ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

EPSC

Grado en Ingeniería Informática
Mención en Ingeniería del Software

TRABAJO FIN DE GRADO

Diseño y desarrollo de una herramienta para analizar tweets de Twitter

MANUAL DE CÓDIGO

Autor:
Antonio Luis Ríos Castillejo
Directores:
Dña. Amelia Zafra Gómez
D. Luis Toribio Castro

Córdoba, junio de 2018

Índice general

Índice de tablas

Índice de Código

Capítulo 1

Introducción

Este es el Manual de Código del Trabajo Fin de Grado ***Diseño y desarrollo de una herramienta para analizar tweets de Twitter***, donde se incluye una descripción detallada de todos los elementos que componen el sistema desarrollado en el mismo. El objetivo de este documento es servir de manual de código para futuras modificaciones o mejoras que puedan desarrollarse sobre la aplicación *software*.

En este manual encontraremos el código de la aplicación ordenado por clases.

Por último es importante recordar que este manual es tan solo una referencia del código de la aplicación y que por tanto, para comprender completamente el funcionamiento de la misma, es imprescindible la consulta del Manual Técnico de dicho Trabajo Fin de Grado.

El objetivo de este Trabajo ha sido el diseñar y desarrollar una herramienta para poder analizar cuentas de Twitter. Principalmente, las cuentas de los principales cuerpos de seguridad que existen en España, los cuales son: *Policía Nacional, Guardia Civil, Policía Municipal de Madrid y Mossos d'Esquadra.*

Además, la aplicación también permite analizar cualquier otra cuenta que el usuario desee.

Se ha utilizado el lenguaje de programación Java para este propósito ya que es en un potente lenguaje orientado a objetos, y además se trata de un lenguaje independiente de la plataforma, es decir, cualquier programa creado a través de Java podrá funcionar correctamente en ordenadores de todo tipo y con sistemas operativos distintos.

En este manual explicaremos cada una de las clases y expondremos tanto la organización como la estructura de las mismos, poniendo énfasis en las características más relevantes del diseño que se ha seguido, así como su funcionalidad y cometido dentro del conjunto del Trabajo Fin de Grado.

Capítulo 2

Características de la implementación

2.1. Notación de la implementación

En relación a la estandarización en la programación, se procurará seguir el estándar estudiado durante la carrera a la hora de definir correctamente cada una de las clases de objetos y sus distintas propiedades.

En cuanto al código en sí, se seguirán las convenciones recomendadas de nombramiento tanto general como específico. Algunas de ellas son:

- Los nombres deben ser lo más autoexplicativos posibles.
- Los nombres de clases deben comenzar por mayúscula. En caso de unión de varias palabras, las palabras se unen con mayúscula las primera letra de cada palabra, siguiendo la convención *CapWords*.
- Los nombres de las variables y funciones deben estar en minúscula. Cuando el nombre es la unión de varias palabras, esta se realiza haciendo uso del guión bajo, para mejorar la legibilidad.
- Los nombres que representen constantes deben estar escritos en mayúscula, usando guión bajo para separar palabras.
- Las abreviaciones y acrónimos no deben ser escritos en mayúscula cuando sean utilizados como nombres, a excepción de para las clases, donde si se han escrito con mayúscula al utilizarse la convención mencionada anteriormente.

Junto con las convenciones en cuanto a la escritura de nombres, existen otras referentes a la estructuración de los paquetes, de los ficheros, de las clases, etc., que también serán tenidas en cuenta.

2.2. Estructura de la documentación

Para documentar detalladamente el código desarrollado en nuestro Trabajo Fin de Grado, iremos definiendo las distintas clases y funciones implementadas para la consecución de la funcionalidad prevista de la aplicación.

En cuanto a las clases, de cada una de ellas se realizará una descripción general de su cometido, para a continuación exponer el conjunto de métodos que componen la misma.

Capítulo 3

Documentación del código

En este capítulo se va a explicar en detalle la implementación de cada una de las clases o funciones que se han desarrollado para cumplir con los objetivos propuestos en nuestro Trabajo Fin de Grado.

3.1. Clase Informe

En este apartado se especifica de forma concisa la clase Informe, que contiene un JFrame para mostrar informes. Esta especificación se muestra en la Tabla 3.1.

Especificación de la clase Informe

Clase: *Informe*	
Esta clase contiene un JFrame para mostrar los diferentes informes.	
Métodos	
+ Informe	Constructor de la clase Informe.
+ main	Función que inicializa la ventana.
+ setText	Modificador de textoInforme.
+ getText	Observador de textoInforme.

Tabla 3.1: Especificación de los métodos de la clase Informe

3.1.1. Fichero Informe.java

```
1
2  package principal;
3
4  /**
5  * Clase Informe contiene un JFrame para mostrar diferentes
       informes en la aplicación.
6  *
7  * @author Antonio Luis Ríos
8  * @version 20180626
9  */
10
11
12 public class Informe extends javax.swing.JFrame {
13
14     /**
15      * Creates new form Informe
16      */
17     public Informe() {
18         initComponents();
19     }
20
21     /**
22      * This method is called from within the constructor to
       initialize the form.
23      * WARNING: Do NOT modify this code. The content of this
       method is always
24      * regenerated by the Form Editor.
25      */
26     @SuppressWarnings("unchecked")
27     // <editor-fold defaultstate="collapsed" desc="Generated
       Code">//GEN-BEGIN:initComponents
28     private void initComponents() {
```

```
29
30          jScrollPane1 = new javax.swing.JScrollPane();
31          textoInforme = new javax.swing.JTextArea();
32
33          setDefaultCloseOperation(javax.swing.WindowConstants.
    DISPOSE_ON_CLOSE);
34          setMinimumSize(new java.awt.Dimension(400, 400));
35
36          textoInforme.setEditable(false);
37          textoInforme.setColumns(20);
38          textoInforme.setRows(5);
39          jScrollPane1.setViewportView(textoInforme);
40
41          javax.swing.GroupLayout layout = new javax.swing.
    GroupLayout(getContentPane());
42          getContentPane().setLayout(layout);
43          layout.setHorizontalGroup(
44              layout.createParallelGroup(javax.swing.GroupLayout.
    Alignment.LEADING)
45              .addGroup(layout.createSequentialGroup()
46                  .addContainerGap()
47                  .addComponent(jScrollPane1, javax.swing.
    GroupLayout.DEFAULT_SIZE, 864, Short.MAX_VALUE)
48                  .addContainerGap())
49          );
50          layout.setVerticalGroup(
51              layout.createParallelGroup(javax.swing.GroupLayout.
    Alignment.LEADING)
52              .addGroup(layout.createSequentialGroup()
53                  .addContainerGap()
54                  .addComponent(jScrollPane1, javax.swing.
    GroupLayout.DEFAULT_SIZE, 537, Short.MAX_VALUE)
55                  .addContainerGap())
56          );
```

```
57
58          pack();
59      }// </editor-fold>//GEN-END:initComponents
60
61      /**
62       * @param args the command line arguments
63       */
64      public static void main(String args[]) {
65          /* Set the Nimbus look and feel */
66          //<editor-fold defaultstate="collapsed" desc=" Look and
     feel setting code (optional) ">
67          /* If Nimbus (introduced in Java SE 6) is not available
    , stay with the default look and feel.
68           * For details see http://download.oracle.com/javase/
    tutorial/uiswing/lookandfeel/plaf.html
69           */
70          try {
71              for (javax.swing.UIManager.LookAndFeelInfo info :
    javax.swing.UIManager.getInstalledLookAndFeels()) {
72                  if ("Nimbus".equals(info.getName())) {
73                      javax.swing.UIManager.setLookAndFeel(info.
    getClassName());
74                      break;
75                  }
76              }
77          } catch (ClassNotFoundException ex) {
78              java.util.logging.Logger.getLogger(Informe.class.
    getName()).log(java.util.logging.Level.SEVERE, null, ex);
79          } catch (InstantiationException ex) {
80              java.util.logging.Logger.getLogger(Informe.class.
    getName()).log(java.util.logging.Level.SEVERE, null, ex);
81          } catch (IllegalAccessException ex) {
82              java.util.logging.Logger.getLogger(Informe.class.
    getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
83          } catch (javax.swing.UnsupportedLookAndFeelException ex
    ) {
84              java.util.logging.Logger.getLogger(Informe.class.
    getName()).log(java.util.logging.Level.SEVERE, null, ex);
85          }
86          //</editor-fold>
87
88          /* Create and display the form */
89          java.awt.EventQueue.invokeLater(new Runnable() {
90              public void run() {
91                  new Informe().setVisible(true);
92              }
93          });
94      }
95
96      public void setText(String text)
97      {
98          textoInforme.setText(text);
99      }
100
101     public String getText()
102     {
103         return textoInforme.getText();
104     }
105
106     // Variables declaration - do not modify//GEN-BEGIN:
    variables
107     private javax.swing.JScrollPane jScrollPane1;
108     private javax.swing.JTextArea textoInforme;
109     // End of variables declaration//GEN-END:variables
110 }
```

Código 3.1: Archivo Informe.java

3.2. Clase Home

En este apartado se especifica de forma concisa la clase Home, que contiene los diferentes métodos para visualizar los distintos gráficos de la aplicación. Esta especificación se muestra en la Tabla 3.2.

Especificación de la clase Home

Clase: *Home*	
Esta clase contiene los diferentes métodos para visualizar los distintos gráficos de la aplicación.	
Métodos	
+ Home	Constructor de la clase Home
+ importarActionPerformed	Función para importar datos a la base de datos.
- almacenamientoImportarActionPerformed	JFileChooser para seleccionar el lugar de almacenamiento desde donde se desea importar la base de datos
- almacenamientoExportarActionPerformed	JFileChooser para seleccionar el lugar de almacenamiento donde se desea exportar la base de datos.
- exportarActionPerformed	Función para exportar la base de datos.
- importarDirectorioActionPerformed	Función para importar un directorio de archivos XML para añadirlo a la base de datos.
- GenerarPDF1ActionPerformed	Función para generar un archivo PDF con la fecha y la hora, del gráfico que se estuviera visualizando en el apartado Çomparar Cuentas"de la aplicación.
- SeleccionarRutaAlmacenamiento1ActionPerformed	Función para seleccionar el lugar de almacenamiento donde deseo guardar el PDF, dentro del apartado Comparar Cuentas, que se va a generar.
- OpcionesInformes1ActionPerformed	Función que recoge la opción seleccionada dentro del menú de opciones en el apartado Çomparar cuentas"de la aplicación.

- GenerarPDFActionPerformed	Función para generar un archivo PDF con la fecha y la hora, del gráfico que se estuviera visualizando en el apartado ”Visualizar Informes”de la aplicación.
- SeleccionarRutaAlmacenamientoActionPerformed	Función para seleccionar el lugar de almacenamiento donde deseo guardar el PDF, dentro del apartado ”Visualizar Informes”, que se va a generar.
- OpcionesInformesActionPerformed	Función que recoge la opción seleccionada dentro del menú de opciones en el apartado ”Visualizar Informes”de la aplicación.
- CuerposDeSeguridadActionPerformed	Función que selecciona el cuerpo de seguridad elegido por el usario en el menú destinado para ello.
- buscarHashtagsActionPerformed	Función que obtiene el total de repeticiones que ha tenido un hashtag concreto en una cuenta determinada.
- ImportarDatosActionPerformed	Función que importa los datos de una cuenta específica.
- PoliciaLocalDeMadridActionPerformed	Función que importa los datos de la cuenta de la Policía Local De Madrid.
- MossosDeEsquadraActionPerformed	Función que importa los datos de la cuenta de los Mossos d’Esquadra.
- DescargaGuardiaCivilActionPerformed	Función que importa los datos de la cuenta oficial de la Guardia Civil.
- PoliciaNacionalActionPerformed	Función que importa los datos de la cuenta oficial de la Policía Nacional.
- eliminarCuentaActionPerformed	Botón para eliminar una cuenta de la base de datos.
- eliminarCuenta	Función para eliminar una cuenta de la base de datos.
- importarDatosCuentaEspecifica	Función para importar datos de una cuenta específica desde el menú de descargas.
- importarDatosDirectorio	Función que importa archivos XML a la base de datos.

- generarInformeIdioma	Función que genera un gráfico con los idiomás más usados en la cuenta seleccionada.
- generarInformeRetweets	Función que genera un gráfico con el TOP 10 retweets de los tweets más retweeteados de la cuenta seleccionada.
- generarInformeFavoritos	Función que genera un gráfico con el TOP 10 favoritos de los tweets marcados como favoritos de la cuenta seleccionada.
- generarInformeTotalRepeticionesHashtags	Función que genera un gráfico con el total de repeticiones del hashtag introducido de la cuenta seleccionada.
- generarInformeTweetsEmitidosPeriodo	Función que genera un gráfico con el total de tweets emitidos por cada cuenta en el período seleccionado.
- generarInformeTotalSeguidoresCuentas	Función que genera un gráfico con el total de seguidores que tiene cada una de las 4 cuentas principales en ese momento.
- generarInformeTotalSeguidores	Función que genera un gráfico con el total de seguidores que tiene la cuenta seleccionada.
- GenerarInformeTotalRetweets	Función que genera un gráfico con el total de retweets que tiene cada una de las 4 cuentas principales.
- generarInformeTweetConMasRetweets	Función que genera un gráfico con el número de retweets que tiene el tweet más retweeteado, de cada una de las 4 cuentas principales.
- generarInformeTweetConMasFavoritos	Función que genera un gráfico con el número de favoritos que tiene el tweet con más favoritos, de cada una de las 4 cuentas principales.
- generarInformeTotalFavoritos	Función que genera un gráfico con la suma total de Favoritos que tiene cada una de las 4 cuentas principales.
- obtenerTweetsConMasRetweets	Función que obtiene el texto, la fecha y el número de retweets de la cuenta seleccionada, de aquellos tweets que tengan más retweets.

- obtenerTweetsConMasFavoritos	Función que obtiene el texto, la fecha y el número de favoritos de la cuenta seleccionada, de aquellos tweets que tengan más favoritos.
- convertirIndiceANombreCuenta	Función que el nombre de las cuentas a un indice numérico.
+ actualizarCuentas	Función que actualiza el total de cuentas que existe en la base de datos.
+ actualizarCuentas1	Función que actualiza el total de cuentas que existe en el menú de aquellas cuentas que pueden ser eliminadas.

