooleanClause.Occur.MUST);
        }
        BooleanQuery q = Constructor.build();
        TopDocs top=isearcher.search(q,1000);
        System.out.println("
            ----------------------------------------------------------
            ");
        for(int i=0;i<top.scoreDocs.length;i++){
            int docId =top.scoreDocs[i].doc;
            Document foundDocument = isearcher.doc(docId);
            System.out.println(foundDocument.get(TWEET) +
                foundDocument.get(PLACE));
```

```
        }
        System.out.println("Encontrado " + top.scoreDocs.length
            + "resultados");
        System.out.println("
            ----------------------------------------------------------

            ");
        d.close();
    }

    //busca en el contenido de los tweets y los muestra
        completos y el lenguaje en el que estan escritos

    public void buscarEnTweetsYLanguage(String ruta ,Analyzer a,
        String busqueda) throws IOException, ParseException{
        Directory d=FSDirectory.open(Paths.get(ruta));
        IndexReader ireader=DirectoryReader.open(d);
        IndexSearcher isearcher =new IndexSearcher(ireader);
        BooleanQuery.Builder Constructor = new BooleanQuery.
            Builder();
        StringTokenizer st =new StringTokenizer(busqueda);
        while(st.hasMoreTokens()){
            TermQuery query = new TermQuery(new Term(TWEET, st.
                nextElement().toString()));
            Constructor.add(query, BooleanClause.Occur.MUST);
        }
        BooleanQuery q = Constructor.build();
        TopDocs top=isearcher.search(q,1000);
        System.out.println("
            ----------------------------------------------------------

            ");
        for(int i=0;i<top.scoreDocs.length;i++){
            int docId =top.scoreDocs[i].doc;
            Document foundDocument = isearcher.doc(docId);
            System.out.println(foundDocument.get(TWEET) +
                foundDocument.get(LANGUAGE));
        }
        System.out.println("Encontrado " + top.scoreDocs.length
            + "resultados");
        System.out.println("
            ----------------------------------------------------------

            ");
        d.close();
    }
```

```
//Se muestran los tweets retweeteados entre los valores a y
    b, EL PROBLEMA ES QUE NO SE PUEDE SABER CUANTOS TIENE
    CADA UNO
//BORRA LO QUE ESTA EN MAYUSCULAS

public void buscarMasReTweets(String ruta,int a , int b)
    throws IOException, ParseException{
     Directory d=FSDirectory.open(Paths.get(ruta));
     IndexReader ireader=DirectoryReader.open(d);
     IndexSearcher isearcher =new IndexSearcher(ireader);
     Query q = newRangeQuery(RTS,a,b);
     ScoreDoc[] hits =isearcher.search(q,1000).scoreDocs;
     for(int i=0;i<hits.length;i++){
          Document hitDoc=isearcher.doc(hits[i].doc);
          System.out.println(hitDoc.get(TWEET).toString());
     }
     ireader.close();
     d.close();
}

//Se muestran los tweets favoritos entre los valores a y b,
    EL PROBLEMA ES QUE NO SE PUEDE SABER CUANTOS TIENE CADA
    UNO
//BORRA LO QUE ESTA EN MAYUSCULAS

public void buscarMasFavs(String ruta,int a , int b) throws
    IOException, ParseException{
     Directory d=FSDirectory.open(Paths.get(ruta));
     IndexReader ireader=DirectoryReader.open(d);
     IndexSearcher isearcher =new IndexSearcher(ireader);
     Query q = newRangeQuery(FAVS,a,b);
     ScoreDoc[] hits =isearcher.search(q,1000).scoreDocs;
     for(int i=0;i<hits.length;i++){
          Document hitDoc=isearcher.doc(hits[i].doc);
          System.out.println(hitDoc.get(TWEET).toString());
     }
     ireader.close();
     d.close();
}

public static void main(String[] args) throws Exception {
     String indexDir = args[0];

     Busqueda b = new Busqueda();
```

```
        Analyzer analyzer=new StandardAnalyzer();

        //LOS SIGUIENTES SON BUENOS EJEMPLOS PARA LA
            PRESENTACION

        //b.buscarBioYUsuario(indexDir, analyzer, "juan");
        //b.buscarEnTweetsYFecha(indexDir, analyzer, "monumento
            barcelona");
        //b.buscarEnTweetsYLanguage(indexDir, analyzer, "
            monumento estrellarse");
        //b.buscarEnTweetsYLugar(indexDir, analyzer, "monumento
            pinal");

        //Para ver que funciona probalos desde los que tienen 1
            a 100 y desde 99 a 100
        //b.buscarMasFavs(indexDir, 1, 100);
        //b.buscarMasFavs(indexDir, 99, 100);

        //Para ver que funciona probalos desde los que tienen 1
            a 100 y desde 90 a 100
        //b.buscarMasReTweets(indexDir, 1, 100);
        //b.buscarMasReTweets(indexDir, 90, 100);

        //b.buscarUsuarioYNacionalidad(indexDir, analyzer, "juan
             jose");
        //b.buscarUsuarios(indexDir, "juan jose");
        //b.buscarUsuariosYNickname(indexDir, "juan jose");
        b.buscarUsuariosYTweets(indexDir, "juan_pedro");



    }
}
```

6. Manual de Usuario

7. Organización de trabajo

En esta práctica la organización de trabajo ha sido Salvador 40 %, Felix 30 % y Pablo 30 %