Tabla 3.2: Especificación de los métodos de la clase Home

3.2.1. Fichero Home.java

```
1
2  package principal;
3  import baseDeDatos.DB;
4  import baseDeDatos.ImportWorker;
5
6  import baseDeDatos.Tweet;
7  import com.itextpdf.text.BadElementException;
8  import com.itextpdf.text.Document;
9  import com.itextpdf.text.DocumentException;
10 import com.itextpdf.text.Element;
11 import com.itextpdf.text.Image;
12 import com.itextpdf.text.PageSize;
13 import com.itextpdf.text.Paragraph;
14 import com.itextpdf.text.pdf.PdfContentByte;
15 import com.itextpdf.text.pdf.PdfWriter;
16 import com.toedter.calendar.JCalendar;
17 import java.awt.BorderLayout;
18 import java.awt.Color;
19 import java.awt.HeadlessException;
20 import java.io.File;
```

```
21 import java.io.FileNotFoundException;
22 import java.io.FileOutputStream;
23 import java.io.IOException;
24 import java.sql.ResultSet;
25 import java.sql.SQLException;
26 import java.text.DateFormat;
27 import java.text.ParseException;
28 import java.text.SimpleDateFormat;
29 import java.util.ArrayList;
30 import java.util.Calendar;
31 import java.util.Date;
32 import java.util.List;
33 import java.util.Locale;
34 import java.util.logging.Level;
35 import java.util.logging.Logger;
36 import javax.swing.JFileChooser;
37 import javax.swing.JOptionPane;
38 import javax.swing.filechooser.FileNameExtensionFilter;
39 import javax.xml.bind.JAXBContext;
40 import javax.xml.bind.JAXBException;
41 import javax.xml.bind.Unmarshaller;
42 import javax.xml.ws.BindingProvider;
43 import org.jfree.chart.ChartFactory;
44 import org.jfree.chart.JFreeChart;
45 import org.jfree.chart.plot.CategoryPlot;
46 import org.jfree.chart.plot.PlotOrientation;
47 import org.jfree.data.category.DefaultCategoryDataset;
48 import twitter4j.TwitterException;
49 import twittermanager.TwitterManager;
50 import org.jfree.chart.ChartPanel;
51 import org.jfree.chart.labels.ItemLabelAnchor;
52 import org.jfree.chart.labels.ItemLabelPosition;
53 import org.jfree.chart.labels.
     StandardCategoryItemLabelGenerator;
```

```
54 import org.jfree.chart.renderer.category.CategoryItemRenderer;
55 import org.jfree.data.category.CategoryDataset;
56 import org.jfree.data.xy.XYZDataset;
57 import org.jfree.ui.TextAnchor;
58 import org.jfree.util.TableOrder;
59
60
61 /**
62 * Clase Home  contiene los diferentes métodos para visualizar
     los distintos gráficos de la aplicación.
63 *
64 * @author Antonio Luis Ríos
65 * @version 20180626
66 */
67 public class Home extends javax.swing.JInternalFrame {
68
69     private static Home instance_ = null;
70
71     public static Home getInstance()
72     {
73         if(instance_ == null)
74             instance_ = new Home();
75         return instance_;
76     }
77
78     private JFreeChart chart;
79
80     /**
81      * Creates new form Home
82      */
83     public Home() {
84         initComponents();
85
86         actualizarCuentas();
```

```
87          actualizarCuentas1();
88          barraProgreso.setVisible(true);
89          jDateFecha1.setDate(new Date());
90          jDateFecha2.setDate(new Date());
91          jDateFecha3.setDate(new Date());
92          jDateFecha4.setDate(new Date());
93      }
94
95      /**
96       * This method is called from within the constructor to
    initialize the form.
97       * WARNING: Do NOT modify this code. The content of this
    method is always
98       * regenerated by the Form Editor.
99       */
100     @SuppressWarnings("unchecked")
101     // <editor-fold defaultstate="collapsed" desc="Generated
    Code">//GEN-BEGIN:initComponents
102     private void initComponents() {
103
104         jMenuItem1 = new javax.swing.JMenuItem();
105         imageBackgroundPanel1 = new principal.
    ImageBackgroundPanel();
106         imageBackgroundPanel2 = new principal.
    ImageBackgroundPanel();
107         jTabbedPane1 = new javax.swing.JTabbedPane();
108         imageBackgroundPanel3 = new principal.
    ImageBackgroundPanel();
109         jLabel17 = new javax.swing.JLabel();
110         jLabel19 = new javax.swing.JLabel();
111         imageBackgroundPanel4 = new principal.
    ImageBackgroundPanel();
112         PoliciaNacional = new javax.swing.JButton();
113         DescargaGuardiaCivil = new javax.swing.JButton();
```

```
114         MossosDeEsquadra = new javax.swing.JButton();
115         PoliciaLocalDeMadrid = new javax.swing.JButton();
116         jLabel5 = new javax.swing.JLabel();
117         nombreCuenta = new javax.swing.JTextField();
118         ImportarDatos = new javax.swing.JButton();
119         jLabel2 = new javax.swing.JLabel();
120         jLabel1 = new javax.swing.JLabel();
121         jLabel6 = new javax.swing.JLabel();
122         jLabel12 = new javax.swing.JLabel();
123         CuerposDeSeguridad1 = new javax.swing.JComboBox<>();
124         jLabel20 = new javax.swing.JLabel();
125         eliminarCuenta = new javax.swing.JButton();
126         imageBackgroundPanel5 = new principal.
    ImageBackgroundPanel();
127         jLabel7 = new javax.swing.JLabel();
128         buscarHashtag = new javax.swing.JTextField();
129         jLabel3 = new javax.swing.JLabel();
130         buscarHashtags = new javax.swing.JButton();
131         jLabel4 = new javax.swing.JLabel();
132         CuerposDeSeguridad = new javax.swing.JComboBox<>();
133         OpcionesInformes = new javax.swing.JComboBox<>();
134         jDateFecha1 = new com.toedter.calendar.JDateChooser();
135         jDateFecha2 = new com.toedter.calendar.JDateChooser();
136         SeleccionarRutaAlmacenamiento = new javax.swing.JButton
    ();
137         txtRuta = new javax.swing.JTextField();
138         GenerarPDF = new javax.swing.JButton();
139         chartContainer = new javax.swing.JPanel();
140         jLabel8 = new javax.swing.JLabel();
141         jLabel9 = new javax.swing.JLabel();
142         imageBackgroundPanel6 = new principal.
    ImageBackgroundPanel();
143         jLabel10 = new javax.swing.JLabel();
144         jDateFecha3 = new com.toedter.calendar.JDateChooser();
```

```
145         jLabel11 = new javax.swing.JLabel();
146         jDateFecha4 = new com.toedter.calendar.JDateChooser();
147         chartContainer1 = new javax.swing.JPanel();
148         OpcionesInformes1 = new javax.swing.JComboBox<>();
149         jLabel16 = new javax.swing.JLabel();
150         SeleccionarRutaAlmacenamiento1 = new javax.swing.
    JButton();
151         txtRuta3 = new javax.swing.JTextField();
152         GenerarPDF1 = new javax.swing.JButton();
153         imageBackgroundPanel7 = new principal.
    ImageBackgroundPanel();
154         importarDirectorio = new javax.swing.JButton();
155         exportar = new javax.swing.JButton();
156         almacenamientoExportar = new javax.swing.JButton();
157         txtRuta1 = new javax.swing.JTextField();
158         almacenamientoImportar = new javax.swing.JButton();
159         txtRuta2 = new javax.swing.JTextField();
160         importar = new javax.swing.JButton();
161         jLabel13 = new javax.swing.JLabel();
162         jLabel14 = new javax.swing.JLabel();
163         jLabel15 = new javax.swing.JLabel();
164         barraProgreso = new javax.swing.JProgressBar();
165         mensajeEstado = new javax.swing.JLabel();
166         jLabel18 = new javax.swing.JLabel();
167
168         jMenuItem1.setText("jMenuItem1");
169
170         imageBackgroundPanel1.setImage(new javax.swing.
    ImageIcon("/home/antonio/Dropbox/4º Ingenieria Informática/
    TFG/Aplicación/ejemplos/fondo3.jpg"));
171
172         jTabbedPane1.setBackground(java.awt.Color.orange);
173         jTabbedPane1.setBorder(new javax.swing.border.
    MatteBorder(null));
```

```
174         jTabbedPane1.setForeground(java.awt.Color.red);
175 
176         jLabel17.setIcon(new javax.swing.ImageIcon("/home/
    antonio/Dropbox/4º Ingenieria Informática/TFG/Aplicación/
    ejemplos/fotoefectos_opt1.jpg")); // NOI18N
177 
178         jLabel19.setIcon(new javax.swing.ImageIcon("/home/
    antonio/Dropbox/4º Ingenieria Informática/TFG/Aplicación/
    ejemplos/procesado tweets A_opt.png")); // NOI18N
179 
180         javax.swing.GroupLayout imageBackgroundPanel3Layout =
    new javax.swing.GroupLayout(imageBackgroundPanel3);
181         imageBackgroundPanel3.setLayout(
    imageBackgroundPanel3Layout);
182         imageBackgroundPanel3Layout.setHorizontalGroup(
183             imageBackgroundPanel3Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
184             .addGroup(imageBackgroundPanel3Layout.
    createSequentialGroup()
185                 .addGap(284, 284, 284)
186                 .addComponent(jLabel19, javax.swing.GroupLayout
    .PREFERRED_SIZE, 697, javax.swing.GroupLayout.PREFERRED_SIZE
    )
187                 .addGap(97, 97, 97)
188                 .addComponent(jLabel17)
189                 .addContainerGap(813, Short.MAX_VALUE))
190         );
191         imageBackgroundPanel3Layout.setVerticalGroup(
192             imageBackgroundPanel3Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
193             .addGroup(imageBackgroundPanel3Layout.
    createSequentialGroup()
194                 .addComponent(jLabel17)
195                 .addGap(0, 0, Short.MAX_VALUE))
```

```
196             .addGroup(imageBackgroundPanel3Layout.
    createSequentialGroup()
197                 .addGap(86, 86, 86)
198                 .addComponent(jLabel19, javax.swing.GroupLayout
    .PREFERRED_SIZE, 377, javax.swing.GroupLayout.PREFERRED_SIZE
    )
199                 .addContainerGap(261, Short.MAX_VALUE))
200         );
201 
202         jTabbedPane1.addTab("Inicio", imageBackgroundPanel3);
203 
204         PoliciaNacional.setBackground(new java.awt.Color(222,
    222, 222));
205         PoliciaNacional.setFont(new java.awt.Font("URW Gothic L
    ", 3, 15)); // NOI18N
206         PoliciaNacional.setForeground(java.awt.Color.white);
207         PoliciaNacional.setIcon(new javax.swing.ImageIcon("/
    home/antonio/Dropbox/4º Ingenieria Informática/TFG/Aplicació
    n/ejemplos/Policía-Nacional_1_opt.png")); // NOI18N
208         PoliciaNacional.setCursor(new java.awt.Cursor(java.awt.
    Cursor.HAND_CURSOR));
209         PoliciaNacional.setHorizontalTextPosition(javax.swing.
    SwingConstants.CENTER);
210         PoliciaNacional.setMaximumSize(new java.awt.Dimension
    (400, 400));
211         PoliciaNacional.addActionListener(new java.awt.event.
    ActionListener() {
212             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
213                 PoliciaNacionalActionPerformed(evt);
214             }
215         });
216 
```

```
217         DescargaGuardiaCivil.setBackground(java.awt.Color.white
    );
218         DescargaGuardiaCivil.setFont(new java.awt.Font("URW
    Gothic L", 3, 15)); // NOI18N
219         DescargaGuardiaCivil.setIcon(new javax.swing.ImageIcon(
    "/home/antonio/Dropbox/4° Ingenieria Informática/TFG/
    Aplicación/ejemplos/guardiacivilnuevo_opt.jpg")); // NOI18N
220         DescargaGuardiaCivil.setCursor(new java.awt.Cursor(java
    .awt.Cursor.HAND_CURSOR));
221         DescargaGuardiaCivil.setPreferredSize(new java.awt.
    Dimension(185, 206));
222         DescargaGuardiaCivil.addActionListener(new java.awt.
    event.ActionListener() {
223             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
224                 DescargaGuardiaCivilActionPerformed(evt);
225             }
226         });
227
228         MossosDeEsquadra.setBackground(java.awt.Color.white);
229         MossosDeEsquadra.setFont(new java.awt.Font("URW Gothic
    L", 3, 15)); // NOI18N
230         MossosDeEsquadra.setIcon(new javax.swing.ImageIcon("/
    home/antonio/Dropbox/4° Ingenieria Informática/TFG/Aplicació
    n/ejemplos/mossos2_opt.jpg")); // NOI18N
231         MossosDeEsquadra.setCursor(new java.awt.Cursor(java.awt
    .Cursor.HAND_CURSOR));
232         MossosDeEsquadra.setHorizontalTextPosition(javax.swing.
    SwingConstants.CENTER);
233         MossosDeEsquadra.setPreferredSize(new java.awt.
    Dimension(170, 161));
234         MossosDeEsquadra.addActionListener(new java.awt.event.
    ActionListener() {
```

```
235             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
236                 MossosDeEsquadraActionPerformed(evt);
237             }
238         });
239 
240         PoliciaLocalDeMadrid.setBackground(java.awt.Color.white
    );
241         PoliciaLocalDeMadrid.setFont(new java.awt.Font("URW
    Gothic L", 3, 15)); // NOI18N
242         PoliciaLocalDeMadrid.setIcon(new javax.swing.ImageIcon(
    "/home/antonio/Dropbox/4° Ingenieria Informática/TFG/
    Aplicación/ejemplos/policiamunicipalnuevo_opt.jpg")); //
    NOI18N
243         PoliciaLocalDeMadrid.setCursor(new java.awt.Cursor(java
    .awt.Cursor.HAND_CURSOR));
244         PoliciaLocalDeMadrid.setHorizontalTextPosition(javax.
    swing.SwingConstants.CENTER);
245         PoliciaLocalDeMadrid.addActionListener(new java.awt.
    event.ActionListener() {
246             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
247                 PoliciaLocalDeMadridActionPerformed(evt);
248             }
249         });
250 
251         jLabel5.setHorizontalAlignment(javax.swing.
    SwingConstants.CENTER);
252         jLabel5.setIcon(new javax.swing.ImageIcon("/home/
    antonio/Dropbox/4° Ingenieria Informática/TFG/Aplicación/
    ejemplos/twitter_7.png")); // NOI18N
253 
254         nombreCuenta.addActionListener(new java.awt.event.
    ActionListener() {
```

```
255             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
256                 nombreCuentaActionPerformed(evt);
257             }
258         });
259 
260         ImportarDatos.setBackground(new java.awt.Color(174,
    255, 0));
261         ImportarDatos.setFont(new java.awt.Font("URW Gothic L",
     3, 15)); // NOI18N
262         ImportarDatos.setText("Descargar e Importar datos");
263         ImportarDatos.setCursor(new java.awt.Cursor(java.awt.
    Cursor.HAND_CURSOR));
264         ImportarDatos.addActionListener(new java.awt.event.
    ActionListener() {
265             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
266                 ImportarDatosActionPerformed(evt);
267             }
268         });
269 
270         jLabel2.setFont(new java.awt.Font("URW Bookman L", 3,
    20)); // NOI18N
271         jLabel2.setForeground(java.awt.Color.white);
272         jLabel2.setText("Indica otra cuenta diferente para
    descargar tweets");
273 
274         jLabel1.setBackground(java.awt.Color.white);
275         jLabel1.setFont(new java.awt.Font("URW Bookman L", 3,
    20)); // NOI18N
276         jLabel1.setForeground(java.awt.Color.white);
277         jLabel1.setText("Selecciona el Cuerpo de Seguridad del
    que desea descargar los Tweets:");
278 
```

```
279         jLabel6.setFont(new java.awt.Font("Waree", 2, 14)); // NOI18N
280         jLabel6.setForeground(java.awt.Color.white);
281         jLabel6.setText("Nota: No escribir @ al inicio del nombre de la cuenta");
282 
283         jLabel12.setFont(new java.awt.Font("URW Gothic L", 3, 24)); // NOI18N
284         jLabel12.setForeground(java.awt.Color.white);
285         jLabel12.setText("@");
286 
287         CuerposDeSeguridad1.setFont(new java.awt.Font("DejaVu Serif", 0, 18)); // NOI18N
288         CuerposDeSeguridad1.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Policia Nacional", "Guardia Civil", "Policia Local de Madrid", "Mossos d'Esquadra" }));
289         CuerposDeSeguridad1.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
290         CuerposDeSeguridad1.addActionListener(new java.awt.event.ActionListener() {
291             public void actionPerformed(java.awt.event.ActionEvent evt) {
292                 CuerposDeSeguridad1ActionPerformed(evt);
293             }
294         });
295 
296         jLabel20.setFont(new java.awt.Font("URW Bookman L", 3, 20)); // NOI18N
297         jLabel20.setForeground(java.awt.Color.white);
298         jLabel20.setText("Seleccione la cuenta que desee eliminar");
299 
300         eliminarCuenta.setBackground(java.awt.Color.red);
```

```
301         eliminarCuenta.setFont(new java.awt.Font("URW Gothic L"
    , 3, 16)); // NOI18N
302         eliminarCuenta.setForeground(java.awt.Color.white);
303         eliminarCuenta.setText("Eliminar Cuenta");
304         eliminarCuenta.setCursor(new java.awt.Cursor(java.awt.
    Cursor.HAND_CURSOR));
305         eliminarCuenta.addActionListener(new java.awt.event.
    ActionListener() {
306             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
307                 eliminarCuentaActionPerformed(evt);
308             }
309         });
310 
311         javax.swing.GroupLayout imageBackgroundPanel4Layout =
    new javax.swing.GroupLayout(imageBackgroundPanel4);
312         imageBackgroundPanel4.setLayout(
    imageBackgroundPanel4Layout);
313         imageBackgroundPanel4Layout.setHorizontalGroup(
314             imageBackgroundPanel4Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
315             .addGroup(imageBackgroundPanel4Layout.
    createSequentialGroup()
316                 .addGroup(imageBackgroundPanel4Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
317                     .addGroup(imageBackgroundPanel4Layout.
    createSequentialGroup()
318                         .addGap(249, 249, 249)
319                         .addGroup(imageBackgroundPanel4Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
320                             .addComponent(jLabel1)
```

```
321                                 .addGroup(
    imageBackgroundPanel4Layout.createSequentialGroup()
322                                     .addComponent(PoliciaNacional,
    javax.swing.GroupLayout.PREFERRED_SIZE, 164, javax.swing.
    GroupLayout.PREFERRED_SIZE)
323                                     .addGap(37, 37, 37)
324                                     .addComponent(
    DescargaGuardiaCivil, javax.swing.GroupLayout.PREFERRED_SIZE
    , 163, javax.swing.GroupLayout.PREFERRED_SIZE)
325                                     .addGap(39, 39, 39)
326                                     .addComponent(MossosDeEsquadra,
     javax.swing.GroupLayout.PREFERRED_SIZE, 162, javax.swing.
    GroupLayout.PREFERRED_SIZE)
327                                     .addGap(41, 41, 41)
328                                     .addComponent(
    PoliciaLocalDeMadrid, javax.swing.GroupLayout.PREFERRED_SIZE
    , 162, javax.swing.GroupLayout.PREFERRED_SIZE))))
329                         .addGroup(imageBackgroundPanel4Layout.
    createSequentialGroup()
330                             .addGap(41, 41, 41)
331                             .addGroup(imageBackgroundPanel4Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
332                                 .addComponent(jLabel2, javax.swing.
    GroupLayout.PREFERRED_SIZE, 638, javax.swing.GroupLayout.
    PREFERRED_SIZE)
333                                 .addGroup(
    imageBackgroundPanel4Layout.createSequentialGroup()
334                                     .addComponent(jLabel5, javax.
    swing.GroupLayout.PREFERRED_SIZE, 86, javax.swing.
    GroupLayout.PREFERRED_SIZE)
335                                     .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.UNRELATED)
```

```
336                                     .addGroup(
    imageBackgroundPanel4Layout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.LEADING)
337                                         .addGroup(
    imageBackgroundPanel4Layout.createSequentialGroup()
338                                             .addComponent(jLabel12,
     javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.
    GroupLayout.PREFERRED_SIZE)
339                                             .addPreferredGap(javax.
    swing.LayoutStyle.ComponentPlacement.RELATED)
340                                             .addComponent(
    nombreCuenta, javax.swing.GroupLayout.PREFERRED_SIZE, 136,
    javax.swing.GroupLayout.PREFERRED_SIZE)
341                                             .addPreferredGap(javax.
    swing.LayoutStyle.ComponentPlacement.RELATED)
342                                             .addComponent(
    ImportarDatos))
343                                         .addComponent(jLabel6))))
344                         .addGroup(imageBackgroundPanel4Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
345                             .addGroup(
    imageBackgroundPanel4Layout.createSequentialGroup()
346                                 .addGap(121, 121, 121)
347                                 .addComponent(jLabel20, javax.
    swing.GroupLayout.PREFERRED_SIZE, 465, javax.swing.
    GroupLayout.PREFERRED_SIZE))
348                             .addGroup(
    imageBackgroundPanel4Layout.createSequentialGroup()
349                                 .addGap(283, 283, 283)
350                                 .addGroup(
    imageBackgroundPanel4Layout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.TRAILING)
```

```
351                                         .addComponent(
    eliminarCuenta, javax.swing.GroupLayout.PREFERRED_SIZE, 175,
     javax.swing.GroupLayout.PREFERRED_SIZE)
352                                         .addComponent(
    CuerposDeSeguridad1, javax.swing.GroupLayout.PREFERRED_SIZE,
     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.
    GroupLayout.PREFERRED_SIZE))))))
353                 .addContainerGap(826, Short.MAX_VALUE))
354         );
355         imageBackgroundPanel4Layout.setVerticalGroup(
356             imageBackgroundPanel4Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
357             .addGroup(imageBackgroundPanel4Layout.
    createSequentialGroup()
358                 .addGap(32, 32, 32)
359                 .addComponent(jLabel1, javax.swing.GroupLayout.
    PREFERRED_SIZE, 48, javax.swing.GroupLayout.PREFERRED_SIZE)
360                 .addGap(37, 37, 37)
361                 .addGroup(imageBackgroundPanel4Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING, false)
362                     .addComponent(PoliciaNacional, javax.swing.
    GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.
    PREFERRED_SIZE, 160, javax.swing.GroupLayout.PREFERRED_SIZE)
363                     .addComponent(DescargaGuardiaCivil, javax.
    swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
364                     .addComponent(MossosDeEsquadra, javax.swing
    .GroupLayout.PREFERRED_SIZE, 160, javax.swing.GroupLayout.
    PREFERRED_SIZE)
365                     .addComponent(PoliciaLocalDeMadrid, javax.
    swing.GroupLayout.PREFERRED_SIZE, 160, javax.swing.
    GroupLayout.PREFERRED_SIZE))
366                 .addGap(107, 107, 107)
```

```
367                 .addGroup(imageBackgroundPanel4Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
368                     .addGroup(imageBackgroundPanel4Layout.
    createSequentialGroup()
369                         .addComponent(jLabel20, javax.swing.
    GroupLayout.PREFERRED_SIZE, 34, javax.swing.GroupLayout.
    PREFERRED_SIZE)
370                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.UNRELATED)
371                         .addComponent(CuerposDeSeguridad1,
    javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
    PREFERRED_SIZE)
372                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
373                         .addComponent(eliminarCuenta))
374                     .addGroup(imageBackgroundPanel4Layout.
    createSequentialGroup()
375                         .addComponent(jLabel2, javax.swing.
    GroupLayout.PREFERRED_SIZE, 34, javax.swing.GroupLayout.
    PREFERRED_SIZE)
376                         .addGroup(imageBackgroundPanel4Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    TRAILING)
377                             .addGroup(
    imageBackgroundPanel4Layout.createSequentialGroup()
378                                 .addGroup(
    imageBackgroundPanel4Layout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.LEADING, false)
379                                     .addGroup(
    imageBackgroundPanel4Layout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.BASELINE)
```

```
380                                                 .addComponent(
    nombreCuenta)
381                                                 .addComponent(jLabel12)
    )
382                                             .addComponent(ImportarDatos
    , javax.swing.GroupLayout.PREFERRED_SIZE, 27, javax.swing.
    GroupLayout.PREFERRED_SIZE))
383                                     .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
384                                     .addComponent(jLabel6, javax.
    swing.GroupLayout.PREFERRED_SIZE, 24, javax.swing.
    GroupLayout.PREFERRED_SIZE)
385                                     .addGap(12, 12, 12))
386                                 .addComponent(jLabel5, javax.swing.
    GroupLayout.PREFERRED_SIZE, 89, javax.swing.GroupLayout.
    PREFERRED_SIZE))))
387                 .addContainerGap(217, Short.MAX_VALUE))
388         );
389
390         jTabbedPane1.addTab("Descargar Datos",
    imageBackgroundPanel4);
391
392         jLabel7.setFont(new java.awt.Font("DejaVu Serif", 3,
    16)); // NOI18N
393         jLabel7.setForeground(java.awt.Color.white);
394         jLabel7.setText("Seleccione un Cuerpo de Seguridad para
     visualizar sus informes:");
395
396         jLabel3.setBackground(java.awt.Color.white);
397         jLabel3.setFont(new java.awt.Font("DejaVu Serif", 3,
    16)); // NOI18N
398         jLabel3.setForeground(new java.awt.Color(254, 250, 107)
    );
```

```
399         jLabel3.setText("Comprobar el número de apariciones de
    un Hashtag");
400 
401         buscarHashtags.setBackground(new java.awt.Color(174,
    255, 0));
402         buscarHashtags.setFont(new java.awt.Font("URW Gothic L"
    , 3, 15)); // NOI18N
403         buscarHashtags.setText("Bucar Hashtag");
404         buscarHashtags.setCursor(new java.awt.Cursor(java.awt.
    Cursor.HAND_CURSOR));
405         buscarHashtags.addActionListener(new java.awt.event.
    ActionListener() {
406             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
407                 buscarHashtagsActionPerformed(evt);
408             }
409         });
410 
411         jLabel4.setFont(new java.awt.Font("Waree", 2, 14)); //
    NOI18N
412         jLabel4.setForeground(java.awt.Color.white);
413         jLabel4.setText("Nota: Debes introducir # al inicio del
     hashtag");
414 
415         CuerposDeSeguridad.setFont(new java.awt.Font("DejaVu
    Serif", 0, 18)); // NOI18N
416         CuerposDeSeguridad.setModel(new javax.swing.
    DefaultComboBoxModel<>(new String[] { "Policia Nacional", "
    Guardia Civil", "Policia Local de Madrid", "Mossos d'
    Esquadra" }));
417         CuerposDeSeguridad.setCursor(new java.awt.Cursor(java.
    awt.Cursor.HAND_CURSOR));
418         CuerposDeSeguridad.addActionListener(new java.awt.event
    .ActionListener() {
```

```
419             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
420                 CuerposDeSeguridadActionPerformed(evt);
421             }
422         });
423 
424         OpcionesInformes.setFont(new java.awt.Font("DejaVu
    Serif", 0, 15)); // NOI18N
425         OpcionesInformes.setModel(new javax.swing.
    DefaultComboBoxModel<>(new String[] { "-- Seleccione una de
    las siguientes opciones --", "Top 10 Retweets", "Top 10
    Favoritos", "Idiomas Usados en los Tweets", "Total de
    Seguidores de cada cuenta", "Cantidad de Tweets emitidos en
    un  período de tiempo...", "Texto del Top 10 Retweets", "
    Texto del Top 10 Favoritos" }));
426         OpcionesInformes.setCursor(new java.awt.Cursor(java.awt
    .Cursor.HAND_CURSOR));
427         OpcionesInformes.addActionListener(new java.awt.event.
    ActionListener() {
428             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
429                 OpcionesInformesActionPerformed(evt);
430             }
431         });
432 
433         SeleccionarRutaAlmacenamiento.setBackground(java.awt.
    Color.lightGray);
434         SeleccionarRutaAlmacenamiento.setFont(new java.awt.Font
    ("Waree", 3, 12)); // NOI18N
435         SeleccionarRutaAlmacenamiento.setForeground(java.awt.
    Color.black);
436         SeleccionarRutaAlmacenamiento.setIcon(new javax.swing.
    ImageIcon("/home/antonio/Dropbox/4° Ingenieria Informática/
    TFG/Aplicación/ejemplos/almacenamiento4_opt.jpg")); //
```

```
    NOI18N
437         SeleccionarRutaAlmacenamiento.setText("Seleccionar
    almacenamiento");
438         SeleccionarRutaAlmacenamiento.setCursor(new java.awt.
    Cursor(java.awt.Cursor.HAND_CURSOR));
439         SeleccionarRutaAlmacenamiento.setHorizontalAlignment(
    javax.swing.SwingConstants.LEFT);
440         SeleccionarRutaAlmacenamiento.setHorizontalTextPosition
    (javax.swing.SwingConstants.LEFT);
441         SeleccionarRutaAlmacenamiento.setIconTextGap(8);
442         SeleccionarRutaAlmacenamiento.addActionListener(new
    java.awt.event.ActionListener() {
443             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
444                 SeleccionarRutaAlmacenamientoActionPerformed(
    evt);
445             }
446         });
447
448         GenerarPDF.setBackground(new java.awt.Color(174, 255,
    0));
449         GenerarPDF.setFont(new java.awt.Font("URW Gothic L", 3,
     15)); // NOI18N
450         GenerarPDF.setText("Generar PDF");
451         GenerarPDF.setCursor(new java.awt.Cursor(java.awt.
    Cursor.HAND_CURSOR));
452         GenerarPDF.addActionListener(new java.awt.event.
    ActionListener() {
453             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
454                 GenerarPDFActionPerformed(evt);
455             }
456         });
457
```

```
458         chartContainer.setBackground(new java.awt.Color(254,
    254, 254));
459         chartContainer.setPreferredSize(new java.awt.Dimension
    (300, 346));
460         chartContainer.setLayout(new java.awt.BorderLayout());
461 
462         jLabel8.setFont(new java.awt.Font("URW Bookman L", 3,
    15)); // NOI18N
463         jLabel8.setForeground(java.awt.Color.white);
464         jLabel8.setText("Fecha Inicio");
465 
466         jLabel9.setFont(new java.awt.Font("URW Bookman L", 3,
    15)); // NOI18N
467         jLabel9.setForeground(java.awt.Color.white);
468         jLabel9.setText("Fecha Fin");
469 
470         javax.swing.GroupLayout imageBackgroundPanel5Layout =
    new javax.swing.GroupLayout(imageBackgroundPanel5);
471         imageBackgroundPanel5.setLayout(
    imageBackgroundPanel5Layout);
472         imageBackgroundPanel5Layout.setHorizontalGroup(
473             imageBackgroundPanel5Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
474             .addGroup(imageBackgroundPanel5Layout.
    createSequentialGroup()
475                 .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
476                     .addComponent(chartContainer, javax.swing.
    GroupLayout.PREFERRED_SIZE, 1345, javax.swing.GroupLayout.
    PREFERRED_SIZE)
477                     .addGroup(imageBackgroundPanel5Layout.
    createSequentialGroup()
478                         .addGap(39, 39, 39)
```

```
479                         .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
480                             .addComponent(CuerposDeSeguridad,
    javax.swing.GroupLayout.PREFERRED_SIZE, 214, javax.swing.
    GroupLayout.PREFERRED_SIZE)
481                             .addComponent(OpcionesInformes,
    javax.swing.GroupLayout.PREFERRED_SIZE, 477, javax.swing.
    GroupLayout.PREFERRED_SIZE)
482                             .addComponent(jLabel7))
483                         .addGap(118, 118, 118)
484                         .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    TRAILING)
485                             .addComponent(jLabel3)
486                             .addComponent(jLabel4)
487                             .addGroup(
    imageBackgroundPanel5Layout.createSequentialGroup()
488                                 .addComponent(buscarHashtag,
    javax.swing.GroupLayout.PREFERRED_SIZE, 163, javax.swing.
    GroupLayout.PREFERRED_SIZE)
489                                 .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
490                                 .addComponent(buscarHashtags)))
    )
491                     .addGroup(imageBackgroundPanel5Layout.
    createSequentialGroup()
492                         .addGap(62, 62, 62)
493                         .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
494                             .addGroup(
    imageBackgroundPanel5Layout.createSequentialGroup()
495                                 .addGap(33, 33, 33)
```

```
496                                     .addComponent(jLabel8)
497                                     .addGap(85, 85, 85)
498                                     .addComponent(jLabel9))
499                                 .addGroup(
    imageBackgroundPanel5Layout.createSequentialGroup()
500                                     .addComponent(jDateFecha1,
    javax.swing.GroupLayout.PREFERRED_SIZE, 163, javax.swing.
    GroupLayout.PREFERRED_SIZE)
501                                     .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
502                                     .addComponent(jDateFecha2,
    javax.swing.GroupLayout.PREFERRED_SIZE, 161, javax.swing.
    GroupLayout.PREFERRED_SIZE)))
503                             .addGap(224, 224, 224)
504                             .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
505                                 .addComponent(txtRuta, javax.swing.
    GroupLayout.PREFERRED_SIZE, 267, javax.swing.GroupLayout.
    PREFERRED_SIZE)
506                                 .addComponent(
    SeleccionarRutaAlmacenamiento, javax.swing.GroupLayout.
    PREFERRED_SIZE, 267, javax.swing.GroupLayout.PREFERRED_SIZE)
507                                 .addGroup(
    imageBackgroundPanel5Layout.createSequentialGroup()
508                                     .addGap(90, 90, 90)
509                                     .addComponent(GenerarPDF)))))
510                 .addContainerGap(746, Short.MAX_VALUE))
511         );
512         imageBackgroundPanel5Layout.setVerticalGroup(
513             imageBackgroundPanel5Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
514             .addGroup(imageBackgroundPanel5Layout.
    createSequentialGroup()
```

```
515                 .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
516                     .addGroup(imageBackgroundPanel5Layout.
    createSequentialGroup()
517                         .addGap(31, 31, 31)
518                         .addComponent(jLabel7, javax.swing.
    GroupLayout.PREFERRED_SIZE, 38, javax.swing.GroupLayout.
    PREFERRED_SIZE)
519                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
520                         .addComponent(CuerposDeSeguridad, javax
    .swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
521                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
522                         .addComponent(OpcionesInformes, javax.
    swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
523                     .addGroup(imageBackgroundPanel5Layout.
    createSequentialGroup()
524                         .addGap(28, 28, 28)
525                         .addComponent(jLabel3, javax.swing.
    GroupLayout.PREFERRED_SIZE, 38, javax.swing.GroupLayout.
    PREFERRED_SIZE)
526                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.UNRELATED)
527                         .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
528                             .addComponent(buscarHashtag, javax.
    swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
529                             .addComponent(buscarHashtags))
```

```
530                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
531                         .addComponent(jLabel4)))
532                 .addGap(22, 22, 22)
533                 .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
534                     .addGroup(javax.swing.GroupLayout.Alignment
    .TRAILING, imageBackgroundPanel5Layout.createSequentialGroup
    ()
535                         .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
536                             .addComponent(jLabel8)
537                             .addComponent(jLabel9))
538                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.UNRELATED)
539                         .addGroup(imageBackgroundPanel5Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
540                             .addComponent(jDateFecha1, javax.
    swing.GroupLayout.PREFERRED_SIZE, 27, javax.swing.
    GroupLayout.PREFERRED_SIZE)
541                             .addComponent(jDateFecha2, javax.
    swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
542                         .addGap(37, 37, 37))
543                     .addGroup(javax.swing.GroupLayout.Alignment
    .TRAILING, imageBackgroundPanel5Layout.createSequentialGroup
    ()
544                         .addComponent(
    SeleccionarRutaAlmacenamiento, javax.swing.GroupLayout.
    PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
545                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
546                         .addComponent(txtRuta, javax.swing.
    GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
547                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
548                         .addComponent(GenerarPDF)
549                         .addGap(8, 8, 8)))
550                 .addComponent(chartContainer, javax.swing.
    GroupLayout.DEFAULT_SIZE, 444, Short.MAX_VALUE)
551                 .addContainerGap())
552         );
553 
554         jTabbedPane1.addTab("Visualizar Informes",
    imageBackgroundPanel5);
555 
556         jLabel10.setFont(new java.awt.Font("URW Bookman L", 3,
    15)); // NOI18N
557         jLabel10.setForeground(java.awt.Color.white);
558         jLabel10.setText("Fecha Inicio");
559 
560         jLabel11.setFont(new java.awt.Font("URW Bookman L", 3,
    15)); // NOI18N
561         jLabel11.setForeground(java.awt.Color.white);
562         jLabel11.setText("Fecha Fin");
563 
564         chartContainer1.setBackground(new java.awt.Color(254,
    254, 254));
565         chartContainer1.setPreferredSize(new java.awt.Dimension
    (300, 346));
566         chartContainer1.setLayout(new java.awt.BorderLayout());
567 
```

```
568         OpcionesInformes1.setFont(new java.awt.Font("DejaVu
    Serif", 0, 15)); // NOI18N
569         OpcionesInformes1.setModel(new javax.swing.
    DefaultComboBoxModel<>(new String[] { "-- Seleccione una de
    las siguientes opciones --", "Total Retweets", "Total
    Favoritos", "Tweet con mayor número de Retweets", "Tweet con
     mayor número de Favoritos", "Número de seguidores" }));
570         OpcionesInformes1.setCursor(new java.awt.Cursor(java.
    awt.Cursor.HAND_CURSOR));
571         OpcionesInformes1.addActionListener(new java.awt.event.
    ActionListener() {
572             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
573                 OpcionesInformes1ActionPerformed(evt);
574             }
575         });
576
577         jLabel16.setFont(new java.awt.Font("DejaVu Serif", 3,
    18)); // NOI18N
578         jLabel16.setForeground(java.awt.Color.white);
579         jLabel16.setText("Seleccione una de las siguientes
    opciones para visualizar la comparativa entre cuentas:");
580
581         SeleccionarRutaAlmacenamiento1.setBackground(java.awt.
    Color.lightGray);
582         SeleccionarRutaAlmacenamiento1.setFont(new java.awt.
    Font("Waree", 3, 12)); // NOI18N
583         SeleccionarRutaAlmacenamiento1.setForeground(java.awt.
    Color.black);
584         SeleccionarRutaAlmacenamiento1.setIcon(new javax.swing.
    ImageIcon("/home/antonio/Dropbox/4° Ingenieria Informática/
    TFG/Aplicación/ejemplos/almacenamiento4_opt.jpg")); //
    NOI18N
```

```
585         SeleccionarRutaAlmacenamiento1.setText("Seleccionar
    almacenamiento");
586         SeleccionarRutaAlmacenamiento1.setCursor(new java.awt.
    Cursor(java.awt.Cursor.HAND_CURSOR));
587         SeleccionarRutaAlmacenamiento1.setHorizontalAlignment(
    javax.swing.SwingConstants.LEFT);
588         SeleccionarRutaAlmacenamiento1.
    setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
589         SeleccionarRutaAlmacenamiento1.setIconTextGap(8);
590         SeleccionarRutaAlmacenamiento1.addActionListener(new
    java.awt.event.ActionListener() {
591             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
592                 SeleccionarRutaAlmacenamiento1ActionPerformed(
    evt);
593             }
594         });
595
596         GenerarPDF1.setBackground(new java.awt.Color(174, 255,
    0));
597         GenerarPDF1.setFont(new java.awt.Font("URW Gothic L",
    3, 15)); // NOI18N
598         GenerarPDF1.setText("Generar PDF");
599         GenerarPDF1.setCursor(new java.awt.Cursor(java.awt.
    Cursor.HAND_CURSOR));
600         GenerarPDF1.addActionListener(new java.awt.event.
    ActionListener() {
601             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
602                 GenerarPDF1ActionPerformed(evt);
603             }
604         });
605
```

```
606         javax.swing.GroupLayout imageBackgroundPanel6Layout =
    new javax.swing.GroupLayout(imageBackgroundPanel6);
607         imageBackgroundPanel6.setLayout(
    imageBackgroundPanel6Layout);
608         imageBackgroundPanel6Layout.setHorizontalGroup(
609             imageBackgroundPanel6Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
610             .addGroup(imageBackgroundPanel6Layout.
    createSequentialGroup()
611                 .addGroup(imageBackgroundPanel6Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
612                     .addGroup(imageBackgroundPanel6Layout.
    createSequentialGroup()
613                         .addGap(29, 29, 29)
614                         .addGroup(imageBackgroundPanel6Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
615                             .addComponent(OpcionesInformes1,
    javax.swing.GroupLayout.PREFERRED_SIZE, 432, javax.swing.
    GroupLayout.PREFERRED_SIZE)
616                             .addComponent(jLabel16)
617                             .addGroup(
    imageBackgroundPanel6Layout.createSequentialGroup()
618                                 .addGroup(
    imageBackgroundPanel6Layout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.LEADING)
619                                     .addGroup(
    imageBackgroundPanel6Layout.createSequentialGroup()
620                                         .addGap(33, 33, 33)
621                                         .addComponent(jLabel10)
    )
622                                     .addComponent(jDateFecha3,
    javax.swing.GroupLayout.PREFERRED_SIZE, 163, javax.swing.
```

```
    GroupLayout.PREFERRED_SIZE))
623                                 .addGroup(
    imageBackgroundPanel6Layout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.LEADING)
624                                     .addGroup(
    imageBackgroundPanel6Layout.createSequentialGroup()
625                                         .addGap(81, 81, 81)
626                                         .addComponent(jLabel11)
    )
627                                     .addGroup(
    imageBackgroundPanel6Layout.createSequentialGroup()
628                                         .addGap(47, 47, 47)
629                                         .addComponent(
    jDateFecha4, javax.swing.GroupLayout.PREFERRED_SIZE, 161,
    javax.swing.GroupLayout.PREFERRED_SIZE)))
630                                 .addGap(259, 259, 259)
631                                 .addGroup(
    imageBackgroundPanel6Layout.createParallelGroup(javax.swing.
    GroupLayout.Alignment.LEADING)
632                                     .addComponent(txtRuta3,
    javax.swing.GroupLayout.PREFERRED_SIZE, 267, javax.swing.
    GroupLayout.PREFERRED_SIZE)
633                                     .addComponent(
    SeleccionarRutaAlmacenamiento1, javax.swing.GroupLayout.
    PREFERRED_SIZE, 267, javax.swing.GroupLayout.PREFERRED_SIZE)
634                                     .addGroup(
    imageBackgroundPanel6Layout.createSequentialGroup()
635                                         .addGap(90, 90, 90)
636                                         .addComponent(
    GenerarPDF1))))))
637                     .addComponent(chartContainer1, javax.swing.
    GroupLayout.PREFERRED_SIZE, 1296, javax.swing.GroupLayout.
    PREFERRED_SIZE))
638                 .addContainerGap(795, Short.MAX_VALUE))
```

```
639         );
640         imageBackgroundPanel6Layout.setVerticalGroup(
641             imageBackgroundPanel6Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
642             .addGroup(imageBackgroundPanel6Layout.
    createSequentialGroup()
643                 .addGap(24, 24, 24)
644                 .addComponent(jLabel16, javax.swing.GroupLayout
    .PREFERRED_SIZE, 38, javax.swing.GroupLayout.PREFERRED_SIZE)
645                 .addGap(21, 21, 21)
646                 .addComponent(OpcionesInformes1, javax.swing.
    GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
647                 .addGroup(imageBackgroundPanel6Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
648                     .addGroup(imageBackgroundPanel6Layout.
    createSequentialGroup()
649                         .addGap(55, 55, 55)
650                         .addGroup(imageBackgroundPanel6Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    TRAILING)
651                             .addGroup(
    imageBackgroundPanel6Layout.createSequentialGroup()
652                                 .addComponent(jLabel10)
653                                 .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
654                                 .addComponent(jDateFecha3,
    javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
    PREFERRED_SIZE))
655                             .addGroup(
    imageBackgroundPanel6Layout.createSequentialGroup()
656                                 .addComponent(jLabel11)
```

```
657                                     .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
658                                     .addComponent(jDateFecha4,
    javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.
    GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
    PREFERRED_SIZE))))
659                     .addGroup(imageBackgroundPanel6Layout.
    createSequentialGroup()
660                         .addGap(21, 21, 21)
661                         .addComponent(
    SeleccionarRutaAlmacenamiento1, javax.swing.GroupLayout.
    PREFERRED_SIZE, 32, javax.swing.GroupLayout.PREFERRED_SIZE)
662                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
663                         .addComponent(txtRuta3, javax.swing.
    GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
664                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
665                         .addComponent(GenerarPDF1)))
666                 .addGap(28, 28, 28)
667                 .addComponent(chartContainer1, javax.swing.
    GroupLayout.DEFAULT_SIZE, 455, Short.MAX_VALUE)
668                 .addContainerGap())
669         );
670 
671         jTabbedPane1.addTab("Comparar Cuentas",
    imageBackgroundPanel6);
672 
673         importarDirectorio.setBackground(new java.awt.Color
    (250, 250, 121));
674         importarDirectorio.setFont(new java.awt.Font("URW
    Gothic L", 3, 15)); // NOI18N
675         importarDirectorio.setForeground(java.awt.Color.black);
```

```
676         importarDirectorio.setText("Seleccionar Directorio");
677         importarDirectorio.setCursor(new java.awt.Cursor(java.
    awt.Cursor.HAND_CURSOR));
678         importarDirectorio.addActionListener(new java.awt.event
    .ActionListener() {
679             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
680                 importarDirectorioActionPerformed(evt);
681             }
682         });
683 
684         exportar.setBackground(new java.awt.Color(174, 255, 0))
    ;
685         exportar.setFont(new java.awt.Font("URW Gothic L", 3,
    15)); // NOI18N
686         exportar.setText("Exportar");
687         exportar.setCursor(new java.awt.Cursor(java.awt.Cursor.
    HAND_CURSOR));
688         exportar.addActionListener(new java.awt.event.
    ActionListener() {
689             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
690                 exportarActionPerformed(evt);
691             }
692         });
693 
694         almacenamientoExportar.setBackground(java.awt.Color.
    lightGray);
695         almacenamientoExportar.setFont(new java.awt.Font("Waree
    ", 3, 12)); // NOI18N
696         almacenamientoExportar.setForeground(java.awt.Color.
    black);
697         almacenamientoExportar.setIcon(new javax.swing.
    ImageIcon("/home/antonio/Dropbox/4° Ingenieria Informática/
```

```
    TFG/Aplicación/ejemplos/almacenamiento4_opt.jpg")); //
    NOI18N
698         almacenamientoExportar.setText("Seleccionar
    almacenamiento");
699         almacenamientoExportar.setCursor(new java.awt.Cursor(
    java.awt.Cursor.HAND_CURSOR));
700         almacenamientoExportar.setHorizontalAlignment(javax.
    swing.SwingConstants.LEFT);
701         almacenamientoExportar.setHorizontalTextPosition(javax.
    swing.SwingConstants.LEFT);
702         almacenamientoExportar.setIconTextGap(8);
703         almacenamientoExportar.addActionListener(new java.awt.
    event.ActionListener() {
704             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
705                 almacenamientoExportarActionPerformed(evt);
706             }
707         });
708
709         almacenamientoImportar.setBackground(java.awt.Color.
    lightGray);
710         almacenamientoImportar.setFont(new java.awt.Font("Waree
    ", 3, 12)); // NOI18N
711         almacenamientoImportar.setForeground(java.awt.Color.
    black);
712         almacenamientoImportar.setIcon(new javax.swing.
    ImageIcon("/home/antonio/Dropbox/4º Ingenieria Informática/
    TFG/Aplicación/ejemplos/almacenamiento4_opt.jpg")); //
    NOI18N
713         almacenamientoImportar.setText("Seleccionar
    almacenamiento");
714         almacenamientoImportar.setCursor(new java.awt.Cursor(
    java.awt.Cursor.HAND_CURSOR));
```

```
715         almacenamientoImportar.setHorizontalAlignment(javax.
    swing.SwingConstants.LEFT);
716         almacenamientoImportar.setHorizontalTextPosition(javax.
    swing.SwingConstants.LEFT);
717         almacenamientoImportar.setIconTextGap(8);
718         almacenamientoImportar.addActionListener(new java.awt.
    event.ActionListener() {
719             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
720                 almacenamientoImportarActionPerformed(evt);
721             }
722         });
723 
724         importar.setBackground(new java.awt.Color(174, 255, 0))
    ;
725         importar.setFont(new java.awt.Font("URW Gothic L", 3,
    15)); // NOI18N
726         importar.setText("Importar");
727         importar.setCursor(new java.awt.Cursor(java.awt.Cursor.
    HAND_CURSOR));
728         importar.addActionListener(new java.awt.event.
    ActionListener() {
729             public void actionPerformed(java.awt.event.
    ActionEvent evt) {
730                 importarActionPerformed(evt);
731             }
732         });
733 
734         jLabel13.setBackground(java.awt.Color.white);
735         jLabel13.setFont(new java.awt.Font("URW Bookman L", 3,
    18)); // NOI18N
736         jLabel13.setForeground(java.awt.Color.white);
737         jLabel13.setText("Importar directorio con archivos XML
    a la Base de Datos");
```

```
738
739         jLabel14.setBackground(java.awt.Color.white);
740         jLabel14.setFont(new java.awt.Font("URW Bookman L", 3,
    18)); // NOI18N
741         jLabel14.setForeground(java.awt.Color.white);
742         jLabel14.setText("Seleccione el lugar de almacenamiento
     desde donde desea importar la Base de Datos");
743
744         jLabel15.setBackground(java.awt.Color.white);
745         jLabel15.setFont(new java.awt.Font("URW Bookman L", 3,
    18)); // NOI18N
746         jLabel15.setForeground(java.awt.Color.white);
747         jLabel15.setText("Seleccione el lugar de almacenamiento
     donde desea exportar la Base de Datos");
748
749         javax.swing.GroupLayout imageBackgroundPanel7Layout =
    new javax.swing.GroupLayout(imageBackgroundPanel7);
750         imageBackgroundPanel7.setLayout(
    imageBackgroundPanel7Layout);
751         imageBackgroundPanel7Layout.setHorizontalGroup(
752             imageBackgroundPanel7Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
753             .addGroup(imageBackgroundPanel7Layout.
    createSequentialGroup()
754                 .addGroup(imageBackgroundPanel7Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
755                     .addGroup(imageBackgroundPanel7Layout.
    createSequentialGroup()
756                         .addGap(257, 257, 257)
757                         .addGroup(imageBackgroundPanel7Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
758                             .addComponent(jLabel14)
```

```
759                                 .addGroup(
    imageBackgroundPanel7Layout.createSequentialGroup()
760                                     .addComponent(
    almacenamientoImportar)
761                                     .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.UNRELATED)
762                                     .addComponent(txtRuta2, javax.
    swing.GroupLayout.PREFERRED_SIZE, 267, javax.swing.
    GroupLayout.PREFERRED_SIZE)
763                                     .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.UNRELATED)
764                                     .addComponent(importar, javax.
    swing.GroupLayout.PREFERRED_SIZE, 94, javax.swing.
    GroupLayout.PREFERRED_SIZE))
765                                 .addComponent(jLabel15)))
766                         .addGroup(imageBackgroundPanel7Layout.
    createSequentialGroup()
767                             .addGap(256, 256, 256)
768                             .addGroup(imageBackgroundPanel7Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
769                                 .addComponent(jLabel13)
770                                 .addGroup(
    imageBackgroundPanel7Layout.createSequentialGroup()
771                                     .addComponent(
    almacenamientoExportar)
772                                     .addGap(18, 18, 18)
773                                     .addComponent(txtRuta1, javax.
    swing.GroupLayout.PREFERRED_SIZE, 267, javax.swing.
    GroupLayout.PREFERRED_SIZE)
774                                     .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.UNRELATED)
775                                     .addComponent(exportar, javax.
    swing.GroupLayout.PREFERRED_SIZE, 94, javax.swing.
```

```
    GroupLayout.PREFERRED_SIZE))
776                             .addComponent(importarDirectorio,
    javax.swing.GroupLayout.PREFERRED_SIZE, 279, javax.swing.
    GroupLayout.PREFERRED_SIZE))))
777                 .addContainerGap(1010, Short.MAX_VALUE))
778         );
779         imageBackgroundPanel7Layout.setVerticalGroup(
780             imageBackgroundPanel7Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
781             .addGroup(imageBackgroundPanel7Layout.
    createSequentialGroup()
782                 .addGap(77, 77, 77)
783                 .addComponent(jLabel14, javax.swing.GroupLayout
    .PREFERRED_SIZE, 48, javax.swing.GroupLayout.PREFERRED_SIZE)
784                 .addGap(18, 18, 18)
785                 .addGroup(imageBackgroundPanel7Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
786                     .addComponent(almacenamientoImportar, javax
    .swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.
    GroupLayout.PREFERRED_SIZE)
787                     .addComponent(txtRuta2, javax.swing.
    GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
788                     .addComponent(importar))
789                 .addGap(70, 70, 70)
790                 .addComponent(jLabel15, javax.swing.GroupLayout
    .PREFERRED_SIZE, 48, javax.swing.GroupLayout.PREFERRED_SIZE)
791                 .addPreferredGap(javax.swing.LayoutStyle.
    ComponentPlacement.RELATED)
792                 .addGroup(imageBackgroundPanel7Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
```

```
793                     .addComponent(almacenamientoExportar, javax
    .swing.GroupLayout.PREFERRED_SIZE, 32, javax.swing.
    GroupLayout.PREFERRED_SIZE)
794                     .addComponent(txtRuta1, javax.swing.
    GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
795                     .addComponent(exportar))
796                 .addGap(61, 61, 61)
797                 .addComponent(jLabel13, javax.swing.GroupLayout
    .PREFERRED_SIZE, 48, javax.swing.GroupLayout.PREFERRED_SIZE)
798                 .addGap(18, 18, 18)
799                 .addComponent(importarDirectorio)
800                 .addContainerGap(240, Short.MAX_VALUE))
801         );
802 
803         jTabbedPane1.addTab("Importar/Exportar",
    imageBackgroundPanel7);
804 
805         barraProgreso.setCursor(new java.awt.Cursor(java.awt.
    Cursor.WAIT_CURSOR));
806 
807         mensajeEstado.setBackground(java.awt.Color.orange);
808         mensajeEstado.setFont(new java.awt.Font("URW Gothic L",
     3, 16)); // NOI18N
809         mensajeEstado.setForeground(java.awt.Color.orange);
810 
811         jLabel18.setIcon(new javax.swing.ImageIcon("/home/
    antonio/Dropbox/4° Ingenieria Informática/TFG/Aplicación/
    ejemplos/procesado tweets AB_opt.png")); // NOI18N
812 
813         javax.swing.GroupLayout imageBackgroundPanel1Layout =
    new javax.swing.GroupLayout(imageBackgroundPanel1);
814         imageBackgroundPanel1.setLayout(
    imageBackgroundPanel1Layout);
```

```
815         imageBackgroundPanel1Layout.setHorizontalGroup(
816             imageBackgroundPanel1Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
817             .addGroup(imageBackgroundPanel1Layout.
    createSequentialGroup()
818                 .addContainerGap()
819                 .addGroup(imageBackgroundPanel1Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
820                     .addGroup(imageBackgroundPanel1Layout.
    createSequentialGroup()
821                         .addComponent(jTabbedPane1, javax.swing
    .GroupLayout.PREFERRED_SIZE, 2097, javax.swing.GroupLayout.
    PREFERRED_SIZE)
822                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED, javax.swing.
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
823                         .addComponent(imageBackgroundPanel2,
    javax.swing.GroupLayout.PREFERRED_SIZE, 98, javax.swing.
    GroupLayout.PREFERRED_SIZE)
824                         .addGap(465, 465, 465))
825                     .addGroup(imageBackgroundPanel1Layout.
    createSequentialGroup()
826                         .addGap(202, 202, 202)
827                         .addComponent(jLabel18)
828                         .addGap(232, 232, 232)
829                         .addGroup(imageBackgroundPanel1Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    TRAILING)
830                             .addComponent(barraProgreso, javax.
    swing.GroupLayout.PREFERRED_SIZE, 305, javax.swing.
    GroupLayout.PREFERRED_SIZE)
831                             .addComponent(mensajeEstado, javax.
    swing.GroupLayout.PREFERRED_SIZE, 696, javax.swing.
```

```
    GroupLayout.PREFERRED_SIZE))
832                         .addContainerGap(javax.swing.
    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
833         );
834         imageBackgroundPanel1Layout.setVerticalGroup(
835             imageBackgroundPanel1Layout.createParallelGroup(
    javax.swing.GroupLayout.Alignment.LEADING)
836             .addGroup(imageBackgroundPanel1Layout.
    createSequentialGroup()
837                 .addGroup(imageBackgroundPanel1Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING, false)
838                     .addGroup(imageBackgroundPanel1Layout.
    createSequentialGroup()
839                         .addContainerGap()
840                         .addComponent(barraProgreso, javax.
    swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.
    GroupLayout.PREFERRED_SIZE)
841                         .addPreferredGap(javax.swing.
    LayoutStyle.ComponentPlacement.RELATED)
842                         .addComponent(mensajeEstado, javax.
    swing.GroupLayout.PREFERRED_SIZE, 30, javax.swing.
    GroupLayout.PREFERRED_SIZE))
843                     .addComponent(jLabel18, javax.swing.
    GroupLayout.PREFERRED_SIZE, 80, Short.MAX_VALUE))
844                 .addPreferredGap(javax.swing.LayoutStyle.
    ComponentPlacement.RELATED)
845                 .addGroup(imageBackgroundPanel1Layout.
    createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
846                     .addGroup(imageBackgroundPanel1Layout.
    createSequentialGroup()
847                         .addGap(0, 0, Short.MAX_VALUE)
```

```
848                         .addComponent(imageBackgroundPanel2,
    javax.swing.GroupLayout.PREFERRED_SIZE, 86, javax.swing.
    GroupLayout.PREFERRED_SIZE))
849                     .addComponent(jTabbedPane1)))
850         );
851 
852         javax.swing.GroupLayout layout = new javax.swing.
    GroupLayout(getContentPane());
853         getContentPane().setLayout(layout);
854         layout.setHorizontalGroup(
855             layout.createParallelGroup(javax.swing.GroupLayout.
    Alignment.LEADING)
856             .addGroup(javax.swing.GroupLayout.Alignment.
    TRAILING, layout.createSequentialGroup()
857                 .addContainerGap(javax.swing.GroupLayout.
    DEFAULT_SIZE, Short.MAX_VALUE)
858                 .addComponent(imageBackgroundPanel1, javax.
    swing.GroupLayout.PREFERRED_SIZE, 2108, javax.swing.
    GroupLayout.PREFERRED_SIZE)
859                 .addGap(0, 0, 0))
860         );
861         layout.setVerticalGroup(
862             layout.createParallelGroup(javax.swing.GroupLayout.
    Alignment.LEADING)
863             .addGroup(layout.createSequentialGroup()
864                 .addComponent(imageBackgroundPanel1, javax.
    swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
    DEFAULT_SIZE, Short.MAX_VALUE)
865                 .addContainerGap())
866         );
867 
868         pack();
869     }// </editor-fold>//GEN-END:initComponents
870 
```

```
871     private void importarActionPerformed(java.awt.event.
    ActionEvent evt) {//GEN-FIRST:event_importarActionPerformed
872 
873         String ruta = txtRuta2.getText();
874 
875         String backus = "";
876 
877         if(ruta.trim().length()!=0){
878             importar.setEnabled(false);
879             try{
880 
881                 ProcessBuilder pBuilder = new ProcessBuilder("
    mysql", "-u" + DB.getInstance().getUsername(), "-p" + DB.
    getInstance().getPassword(), DB.getInstance().getDbName());
882                 pBuilder.redirectInput(ProcessBuilder.Redirect.
    from(new File(ruta)));
883                 Process process = pBuilder.start();
884                 int exitValue = process.waitFor();
885                 System.out.println(exitValue);
886                 JOptionPane.showMessageDialog(null, "Importado
    Correctamente");
887 
888             }catch(HeadlessException | IOException ex){
889                 JOptionPane.showMessageDialog(null, ex.
    getMessage());
890             } catch (InterruptedException ex) {
891                 Logger.getLogger(Home.class.getName()).log(
    Level.SEVERE, null, ex);
892             }
893             importar.setEnabled(true);
894         }
895 
896     }//GEN-LAST:event_importarActionPerformed
897 
```

```
898     private void almacenamientoImportarActionPerformed(java.awt
    .event.ActionEvent evt) {//GEN-FIRST:
    event_almacenamientoImportarActionPerformed
899 
900         JFileChooser dlg= new JFileChooser();
901         FileNameExtensionFilter fil = new
    FileNameExtensionFilter("SQL","sql");
902         dlg.setFileFilter(fil);
903         int option = dlg.showOpenDialog(null);
904         if(option == JFileChooser.APPROVE_OPTION){
905             String ruta = dlg.getSelectedFile().getPath();
906             txtRuta2.setText(ruta);
907         }
908 
909     }//GEN-LAST:event_almacenamientoImportarActionPerformed
910 
911     private void almacenamientoExportarActionPerformed(java.awt
    .event.ActionEvent evt) {//GEN-FIRST:
    event_almacenamientoExportarActionPerformed
912 
913         JFileChooser dlg= new JFileChooser();
914         int option = dlg.showSaveDialog(this);
915         if(option == JFileChooser.APPROVE_OPTION){
916             File f= dlg.getSelectedFile();
917             txtRuta1.setText(f.toString());
918         }
919     }//GEN-LAST:event_almacenamientoExportarActionPerformed
920 
921     private void exportarActionPerformed(java.awt.event.
    ActionEvent evt) {//GEN-FIRST:event_exportarActionPerformed
922 
923         String ruta = txtRuta1.getText();
924         String backus = "";
925 
```

```
926         exportar.setEnabled(false);
927 
928         if(ruta.trim().length()!=0){
929 
930             try{
931                 backus= "mysqldump --opt -u"+DB.getInstance().
    getUsername()+" -p"+DB.getInstance().getPassword()+" -B "+DB
    .getInstance().getDbName() +" -r "+ruta;
932                 Runtime rt = Runtime.getRuntime();
933                 Process process = rt.exec(backus);
934                 process.waitFor();
935                 JOptionPane.showMessageDialog(null, "Exportado
    Correctamente");
936 
937             }catch(Exception ex){
938                 JOptionPane.showMessageDialog(null, ex.
    getMessage());
939             }
940         }
941         exportar.setEnabled(true);
942      }//GEN-LAST:event_exportarActionPerformed
943 
944      private void importarDirectorioActionPerformed(java.awt.
    event.ActionEvent evt) {//GEN-FIRST:
    event_importarDirectorioActionPerformed
945 
946         JFileChooser chooser = new JFileChooser();
947         chooser.setCurrentDirectory(new java.io.File("."));
948         chooser.setDialogTitle("Seleccione la carpeta que
    contiene los archivos .xml");
949         chooser.setFileSelectionMode(JFileChooser.
    DIRECTORIES_ONLY);
950         chooser.setAcceptAllFileFilterUsed(false);
951 
```

```
952         if (chooser.showOpenDialog(this) == JFileChooser.
    APPROVE_OPTION) {
953             System.out.println("getCurrentDirectory(): "
954                 +  chooser.getCurrentDirectory());
955             System.out.println("getSelectedFile() : "
956                 +  chooser.getSelectedFile());
957 
958             importarDatosDirectorio(chooser.getSelectedFile().
    toString());
959 
960         }
961      }//GEN-LAST:event_importarDirectorioActionPerformed
962 
963      private void GenerarPDF1ActionPerformed(java.awt.event.
    ActionEvent evt) {//GEN-FIRST:
    event_GenerarPDF1ActionPerformed
964 
965         String ruta=txtRuta3.getText();
966 
967 
968         try{
969             FileOutputStream archivo = new FileOutputStream(
    ruta);
970             Document doc = new Document(PageSize.A4.rotate());
971             PdfWriter writer = PdfWriter.getInstance(doc,
    archivo);
972 
973             doc.open();
974             PdfContentByte cb = writer.getDirectContent();
975             doc.add(new Paragraph(new SimpleDateFormat("dd/MM/
    YYYY HH:mm").format(new Date())));
976             if(this.chart != null)
977             {
978 
```

```
979                 Image imgChart = Image.getInstance(this.chart.
    createBufferedImage(1000, 400).getScaledInstance(700, 280,
    java.awt.Image.SCALE_SMOOTH), null);
980                 doc.add(imgChart);
981             }
982 
983             doc.close();
984 
985             JOptionPane.showMessageDialog(null, "PDF creado
    correctamente");
986 
987         } catch (DocumentException | HeadlessException e){
988             System.out.println("error:"+e);
989         } catch (FileNotFoundException ex) {
990             Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
991         } catch (IOException ex) {
992             Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
993         }
994      }//GEN-LAST:event_GenerarPDF1ActionPerformed
995 
996      private void SeleccionarRutaAlmacenamiento1ActionPerformed(
    java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_SeleccionarRutaAlmacenamiento1ActionPerformed
997 
998         JFileChooser dlg= new JFileChooser();
999         int option = dlg.showSaveDialog(this);
1000        if(option == JFileChooser.APPROVE_OPTION){
1001            File f= dlg.getSelectedFile();
1002            txtRuta3.setText(f.toString());
1003        }
1004     }//GEN-LAST:
    event_SeleccionarRutaAlmacenamiento1ActionPerformed
```

```
1005
1006     private void OpcionesInformes1ActionPerformed(java.awt.
    event.ActionEvent evt) {//GEN-FIRST:
    event_OpcionesInformes1ActionPerformed
1007
1008         switch(OpcionesInformes1.getSelectedIndex()){
1009
1010             case 1:
1011             System.out.println("Primera opción");
1012             GenerarInformeTotalRetweets();//
    convertirIndiceANombreCuenta(CuerposDeSeguridad.
    getSelectedIndex()));
1013         break;
1014         case 2:
1015         System.out.println("Segunda opción");
1016         generarInformeTotalFavoritos();//
    convertirIndiceANombreCuenta(CuerposDeSeguridad.
    getSelectedIndex()));
1017         break;
1018         case 3:
1019         System.out.println("Tercera opción");
1020         generarInformeTweetConMasRetweets();//
    convertirIndiceANombreCuenta(CuerposDeSeguridad.
    getSelectedIndex()));
1021         break;
1022         case 4:
1023         System.out.println("Cuarta opción");
1024         generarInformeTweetConMasFavoritos();//
    convertirIndiceANombreCuenta(CuerposDeSeguridad.
    getSelectedIndex()));
1025         break;
1026         case 5:
1027         System.out.println("Quinta opción");
```

```
1028        generarInformeTotalSeguidoresCuentas();//
    convertirIndiceANombreCuenta(CuerposDeSeguridad.
    getSelectedIndex()));
1029        break;
1030        }
1031        OpcionesInformes1.setSelectedIndex(0);
1032     }//GEN-LAST:event_OpcionesInformes1ActionPerformed
1033
1034     private void GenerarPDFActionPerformed(java.awt.event.
    ActionEvent evt) {//GEN-FIRST:
    event_GenerarPDFActionPerformed
1035
1036        String ruta=txtRuta.getText();
1037
1038
1039        try{
1040            FileOutputStream archivo = new FileOutputStream(
    ruta);
1041            Document doc = new Document(PageSize.A4.rotate());
1042            PdfWriter writer = PdfWriter.getInstance(doc,
    archivo);
1043
1044            doc.open();
1045            PdfContentByte cb = writer.getDirectContent();
1046            doc.add(new Paragraph(new SimpleDateFormat("dd/MM/
    YYYY HH:mm").format(new Date())));
1047            if(this.chart != null)
1048            {
1049
1050                Image imgChart = Image.getInstance(this.chart.
    createBufferedImage(1000, 400).getScaledInstance(700, 280,
    java.awt.Image.SCALE_SMOOTH), null);
1051                doc.add(imgChart);
1052            }
```

```
1053
1054            doc.close();
1055
1056            JOptionPane.showMessageDialog(null, "PDF creado
    correctamente");
1057
1058        } catch (DocumentException | HeadlessException e){
1059            System.out.println("error:"+e);
1060        } catch (FileNotFoundException ex) {
1061            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1062        } catch (IOException ex) {
1063            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1064        }
1065
1066     }//GEN-LAST:event_GenerarPDFActionPerformed
1067
1068     private void SeleccionarRutaAlmacenamientoActionPerformed(
    java.awt.event.ActionEvent evt) {//GEN-FIRST:
    event_SeleccionarRutaAlmacenamientoActionPerformed
1069
1070        JFileChooser dlg= new JFileChooser();
1071        int option = dlg.showSaveDialog(this);
1072        if(option == JFileChooser.APPROVE_OPTION){
1073            File f= dlg.getSelectedFile();
1074            txtRuta.setText(f.toString());
1075        }
1076
1077     }//GEN-LAST:
    event_SeleccionarRutaAlmacenamientoActionPerformed
1078
1079     private void OpcionesInformesActionPerformed(java.awt.event
    .ActionEvent evt) {//GEN-FIRST:
```

```
    event_OpcionesInformesActionPerformed
1080
1081        switch(OpcionesInformes.getSelectedIndex()){
1082
1083        case 1:
1084        System.out.println("Primera opción");
1085        generarInformeRetweets(CuerposDeSeguridad.
    getSelectedItem().toString());//convertirIndiceANombreCuenta
    (CuerposDeSeguridad.getSelectedIndex()));
1086        break;
1087        case 2:
1088        System.out.println("Segunda opción");
1089        generarInformeFavoritos(CuerposDeSeguridad.
    getSelectedItem().toString());//convertirIndiceANombreCuenta
    (CuerposDeSeguridad.getSelectedIndex()));
1090        break;
1091        case 3:
1092        System.out.println("Tercera opción");
1093        generarInformeIdioma(CuerposDeSeguridad.getSelectedItem
    ().toString());//convertirIndiceANombreCuenta(
    CuerposDeSeguridad.getSelectedIndex()));
1094        break;
1095        case 4:
1096        System.out.println("Cuarta opción");
1097        generarInformeTotalSeguidores(CuerposDeSeguridad.
    getSelectedItem().toString());//convertirIndiceANombreCuenta
    (CuerposDeSeguridad.getSelectedIndex()));
1098        break;
1099        case 5:
1100        System.out.println("Quinta opción");
1101        generarInformeTweetsEmitidosPeriodo(CuerposDeSeguridad.
    getSelectedItem().toString());//convertirIndiceANombreCuenta
    (CuerposDeSeguridad.getSelectedIndex()));
1102        break;
```

```
1103        case 6:
1104        System.out.println("Sexta opción");
1105        obtenerTweetsConMasRetweets(CuerposDeSeguridad.
    getSelectedItem().toString());//convertirIndiceANombreCuenta
    (CuerposDeSeguridad.getSelectedIndex()));
1106        String texto = obtenerTweetsConMasRetweets(
    CuerposDeSeguridad.getSelectedItem().toString());//
    convertirIndiceANombreCuenta(CuerposDeSeguridad.
    getSelectedIndex()));
1107
1108        Informe informe = new Informe();
1109        informe.setText(texto);
1110        informe.setVisible(true);
1111        generarInformeRetweets(CuerposDeSeguridad.
    getSelectedItem().toString());
1112        break;
1113        case 7:
1114        System.out.println("Séptima opción");
1115        obtenerTweetsConMasFavoritos(CuerposDeSeguridad.
    getSelectedItem().toString());//convertirIndiceANombreCuenta
    (CuerposDeSeguridad.getSelectedIndex()));
1116        String texto2 = obtenerTweetsConMasFavoritos(
    CuerposDeSeguridad.getSelectedItem().toString());//
    convertirIndiceANombreCuenta(CuerposDeSeguridad.
    getSelectedIndex()));
1117
1118        Informe informe2 = new Informe();
1119        informe2.setText(texto2);
1120        informe2.setVisible(true);
1121        generarInformeFavoritos(CuerposDeSeguridad.
    getSelectedItem().toString());
1122        break;
1123        }
1124
```

```
1125         OpcionesInformes.setSelectedIndex(0);
1126 
1127     }//GEN-LAST:event_OpcionesInformesActionPerformed
1128 
1129     private void CuerposDeSeguridadActionPerformed(java.awt.
    event.ActionEvent evt) {//GEN-FIRST:
    event_CuerposDeSeguridadActionPerformed
1130 
1131         switch(CuerposDeSeguridad.getSelectedIndex()){
1132             case 0:
1133             System.out.println("Primera opción");
1134             break;
1135             case 1:
1136             System.out.println("Segunda opción");
1137             break;
1138             case 2:
1139             System.out.println("Tercera opción");
1140             break;
1141             case 3:
1142             System.out.println("Cuarta opción");
1143             break;
1144 
1145         }
1146 
1147     }//GEN-LAST:event_CuerposDeSeguridadActionPerformed
1148 
1149     private void buscarHashtagsActionPerformed(java.awt.event.
    ActionEvent evt) {//GEN-FIRST:
    event_buscarHashtagsActionPerformed
1150 
1151         generarInformeTotalRepeticionesHashtags(
    CuerposDeSeguridad.getSelectedItem().toString());
1152 
1153     }//GEN-LAST:event_buscarHashtagsActionPerformed
```

```
1154
1155     private void ImportarDatosActionPerformed(java.awt.event.
         ActionEvent evt) {//GEN-FIRST:
         event_ImportarDatosActionPerformed
1156
1157         importarDatosCuentaEspecifica(nombreCuenta.getText());
1158         actualizarCuentas1();
1159
1160     }//GEN-LAST:event_ImportarDatosActionPerformed
1161
1162     private void nombreCuentaActionPerformed(java.awt.event.
         ActionEvent evt) {//GEN-FIRST:
         event_nombreCuentaActionPerformed
1163
1164     }//GEN-LAST:event_nombreCuentaActionPerformed
1165
1166     private void PoliciaLocalDeMadridActionPerformed(java.awt.
         event.ActionEvent evt) {//GEN-FIRST:
         event_PoliciaLocalDeMadridActionPerformed
1167
1168         importarDatosCuentaEspecifica("policiademadrid");
1169     }//GEN-LAST:event_PoliciaLocalDeMadridActionPerformed
1170
1171     private void MossosDeEsquadraActionPerformed(java.awt.event
         .ActionEvent evt) {//GEN-FIRST:
         event_MossosDeEsquadraActionPerformed
1172
1173         importarDatosCuentaEspecifica("mossos");
1174     }//GEN-LAST:event_MossosDeEsquadraActionPerformed
1175
1176     private void DescargaGuardiaCivilActionPerformed(java.awt.
         event.ActionEvent evt) {//GEN-FIRST:
         event_DescargaGuardiaCivilActionPerformed
1177
```

```
1178            importarDatosCuentaEspecifica("guardiacivil");
1179        }//GEN-LAST:event_DescargaGuardiaCivilActionPerformed
1180
1181        private void PoliciaNacionalActionPerformed(java.awt.event.
        ActionEvent evt) {//GEN-FIRST:
        event_PoliciaNacionalActionPerformed
1182
1183            importarDatosCuentaEspecifica("policia");
1184        }//GEN-LAST:event_PoliciaNacionalActionPerformed
1185
1186        private void CuerposDeSeguridad1ActionPerformed(java.awt.
        event.ActionEvent evt) {//GEN-FIRST:
        event_CuerposDeSeguridad1ActionPerformed
1187
1188
1189        }//GEN-LAST:event_CuerposDeSeguridad1ActionPerformed
1190
1191        private void eliminarCuentaActionPerformed(java.awt.event.
        ActionEvent evt) {//GEN-FIRST:
        event_eliminarCuentaActionPerformed
1192
1193            eliminarCuenta(CuerposDeSeguridad1.getSelectedItem().
        toString());
1194
1195            actualizarCuentas();
1196            actualizarCuentas1();
1197        }//GEN-LAST:event_eliminarCuentaActionPerformed
1198
1199
1200        private void eliminarCuenta(String cuenta){
1201
1202            ResultSet rs;
1203            try {
1204
```

```
1205            DB.getInstance().query(String.format("Delete from
    tweets where accountName='%s';",cuenta));
1206            rs = DB.getInstance().getResultSet();
1207            rs.next();
1208        } catch (SQLException ex) {
1209            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1210        }
1211     }
1212
1213     private void importarDatosCuentaEspecifica(String cuenta){
1214
1215        String date = new SimpleDateFormat("YYYY_MM_dd_HH_mm_ss
    ").format(new Date());
1216        ImportWorker worker = new ImportWorker(cuenta, String.
    format("%s/TweetsDescargados/%s_%s",System.getProperty("user
    .home"),cuenta,date));
1217        worker.setProgressBar(barraProgreso);
1218        worker.setStatusMessage(mensajeEstado);
1219        worker.setDoneFunction((t) -> { actualizarCuentas();
    actualizarCuentas1(); return null; });
1220
1221        worker.execute();
1222     }
1223
1224     private void importarDatosDirectorio(String directorio)
1225     {
1226        ImportWorker worker = new ImportWorker("", directorio);
1227        worker.setProgressBar(barraProgreso);
1228        worker.setStatusMessage(mensajeEstado);
1229        worker.execute();
1230     }
1231
1232     private void generarInformeIdioma(String cuenta){
```

```
1233
1234        DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1235
1236        ResultSet rs;
1237        try {
1238            DB.getInstance().queryData(String.format("select
    count(lang) as español from tweets where lang=\"es\" and
    accountName='%s';",cuenta));
1239            rs = DB.getInstance().getResultSet();
1240            rs.next();
1241            dataset.addValue(rs.getInt("español"), "Titulo", "
    Español");
1242
1243            DB.getInstance().queryData(String.format("select
    count(lang) as ingles from tweets where lang=\"en\" and
    accountName='%s';",cuenta));
1244            rs = DB.getInstance().getResultSet();
1245            rs.next();
1246            dataset.addValue(rs.getInt("ingles"), "Titulo", "
    Inglés");
1247
1248            DB.getInstance().queryData(String.format("select
    count(lang) as frances from tweets where lang=\"fr\" and
    accountName='%s';",cuenta));
1249            rs = DB.getInstance().getResultSet();
1250            rs.next();
1251            dataset.addValue(rs.getInt("frances"), "Titulo", "
    Francés");
1252
1253
1254            DB.getInstance().queryData(String.format("select
    count(lang) as portugues from tweets where lang=\"pt\"and
    accountName='%s';",cuenta));
```

```
1255            rs = DB.getInstance().getResultSet();
1256            rs.next();
1257            dataset.addValue(rs.getInt("portugues"), "Titulo",
    "Portugués");
1258
1259        } catch (SQLException ex) {
1260            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1261        }
1262
1263
1264        this.chart = ChartFactory.createBarChart("Idiomas
    Usados en los Tweets", "Idioma", cuenta, dataset,
    PlotOrientation.HORIZONTAL, false, false, false);
1265        CategoryPlot plot = this.chart.getCategoryPlot();
1266        plot.setRangeGridlinePaint(Color.BLACK);
1267
1268        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1269
1270        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1271        renderer.setBaseItemLabelsVisible(true);
1272        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.CENTER,
1273                TextAnchor.CENTER);
1274        renderer.setBasePositiveItemLabelPosition(position);
1275
1276
1277        ChartPanel chartPanel = new ChartPanel(this.chart);
1278
1279        chartContainer.removeAll();
1280        chartContainer.add(chartPanel, BorderLayout.CENTER);
1281        chartContainer.validate();
```

```
1282     }
1283 
1284     private void generarInformeRetweets(String cuenta){
1285 
1286         DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1287 
1288         ResultSet rs;
1289         try {
1290             DB.getInstance().queryData(String.format("select
    retweetCount from tweets where accountName= '%s' order by
    retweetCount DESC limit 10",cuenta));
1291             rs = DB.getInstance().getResultSet();
1292 
1293             int index = 1;
1294 
1295             while (rs.next())
1296             {
1297                 dataset.addValue(rs.getInt("retweetCount"), ""
    + index++, "");
1298             }
1299 
1300 
1301         } catch (SQLException ex) {
1302             Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1303         }
1304 
1305 
1306         this.chart = ChartFactory.createBarChart("Lista de
    tweets con mayor número de Retweets", "Tweets", cuenta,
    dataset, PlotOrientation.HORIZONTAL, false, false, false);
1307         CategoryPlot plot = chart.getCategoryPlot();
1308         plot.setRangeGridlinePaint(Color.BLACK);
```

```
1309
1310        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1311
1312        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1313        renderer.setBaseItemLabelsVisible(true);
1314        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.CENTER,
1315                TextAnchor.CENTER);
1316        renderer.setBasePositiveItemLabelPosition(position);
1317
1318
1319        ChartPanel chartPanel = new ChartPanel(chart);
1320
1321        chartContainer.removeAll();
1322        chartContainer.add(chartPanel, BorderLayout.CENTER);
1323        chartContainer.validate();
1324     }
1325
1326
1327     private void generarInformeFavoritos(String cuenta){
1328
1329          DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1330
1331        ResultSet rs;
1332        try {
1333
1334            DB.getInstance().queryData(String.format("select
    favoriteCount from tweets where accountName= '%s' order by
    favoriteCount DESC limit 10;",cuenta));
1335            rs = DB.getInstance().getResultSet();
1336
```

```
1337            int index = 1;
1338
1339            while (rs.next())
1340            {
1341                dataset.addValue(rs.getInt("favoriteCount"), ""
     + index++, "");
1342            }
1343
1344
1345        } catch (SQLException ex) {
1346            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1347        }
1348
1349
1350        this.chart = ChartFactory.createBarChart("Lista de
    tweets con mayor número de Favoritos", "Tweets", cuenta,
    dataset, PlotOrientation.HORIZONTAL, false, false, false);
1351        CategoryPlot plot = chart.getCategoryPlot();
1352        plot.setRangeGridlinePaint(Color.BLACK);
1353
1354        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1355
1356        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1357        renderer.setBaseItemLabelsVisible(true);
1358        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.CENTER,
1359                TextAnchor.CENTER);
1360        renderer.setBasePositiveItemLabelPosition(position);
1361
1362
1363        ChartPanel chartPanel = new ChartPanel(chart);
```

```
1364
1365            chartContainer.removeAll();
1366            chartContainer.add(chartPanel, BorderLayout.CENTER);
1367            chartContainer.validate();
1368        }
1369
1370
1371
1372
1373
1374        private void generarInformeTotalRepeticionesHashtags(String
        cuenta){
1375            DefaultCategoryDataset dataset = new
        DefaultCategoryDataset();
1376
1377           String nombre = (buscarHashtag.getText());
1378           int index = 1;
1379            ResultSet rs;
1380            try {
1381                DB.getInstance().queryData(String.format("select
        count(hashtagEntities) as hashtag from tweets where
        hashtagEntities LIKE '%s' AND accountName='%s';",'%'+nombre+
        '%',cuenta));
1382
1383                rs = DB.getInstance().getResultSet();
1384
1385
1386
1387                while (rs.next())
1388                {
1389                    dataset.addValue(rs.getInt("hashtag"), "" +
        index++, "");
1390                }
1391
```

```
1392
1393        } catch (SQLException ex) {
1394            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1395        }
1396
1397
1398        this.chart = ChartFactory.createBarChart("Total de
    veces que aparece ese hashtag", "", "Número total de
    repeticiones de ese hashtag", dataset, PlotOrientation.
    HORIZONTAL, false, false, false);
1399        CategoryPlot plot = chart.getCategoryPlot();
1400        plot.setRangeGridlinePaint(Color.BLACK);
1401
1402        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1403
1404        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1405        renderer.setBaseItemLabelsVisible(true);
1406        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.CENTER,
1407                TextAnchor.CENTER);
1408        renderer.setBasePositiveItemLabelPosition(position);
1409
1410
1411        ChartPanel chartPanel = new ChartPanel(chart);
1412
1413        chartContainer.removeAll();
1414        if(dataset.getValue(0, 0).intValue() > 0){
1415            chartContainer.add(chartPanel, BorderLayout.CENTER)
    ;
1416        }
1417        else{
```

```
1418            JOptionPane.showMessageDialog(null, "No existen
    datos para el informe seleccionado.", "", JOptionPane.
    WARNING_MESSAGE);
1419        }
1420        System.out.println(dataset.getRowCount());
1421        System.out.println("select count(hashtagEntities) as
    hashtag from tweets where hashtagEntities LIKE '%#seguridad
    %';");
1422        chartContainer.validate();
1423     }
1424
1425
1426
1427
1428     private void generarInformeTweetsEmitidosPeriodo(String
    cuenta){
1429
1430        DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1431        Calendar c1 = jDateFecha1.getCalendar();
1432        Calendar c2 = jDateFecha2.getCalendar();
1433
1434
1435        c2.add(Calendar.DATE, 1);
1436
1437        String f=new SimpleDateFormat("YYYYMMdd").format(c1.
    getTime());
1438        String f2=new SimpleDateFormat("YYYYMMdd").format(c2.
    getTime());
1439
1440        int index = 1;
1441        ResultSet rs;
1442        try {
```

```
1443            DB.getInstance().queryData(String.format("select
    count(id) as TotalTweets from tweets where accountName='%s'
    and createdAt>='%s' AND createdAt<'%s';",cuenta,f,f2));
1444            rs = DB.getInstance().getResultSet();
1445
1446
1447
1448            while (rs.next())
1449            {
1450                dataset.addValue(rs.getInt("TotalTweets"), "" +
     index++, "");
1451            }
1452
1453
1454        } catch (SQLException ex) {
1455            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1456        }
1457
1458
1459        this.chart = ChartFactory.createBarChart("Total de
    tweets en el periodo de tiempo indicado", "Total de tweets "
    ,cuenta , dataset, PlotOrientation.HORIZONTAL, false, false,
     false);
1460        CategoryPlot plot = chart.getCategoryPlot();
1461        plot.setRangeGridlinePaint(Color.BLACK);
1462
1463        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1464
1465        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1466        renderer.setBaseItemLabelsVisible(true);
```

```
1467        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.CENTER,
1468                TextAnchor.CENTER);
1469        renderer.setBasePositiveItemLabelPosition(position);
1470
1471
1472        ChartPanel chartPanel = new ChartPanel(chart);
1473
1474        chartContainer.removeAll();
1475         if(dataset.getValue(0, 0).intValue()>0){
1476            chartContainer.add(chartPanel, BorderLayout.CENTER)
    ;
1477        }
1478        else{
1479            JOptionPane.showMessageDialog(null, "No existen
    datos para las fechas seleccionadas. Modifique las fechas de
     Inicio y Fin", "", JOptionPane.WARNING_MESSAGE);
1480        }
1481
1482        chartContainer.validate();
1483
1484     }
1485
1486     private void generarInformeTotalSeguidoresCuentas(){
1487
1488        DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1489
1490        ResultSet rs;
1491        try {
1492            DB.getInstance().queryData(String.format("select
    max(followersCount) as policia from tweets where accountName
    ='policia';"));
1493            rs = DB.getInstance().getResultSet();
```

```
1494            rs.next();
1495            dataset.addValue(rs.getInt("policia"), "prueba", "
    Policia Nacional");
1496
1497            DB.getInstance().queryData(String.format("select
    max(followersCount) as guardiacivil from tweets where
    accountName='guardiacivil';"));
1498            rs = DB.getInstance().getResultSet();
1499            rs.next();
1500            dataset.addValue(rs.getInt("guardiacivil"), "
    prueba1", "Guardia civil");
1501
1502            DB.getInstance().queryData(String.format("select
    max(followersCount) as policiademadrid from tweets where
    accountName='policiademadrid';"));
1503            rs = DB.getInstance().getResultSet();
1504            rs.next();
1505            dataset.addValue(rs.getInt("policiademadrid"), "
    prueba2", "Policia Local de Madrid");
1506
1507            DB.getInstance().queryData(String.format("select
    max(followersCount) as mossos from tweets where accountName
    ='mossos';"));
1508            rs = DB.getInstance().getResultSet();
1509            rs.next();
1510            dataset.addValue(rs.getInt("mossos"), "prueba3", "
    Mossos d'Esquadra");
1511
1512        } catch (SQLException ex) {
1513            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1514        }
1515
1516
```

```
1517        this.chart = ChartFactory.createBarChart("Número total
    de seguidores de cada cuenta", "", "", dataset,
    PlotOrientation.HORIZONTAL, false, false, false);
1518        CategoryPlot plot = chart.getCategoryPlot();
1519        plot.setRangeGridlinePaint(Color.BLACK);
1520
1521        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1522
1523        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1524        renderer.setBaseItemLabelsVisible(true);
1525        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.OUTSIDE5,
1526                TextAnchor.TOP_CENTER);
1527        renderer.setBasePositiveItemLabelPosition(position);
1528
1529
1530        ChartPanel chartPanel = new ChartPanel(chart);
1531
1532        chartContainer1.removeAll();
1533        chartContainer1.add(chartPanel, BorderLayout.CENTER);
1534        chartContainer1.validate();
1535
1536     }
1537
1538     private void generarInformeTotalSeguidores(String cuenta){
1539
1540          DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1541
1542        ResultSet rs;
1543        try {
```

```
1544            DB.getInstance().queryData(String.format("select
    max(followersCount) as total from tweets where accountName
    ='%s';",cuenta));
1545            rs = DB.getInstance().getResultSet();
1546            rs.next();
1547            dataset.addValue(rs.getInt("total"), "Titulo", "
    total");
1548
1549
1550
1551        } catch (SQLException ex) {
1552            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1553        }
1554
1555
1556        this.chart = ChartFactory.createBarChart("Número de
    seguidores actuales de la cuenta", "", cuenta, dataset,
    PlotOrientation.HORIZONTAL, false, false, false);
1557        CategoryPlot plot = chart.getCategoryPlot();
1558        plot.setRangeGridlinePaint(Color.BLACK);
1559
1560        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1561
1562        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1563        renderer.setBaseItemLabelsVisible(true);
1564        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.CENTER,
1565                TextAnchor.CENTER);
1566        renderer.setBasePositiveItemLabelPosition(position);
1567
1568
```

```
1569        ChartPanel chartPanel = new ChartPanel(chart);
1570
1571        chartContainer.removeAll();
1572        chartContainer.add(chartPanel, BorderLayout.CENTER);
1573        chartContainer.validate();
1574     }
1575
1576
1577
1578
1579      private void GenerarInformeTotalRetweets(){
1580        DefaultCategoryDataset dataset = new
      DefaultCategoryDataset();
1581        Calendar c3 = jDateFecha3.getCalendar();
1582        Calendar c4 = jDateFecha4.getCalendar();
1583
1584
1585        c4.add(Calendar.DATE, 1);
1586
1587        String f3=new SimpleDateFormat("YYYYMMdd").format(c3.
      getTime());
1588        String f4=new SimpleDateFormat("YYYYMMdd").format(c4.
      getTime());
1589
1590        ResultSet rs;
1591        try {
1592            DB.getInstance().queryData(String.format("select
      sum(retweetCount) as policia from tweets where accountName
      =\"policia\" and createdAt>'%s' AND createdAt<'%s';",f3,f4))
      ;
1593            rs = DB.getInstance().getResultSet();
1594            rs.next();
1595            dataset.addValue(rs.getInt("policia"), "prueba", "
      Policia Nacional");
```

```
1596
1597            DB.getInstance().queryData(String.format("select
    sum(retweetCount) as guardiacivil from tweets where
    accountName=\"guardiacivil\" and createdAt>'%s' AND
    createdAt<'%s';",f3,f4));
1598            rs = DB.getInstance().getResultSet();
1599            rs.next();
1600            dataset.addValue(rs.getInt("guardiacivil"), "
    prueba1", "Guardia civil");
1601
1602            DB.getInstance().queryData(String.format("select
    sum(retweetCount) as policiademadrid from tweets where
    accountName=\"policiademadrid\" and createdAt>'%s' AND
    createdAt<'%s';",f3,f4));
1603            rs = DB.getInstance().getResultSet();
1604            rs.next();
1605            dataset.addValue(rs.getInt("policiademadrid"), "
    prueba2", "Policia Local de Madrid");
1606
1607            DB.getInstance().queryData(String.format("select
    sum(retweetCount) as mossos from tweets where accountName=\"
    mossos\" and createdAt>'%s' AND createdAt<'%s';",f3,f4));
1608            rs = DB.getInstance().getResultSet();
1609            rs.next();
1610            dataset.addValue(rs.getInt("mossos"), "prueba3", "
    Mossos d'Esquadra");
1611
1612        } catch (SQLException ex) {
1613            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1614        }
1615
1616
```

```
1617        this.chart = ChartFactory.createBarChart("Suma total de
     Retweets que tiene cada cuenta", "", "", dataset,
    PlotOrientation.HORIZONTAL, false, false, false);
1618        CategoryPlot plot = chart.getCategoryPlot();
1619        plot.setRangeGridlinePaint(Color.BLACK);
1620
1621        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1622
1623        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1624        renderer.setBaseItemLabelsVisible(true);
1625        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.OUTSIDE5,
1626                TextAnchor.TOP_CENTER);
1627        renderer.setBasePositiveItemLabelPosition(position);
1628
1629
1630        ChartPanel chartPanel = new ChartPanel(chart);
1631
1632        chartContainer1.removeAll();
1633            if(dataset.getValue(0, 0).intValue()>0){
1634                chartContainer1.add(chartPanel, BorderLayout.
    CENTER);
1635            }
1636            else{
1637                JOptionPane.showMessageDialog(null, "No existen
     datos para las fechas seleccionadas. Modifique las fechas
    de Inicio y Fin", "", JOptionPane.WARNING_MESSAGE);
1638            }
1639
1640        chartContainer1.validate();
1641     }
1642
```

```
1643     private void generarInformeTweetConMasRetweets(){
1644 
1645                 DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1646 
1647         ResultSet rs;
1648         try {
1649             DB.getInstance().queryData(String.format("select
    max(retweetCount) as policia from tweets where accountName
    =\"policia\";"));
1650             rs = DB.getInstance().getResultSet();
1651             rs.next();
1652             dataset.addValue(rs.getInt("policia"), "prueba", "
    Policia Nacional");
1653 
1654             DB.getInstance().queryData(String.format("select
    max(retweetCount) as guardiacivil from tweets where
    accountName=\"guardiacivil\";"));
1655             rs = DB.getInstance().getResultSet();
1656             rs.next();
1657             dataset.addValue(rs.getInt("guardiacivil"), "
    prueba1", "Guardia civil");
1658 
1659             DB.getInstance().queryData(String.format("select
    max(retweetCount) as policiademadrid from tweets where
    accountName=\"policiademadrid\";"));
1660             rs = DB.getInstance().getResultSet();
1661             rs.next();
1662             dataset.addValue(rs.getInt("policiademadrid"), "
    prueba2", "Policia Local de Madrid");
1663 
1664             DB.getInstance().queryData(String.format("select
    max(retweetCount) as mossos from tweets where accountName=\"
    mossos\";"));
```

```
1665            rs = DB.getInstance().getResultSet();
1666            rs.next();
1667            dataset.addValue(rs.getInt("mossos"), "prueba3", "
    Mossos d'Esquadra");
1668
1669        } catch (SQLException ex) {
1670            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1671        }
1672
1673
1674        this.chart = ChartFactory.createBarChart("Tweet con
    mayor número de Retweets que tiene cada cuenta", "", "",
    dataset, PlotOrientation.HORIZONTAL, false, false, false);
1675        CategoryPlot plot = chart.getCategoryPlot();
1676        plot.setRangeGridlinePaint(Color.BLACK);
1677
1678        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1679
1680        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1681        renderer.setBaseItemLabelsVisible(true);
1682        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.OUTSIDE5,
1683                TextAnchor.TOP_CENTER);
1684        renderer.setBasePositiveItemLabelPosition(position);
1685
1686
1687        ChartPanel chartPanel = new ChartPanel(chart);
1688
1689        chartContainer1.removeAll();
1690        chartContainer1.add(chartPanel, BorderLayout.CENTER);
1691        chartContainer1.validate();
```

```
1692     }
1693
1694
1695     private void generarInformeTweetConMasFavoritos(){
1696
1697         DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1698
1699         ResultSet rs;
1700         try {
1701             DB.getInstance().queryData(String.format("select
    max(favoriteCount) as policia from tweets where accountName
    =\"policia\";"));
1702             rs = DB.getInstance().getResultSet();
1703             rs.next();
1704             dataset.addValue(rs.getInt("policia"), "prueba", "
    Policia Nacional");
1705
1706             DB.getInstance().queryData(String.format("select
    max(favoriteCount) as guardiacivil from tweets where
    accountName=\"guardiacivil\";"));
1707             rs = DB.getInstance().getResultSet();
1708             rs.next();
1709             dataset.addValue(rs.getInt("guardiacivil"), "
    prueba1", "Guardia civil");
1710
1711             DB.getInstance().queryData(String.format("select
    max(favoriteCount) as policiademadrid from tweets where
    accountName=\"policiademadrid\";"));
1712             rs = DB.getInstance().getResultSet();
1713             rs.next();
1714             dataset.addValue(rs.getInt("policiademadrid"), "
    prueba2", "Policia Local de Madrid");
1715
```

```
1716            DB.getInstance().queryData(String.format("select
    max(favoriteCount) as mossos from tweets where accountName
    =\"mossos\";"));
1717            rs = DB.getInstance().getResultSet();
1718            rs.next();
1719            dataset.addValue(rs.getInt("mossos"), "prueba3", "
    Mossos d'Esquadra");
1720
1721        } catch (SQLException ex) {
1722            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1723        }
1724
1725
1726        this.chart = ChartFactory.createBarChart("Tweet con
    mayor número de Favoritos que tiene cada cuenta", "", "",
    dataset, PlotOrientation.HORIZONTAL, false, false, false);
1727        CategoryPlot plot = chart.getCategoryPlot();
1728        plot.setRangeGridlinePaint(Color.BLACK);
1729
1730
1731        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1732
1733        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1734        renderer.setBaseItemLabelsVisible(true);
1735        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.OUTSIDE5,
1736                TextAnchor.TOP_CENTER);
1737        renderer.setBasePositiveItemLabelPosition(position);
1738
1739
1740        ChartPanel chartPanel = new ChartPanel(chart);
```

```
1741
1742        chartContainer1.removeAll();
1743        chartContainer1.add(chartPanel, BorderLayout.CENTER);
1744        chartContainer1.validate();
1745    }
1746
1747    private void generarInformeTotalFavoritos(){
1748
1749        DefaultCategoryDataset dataset = new
    DefaultCategoryDataset();
1750        Calendar c3 = jDateFecha3.getCalendar();
1751        Calendar c4 = jDateFecha4.getCalendar();
1752
1753
1754        c4.add(Calendar.DATE, 1);
1755
1756        String f3=new SimpleDateFormat("YYYYMMdd").format(c3.
    getTime());
1757        String f4=new SimpleDateFormat("YYYYMMdd").format(c4.
    getTime());
1758
1759        ResultSet rs;
1760        try {
1761            DB.getInstance().queryData(String.format("select
    sum(favoriteCount) as policia from tweets where accountName
    =\"policia\" and createdAt>'%s' AND createdAt<'%s';",f3,f4))
    ;
1762            rs = DB.getInstance().getResultSet();
1763            rs.next();
1764            dataset.addValue(rs.getInt("policia"), "prueba", "
    Policia Nacional");
1765
1766            DB.getInstance().queryData(String.format("select
    sum(favoriteCount) as guardiacivil from tweets where
```

```
accountName=\"guardiacivil\" and createdAt>'%s' AND
createdAt<'%s';",f3,f4));
1767            rs = DB.getInstance().getResultSet();
1768            rs.next();
1769            dataset.addValue(rs.getInt("guardiacivil"), "
prueba1", "Guardia civil");
1770
1771            DB.getInstance().queryData(String.format("select
sum(favoriteCount) as policiademadrid from tweets where
accountName=\"policiademadrid\" and createdAt>'%s' AND
createdAt<'%s';",f3,f4));
1772            rs = DB.getInstance().getResultSet();
1773            rs.next();
1774            dataset.addValue(rs.getInt("policiademadrid"), "
prueba2", "Policia Local de Madrid");
1775
1776            DB.getInstance().queryData(String.format("select
sum(favoriteCount) as mossos from tweets where accountName
=\"mossos\" and createdAt>'%s' AND createdAt<'%s';",f3,f4));
1777            rs = DB.getInstance().getResultSet();
1778            rs.next();
1779            dataset.addValue(rs.getInt("mossos"), "prueba3", "
Mossos d'Esquadra");
1780
1781        } catch (SQLException ex) {
1782            Logger.getLogger(Home.class.getName()).log(Level.
SEVERE, null, ex);
1783        }
1784
1785
1786        this.chart = ChartFactory.createBarChart("Suma total de
 Favoritos que tiene cada cuenta", "", "", dataset,
PlotOrientation.HORIZONTAL, false, false, false);
1787        CategoryPlot plot = chart.getCategoryPlot();
```

```
1788        plot.setRangeGridlinePaint(Color.BLACK);
1789
1790
1791        CategoryItemRenderer renderer = ((CategoryPlot)chart.
    getPlot()).getRenderer();
1792
1793        renderer.setBaseItemLabelGenerator(new
    StandardCategoryItemLabelGenerator());
1794        renderer.setBaseItemLabelsVisible(true);
1795        ItemLabelPosition position = new ItemLabelPosition(
    ItemLabelAnchor.OUTSIDE5,
1796                TextAnchor.TOP_CENTER);
1797        renderer.setBasePositiveItemLabelPosition(position);
1798
1799
1800        ChartPanel chartPanel = new ChartPanel(chart);
1801
1802        chartContainer1.removeAll();
1803            if(dataset.getValue(0, 0).intValue()>0){
1804                chartContainer1.add(chartPanel, BorderLayout.
    CENTER);
1805            }
1806            else{
1807                JOptionPane.showMessageDialog(null, "No existen
     datos para las fechas seleccionadas. Modifique las fechas
    de Inicio y Fin", "", JOptionPane.WARNING_MESSAGE);
1808            }
1809        chartContainer1.validate();
1810     }
1811
1812     private String obtenerTweetsConMasRetweets(String cuenta)
1813     {
1814
1815        ResultSet rs;
```

```
1816        String informe = "";
1817        try {
1818            DB.getInstance().queryData(String.format("select
    text, createdAt, retweetCount from tweets where accountName=
     '%s' order by retweetCount DESC limit 10", cuenta));
1819            rs = DB.getInstance().getResultSet();
1820
1821            int index = 1;
1822
1823
1824            while (rs.next())
1825            {
1826                DateFormat format = new SimpleDateFormat("
    yyyyMMddhhmmss", Locale.ENGLISH);
1827
1828                  Date date=new Date();
1829                    try {
1830                        date = format.parse(rs.getString("
    createdAt"));
1831                    } catch (ParseException ex) {
1832                        Logger.getLogger(Tweet.class.getName())
    .log(Level.SEVERE, null, ex);
1833                    }
1834                    System.out.println(date);
1835
1836                    DateFormat newformat=new SimpleDateFormat("
    dd/MM/yyyy");
1837                    String newDate = newformat.format(date);
1838
1839
1840
1841                informe += String.format("Tweet %d. %s\n\nFecha
    : [%s] \n(%d retweets)\n\n\n\n\n", index, rs.getString("text
    "), newDate, rs.getInt("retweetCount"));
```

```
1842                index++;
1843
1844            }
1845
1846
1847            return informe;
1848
1849        } catch (SQLException ex) {
1850            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1851        }
1852
1853        return "";
1854     }
1855
1856     private String obtenerTweetsConMasFavoritos(String cuenta)
1857     {
1858        ResultSet rs;
1859        String informe = "";
1860        try {
1861            DB.getInstance().queryData(String.format("select
    text, createdAt, favoriteCount from tweets where accountName
    = '%s' order by favoriteCount DESC limit 10", cuenta));
1862            rs = DB.getInstance().getResultSet();
1863
1864            int index = 1;
1865
1866            while (rs.next())
1867            {
1868                 DateFormat format = new SimpleDateFormat("
    yyyyMMddhhmmss", Locale.ENGLISH);
1869
1870                  Date date=new Date();
1871                    try {
```

```
1872                        date = format.parse(rs.getString("
    createdAt"));
1873                    } catch (ParseException ex) {
1874                        Logger.getLogger(Tweet.class.getName())
    .log(Level.SEVERE, null, ex);
1875                    }
1876                    System.out.println(date);
1877 
1878                    DateFormat newformat=new SimpleDateFormat("
    dd/MM/yyyy");
1879                    String newDate = newformat.format(date);
1880 
1881                informe += String.format("Tweet %d. %s\n\nFecha
    : [%s]\n(%d favoritos)\n\n\n\n\n", index, rs.getString("text
    "),newDate, rs.getInt("favoriteCount"));
1882                index++;
1883            }
1884 
1885 
1886            return informe;
1887 
1888        } catch (SQLException ex) {
1889            Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1890        }
1891 
1892        return "";
1893     }
1894 
1895        private String convertirIndiceANombreCuenta(int indice){
1896 
1897        switch(indice){
1898            case 0:
1899                return "policia";
```

```
1900                case 1:
1901                    return "guardiacivil";
1902                case 2:
1903                    return "policiademadrid";
1904                case 3:
1905                    return "mossos";
1906
1907            }
1908            return "";
1909           }
1910
1911
1912
1913
1914
1915     public final void actualizarCuentas()
1916     {
1917          CuerposDeSeguridad.removeAllItems();
1918          DB.getInstance().queryData(String.format("select
     distinct accountName from tweets;"));
1919          ResultSet rs;
1920          rs = DB.getInstance().getResultSet();
1921          try {
1922              while(rs.next())
1923              {
1924                  String name = rs.getString("accountName");
1925                  if (!"".equals(name.trim()))
1926                      CuerposDeSeguridad.addItem(name);
1927              }
1928          } catch (SQLException ex) {
1929              Logger.getLogger(Home.class.getName()).log(Level.
     SEVERE, null, ex);
1930          }
1931     }
```

```
1932
1933
1934     public final void actualizarCuentas1()
1935     {
1936         CuerposDeSeguridad1.removeAllItems();
1937         DB.getInstance().queryData(String.format("select
    distinct accountName from tweets where accountName<>'policia
    ' and accountName<>'policiademadrid' and accountName<>'
    mossos' and accountName<>'guardiacivil';"));
1938         ResultSet rs;
1939         rs = DB.getInstance().getResultSet();
1940         try {
1941             while(rs.next())
1942             {
1943                 String name = rs.getString("accountName");
1944                 if (!"".equals(name.trim()))
1945                     CuerposDeSeguridad1.addItem(name);
1946             }
1947         } catch (SQLException ex) {
1948             Logger.getLogger(Home.class.getName()).log(Level.
    SEVERE, null, ex);
1949         }
1950     }
1951
1952
1953
1954     // Variables declaration - do not modify//GEN-BEGIN:
    variables
1955     private javax.swing.JComboBox<String> CuerposDeSeguridad;
1956     private javax.swing.JComboBox<String> CuerposDeSeguridad1;
1957     private javax.swing.JButton DescargaGuardiaCivil;
1958     private javax.swing.JButton GenerarPDF;
1959     private javax.swing.JButton GenerarPDF1;
1960     private javax.swing.JButton ImportarDatos;
```

```
1961     private javax.swing.JButton MossosDeEsquadra;
1962     private javax.swing.JComboBox<String> OpcionesInformes;
1963     private javax.swing.JComboBox<String> OpcionesInformes1;
1964     private javax.swing.JButton PoliciaLocalDeMadrid;
1965     private javax.swing.JButton PoliciaNacional;
1966     private javax.swing.JButton SeleccionarRutaAlmacenamiento;
1967     private javax.swing.JButton SeleccionarRutaAlmacenamiento1;
1968     private javax.swing.JButton almacenamientoExportar;
1969     private javax.swing.JButton almacenamientoImportar;
1970     private javax.swing.JProgressBar barraProgreso;
1971     private javax.swing.JTextField buscarHashtag;
1972     private javax.swing.JButton buscarHashtags;
1973     private javax.swing.JPanel chartContainer;
1974     private javax.swing.JPanel chartContainer1;
1975     private javax.swing.JButton eliminarCuenta;
1976     private javax.swing.JButton exportar;
1977     private principal.ImageBackgroundPanel
    imageBackgroundPanel1;
1978     private principal.ImageBackgroundPanel
    imageBackgroundPanel2;
1979     private principal.ImageBackgroundPanel
    imageBackgroundPanel3;
1980     private principal.ImageBackgroundPanel
    imageBackgroundPanel4;
1981     private principal.ImageBackgroundPanel
    imageBackgroundPanel5;
1982     private principal.ImageBackgroundPanel
    imageBackgroundPanel6;
1983     private principal.ImageBackgroundPanel
    imageBackgroundPanel7;
1984     private javax.swing.JButton importar;
1985     private javax.swing.JButton importarDirectorio;
1986     private com.toedter.calendar.JDateChooser jDateFecha1;
1987     private com.toedter.calendar.JDateChooser jDateFecha2;
```

```
1988     private com.toedter.calendar.JDateChooser jDateFecha3;
1989     private com.toedter.calendar.JDateChooser jDateFecha4;
1990     private javax.swing.JLabel jLabel1;
1991     private javax.swing.JLabel jLabel10;
1992     private javax.swing.JLabel jLabel11;
1993     private javax.swing.JLabel jLabel12;
1994     private javax.swing.JLabel jLabel13;
1995     private javax.swing.JLabel jLabel14;
1996     private javax.swing.JLabel jLabel15;
1997     private javax.swing.JLabel jLabel16;
1998     private javax.swing.JLabel jLabel17;
1999     private javax.swing.JLabel jLabel18;
2000     private javax.swing.JLabel jLabel19;
2001     private javax.swing.JLabel jLabel2;
2002     private javax.swing.JLabel jLabel20;
2003     private javax.swing.JLabel jLabel3;
2004     private javax.swing.JLabel jLabel4;
2005     private javax.swing.JLabel jLabel5;
2006     private javax.swing.JLabel jLabel6;
2007     private javax.swing.JLabel jLabel7;
2008     private javax.swing.JLabel jLabel8;
2009     private javax.swing.JLabel jLabel9;
2010     private javax.swing.JMenuItem jMenuItem1;
2011     private javax.swing.JTabbedPane jTabbedPane1;
2012     private javax.swing.JLabel mensajeEstado;
2013     private javax.swing.JTextField nombreCuenta;
2014     private javax.swing.JTextField txtRuta;
2015     private javax.swing.JTextField txtRuta1;
2016     private javax.swing.JTextField txtRuta2;
2017     private javax.swing.JTextField txtRuta3;
2018     // End of variables declaration//GEN-END:variables
2019 }
```

Código 3.2: Archivo Home.java

3.3. Clase ImageBackgroundPanel

En este apartado se especifica de forma concisa la clase ImageBackgroundPanel. Esta clase contendrá un componente de Java Swing que consiste en un Panel con una imagen de fondo. Esta especificación se muestra en la Tabla 3.3.

Especificación de la clase ImageBackgroundPanel

Clase: *ImageBackgroundPanel*	
Esta clase contendrá un componente de Java Swing que consiste en un Panel con una imagen de fondo.	
Métodos	
+ setImage	Modificador de la variable bgImage
+ getImage	Observador de la variable bgImage
+ paintComponent	Función que dibuja el componente.

Tabla 3.3: Especificación de los métodos de la clase ImageBackgroundPanel

3.3.1. Fichero ImageBackgroundPanel.java

```
1 /*
2  * To change this license header, choose License Headers in
    Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package principal;
7
8 import java.awt.Dimension;
```

```
9  import java.awt.Graphics;
10 import java.beans.*;
11 import java.io.Serializable;
12 import javax.swing.ImageIcon;
13 import javax.swing.JPanel;
14
15 /**
16 * Clase ImageBackgroundPanel contiene un componente de Java
       Swing que consiste en un Panel con una imagen de fondo.
17 *
18 * @author Antonio Luis Ríos
19 * @version 20180626
20 */
21 public class ImageBackgroundPanel extends JPanel {
22
23
24    private ImageIcon bgImage;
25
26    public void setImage(ImageIcon image)
27    {
28        this.bgImage = image;
29    }
30
31    public ImageIcon getImage()
32    {
33        return this.bgImage;
34    }
35
36    @Override
37    public void paintComponent(Graphics g) {
38       Dimension d = getSize();
39       if (this.bgImage != null)
40         g.drawImage(bgImage.getImage(), 0, 0, (int)d.getWidth()
       , (int)d.getHeight() , this);
```

```
41     }
42
43 }
```

Código 3.3: Archivo ImageBackgroundPanel.java

3.4. Clase MainFrame

En este apartado se especifica de forma concisa la clase MainFrame, la cual contiene la ventana principal de la aplicación. Esta especificación se muestra en la Tabla 3.4.

Especificación de la clase MainFrame

Clase: *MainFrame*	
Esta clase contiene la ventana principal de la aplicación.	
Métodos	
+ MainFrame	Constructor de la clase MainFrame.
+ main	Función que inicializa la ventana.

Tabla 3.4: Especificación de los métodos de la clase MainFrame

3.4.1. Fichero MainFrame.java

```
1
2 package principal;
3
4 import baseDeDatos.DB;
5
6 import baseDeDatos.Tweet;
7 import java.beans.PropertyVetoException;
8 import javax.swing.JFrame;
```

```
9  import javax.swing.plaf.basic.BasicInternalFrameUI;
10
11 /**
12 * Clase MainFrame contiene la ventana principal de la aplicació
       n.
13 *
14 * @author Antonio Luis Ríos
15 * @version 20180626
16 */
17 public class MainFrame extends javax.swing.JFrame {
18
19     /**
20      * Creates new form Home
21      */
22     public MainFrame() {
23
24         DB.createInstance("com.mysql.jdbc.Driver","jdbc:mysql
       ://localhost/tfg?useUnicode=true&characterEncoding=UTF-8", "
       tfg", "despintado", "tfg");
25
26         if (!Tweet.tableExists())
27             Tweet.createTable();
28
29
30         initComponents();
31
32         ((BasicInternalFrameUI)Home.getInstance().getUI()).
       setNorthPane(null);
33
34         desktop.add(Home.getInstance());
35
36         try
37         {
38             Home.getInstance().setMaximum(true);
```

```
39          }
40          catch(PropertyVetoException excepcion)
41          {
42              System.err.println("Error al crear frame interno");
43          }
44
45          Home.getInstance().setVisible(true);
46
47      }
48
49      /**
50       * This method is called from within the constructor to
    initialize the form.
51       * WARNING: Do NOT modify this code. The content of this
    method is always
52       * regenerated by the Form Editor.
53       */
54      @SuppressWarnings("unchecked")
55      // <editor-fold defaultstate="collapsed" desc="Generated
    Code">//GEN-BEGIN:initComponents
56      private void initComponents() {
57
58          desktop = new javax.swing.JDesktopPane();
59
60          setDefaultCloseOperation(javax.swing.WindowConstants.
    EXIT_ON_CLOSE);
61
62          javax.swing.GroupLayout layout = new javax.swing.
    GroupLayout(getContentPane());
63          getContentPane().setLayout(layout);
64          layout.setHorizontalGroup(
65              layout.createParallelGroup(javax.swing.GroupLayout.
    Alignment.LEADING)
```

```
66              .addComponent(desktop, javax.swing.GroupLayout.
    Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
    888, Short.MAX_VALUE)
67          );
68          layout.setVerticalGroup(
69              layout.createParallelGroup(javax.swing.GroupLayout.
    Alignment.LEADING)
70              .addComponent(desktop, javax.swing.GroupLayout.
    Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
    561, Short.MAX_VALUE)
71          );
72
73          pack();
74      }// </editor-fold>//GEN-END:initComponents
75
76      /**
77       * @param args the command line arguments
78       */
79      public static void main(String args[]) {
80          /* Set the Nimbus look and feel */
81          //<editor-fold defaultstate="collapsed" desc=" Look and
     feel setting code (optional) ">
82          /* If Nimbus (introduced in Java SE 6) is not available
    , stay with the default look and feel.
83           * For details see http://download.oracle.com/javase/
    tutorial/uiswing/lookandfeel/plaf.html
84           */
85          try {
86              for (javax.swing.UIManager.LookAndFeelInfo info :
    javax.swing.UIManager.getInstalledLookAndFeels()) {
87                  if ("Nimbus".equals(info.getName())) {
88                      javax.swing.UIManager.setLookAndFeel(info.
    getClassName());
89                      break;
```

```
90                  }
91              }
92          } catch (ClassNotFoundException ex) {
93              java.util.logging.Logger.getLogger(MainFrame.class.
    getName()).log(java.util.logging.Level.SEVERE, null, ex);
94          } catch (InstantiationException ex) {
95              java.util.logging.Logger.getLogger(MainFrame.class.
    getName()).log(java.util.logging.Level.SEVERE, null, ex);
96          } catch (IllegalAccessException ex) {
97              java.util.logging.Logger.getLogger(MainFrame.class.
    getName()).log(java.util.logging.Level.SEVERE, null, ex);
98          } catch (javax.swing.UnsupportedLookAndFeelException ex
    ) {
99              java.util.logging.Logger.getLogger(MainFrame.class.
    getName()).log(java.util.logging.Level.SEVERE, null, ex);
100         }
101         //</editor-fold>
102         //</editor-fold>
103
104         /* Create and display the form */
105         java.awt.EventQueue.invokeLater(() -> {
106             MainFrame frame = new MainFrame();
107             frame.setVisible(true);
108             frame.setExtendedState(frame.getExtendedState() |
    JFrame.MAXIMIZED_BOTH);
109         });
110     }
111
112     // Variables declaration - do not modify//GEN-BEGIN:
    variables
113     private javax.swing.JDesktopPane desktop;
114     // End of variables declaration//GEN-END:variables
115 }
```

Código 3.4: Archivo MainFrame.java

3.5. Clase DB

En este apartado se especifica de forma concisa la clase DB, que implementa los métodos necesarios para trabajar con la base de datos de nuestro sistema. Esta especificación se muestra en la Tabla 3.5.

Especificación de la clase DB

Clase: *DB*	
Esta clase implementa los métodos necesarios para trabajar con la base de datos de nuestro sistema.	
Métodos	
+ getDbName	Observador de dbName.
+ setDbName	Modificador de dbName.
+ getDriver	Observador de driver.
+ setDriver	Modificador de driver.
+ getUrl	Observador de url.
+ setUrl	Modificador de url.
+ getUsername	Observador de userName.
+ setUsername	Modificador de userName.
+ getPassword	Observador de password.
+ setPassword	Modificador de password.
+ getInstance	Observador de instance.
+ createInstance	Función para crear una instancia.
-	Constructor de la clase.
+ getConnection	Observador de connection.
+ query	Función para realizar una acción en la base de datos.

+ queryData	Función para realizar una consulta a la base de datos.
+ setPreparedStatement	Modificador de preparedStatement.
+ getPreparedStatement	Observador de preparedStatement.
+ close	Función para cerrar la conexión con la base de datos.
+ getResultSet	Observador de resultSet.

Tabla 3.5: Especificación de los métodos de la clase DB

3.5.1. Fichero DB.java

```
1
2  package baseDeDatos;
3  import java.sql.*;
4  import java.util.Properties;
5  import java.util.logging.Level;
6  import java.util.logging.Logger;
7
8  /**
9  * Clase DB implementa los métodos necesarios para trabajar con
       la base de datos de nuestro sistema.
10 *
11 * @author Antonio Luis Ríos
12 * @version 20180626
13 */
14 public class DB {
15
16     /**
17      * @return the dbName
18      */
19     public String getDbName() {
20         return dbName;
```

```
21      }
22
23      /**
24       * @param dbName the dbName to set
25       */
26      public void setDbName(String dbName) {
27          this.dbName = dbName;
28      }
29
30      /**
31       * @return the driver
32       */
33      public String getDriver() {
34          return driver;
35      }
36
37      /**
38       * @param driver the driver to set
39       */
40      public void setDriver(String driver) {
41          this.driver = driver;
42      }
43
44      /**
45       * @return the url
46       */
47      public String getUrl() {
48          return url;
49      }
50
51      /**
52       * @param url the url to set
53       */
54      public void setUrl(String url) {
```

```
55          this.url = url;
56      }
57
58      /**
59       * @return the username
60       */
61      public String getUsername() {
62          return username;
63      }
64
65      /**
66       * @param username the username to set
67       */
68      public void setUsername(String username) {
69          this.username = username;
70      }
71
72      /**
73       * @return the password
74       */
75      public String getPassword() {
76          return password;
77      }
78
79      /**
80       * @param password the password to set
81       */
82      public void setPassword(String password) {
83          this.password = password;
84      }
85
86      private static DB instance=null;
87      public static DB getInstance(){
88
```

```
89          return instance;
90      }
91
92      public static DB createInstance(String driver, String url,
    String username, String password, String dbName){
93          if(instance==null){
94
95              instance=new DB(driver, url, username, password,
    dbName);
96
97          }
98          return instance;
99      }
100
101
102     private String driver;//="com.mysql.jdbc.Driver";
103     private String url;//="jdbc:mysql://localhost/tfg";
104     private String username;//="tfg";
105     private String password;//="despintado";
106     private String dbName; // "tfg"
107     private Connection conn;
108     private Statement stmt;
109     private PreparedStatement prepStmt;
110     private ResultSet rs;
111
112
113
114
115
116     private DB(String driver, String url, String username,
    String password, String dbName){
117
118         this.driver=driver;
119         this.url=url;
```

```
120         this.username=username;
121         this.password=password;
122         this.dbName = dbName;
123     }
124 
125     public Connection getConnection() throws SQLException,
    ClassNotFoundException, InstantiationException,
    IllegalAccessException{
126 
127         if (conn != null) return conn;
128 
129         Properties props = new Properties();
130         props.put("user", this.getUsername());
131         props.put("password", this.getPassword());
132         props.put("useUnicode", "true");
133 
134 
135 
136         return DriverManager.getConnection(getUrl(), props);
137 
138     }
139 
140     public int query(String sql){
141         this.close();
142         conn=null;
143         stmt=null;
144 
145         try{
146             conn=getConnection();
147             stmt=conn.createStatement();
148             int result = stmt.executeUpdate(sql);
149             return result;
150         }
```

```
151         catch(SQLException | ClassNotFoundException |
    InstantiationException | IllegalAccessException ex){
152             System.out.println(ex.getMessage());
153             return -1;
154         }
155 
156      }
157 
158      public int query(){
159 
160         try{
161             int result = prepStmt.executeUpdate();
162             return result;
163         }
164         catch(SQLException ex){
165             System.out.println(ex.getMessage());
166             return -1;
167         }
168 
169      }
170 
171      public void queryData(String sql){
172         this.close();
173         conn=null;
174         stmt=null;
175 
176         try{
177             conn=getConnection();
178             stmt=conn.createStatement();
179             rs = stmt.executeQuery(sql);
180         }
181         catch(SQLException | ClassNotFoundException |
    InstantiationException | IllegalAccessException ex){
182             System.out.println(ex.getMessage());
```

```
183         }
184 
185     }
186 
187     public void queryData(){
188 
189         try{
190             rs = prepStmt.executeQuery();
191         }
192         catch(SQLException ex){
193             System.out.println(ex.getMessage());
194         }
195 
196     }
197 
198     public void setPreparedStatement(String sql)
199     {
200         this.close();
201 
202         try {
203             conn = getConnection();
204             this.prepStmt = this.conn.prepareStatement(sql);
205         } catch (SQLException | ClassNotFoundException |
    InstantiationException | IllegalAccessException ex) {
206             Logger.getLogger(DB.class.getName()).log(Level.
    SEVERE, null, ex);
207         }
208     }
209 
210     public PreparedStatement getPreparedStatement()
211     {
212         return this.prepStmt;
213     }
214 
```

```
215     public void close()
216     {
217         if(conn!=null){
218             try {
219                 conn.close();
220                 conn = null;
221             } catch (SQLException ex1) {
222 
223             }
224         }
225         if(stmt!=null){
226             try {
227                 stmt.close();
228                 stmt = null;
229             } catch (SQLException ex1) {
230             }
231         }
232 
233         if(prepStmt!=null){
234             try {
235                 prepStmt.close();
236                 prepStmt = null;
237             } catch (SQLException ex1) {
238             }
239         }
240     }
241 
242     /**
243      * @return the rs
244      */
245     public ResultSet getResultSet() {
246         return rs;
247     }
248 
```

```
249
250
251 }
```

Código 3.5: Archivo DB.java

3.6. Clase Hashtag

En este apartado se especifica de forma concisa la clase Hashtag, la cual representa la información que contienen los hashtag de Twitter. Esta especificación se muestra en la Tabla 3.6.

Especificación de la clase Hashtag

Clase: *Hashtag*	
Esta clase representa la información que contienen los hashtag de Twitter.	
Métodos	
+ getStart	Observador de start.
+ setStart	Modificador de start.
+ setStart	Modificador de start.
+ getEnd	Observador de end.
+ setEnd	Modificador de end.
+ getText	Observador de text.
+ setText	Modificador de text.

Tabla 3.6: Especificación de los métodos de la clase Hashtag

3.6.1. Fichero Hashtag.java

```
1
2 package baseDeDatos;
```

```
3
4  import javax.xml.bind.annotation.XmlElement;
5  /**
6  * Clase Hashtag representa la información que contienen los
       hashtag de Twitter.
7  *
8  * @author Antonio Luis Ríos
9  * @version 20180626
10 */
11
12
13 public class Hashtag {
14
15     int start;
16     int end;
17     String text;
18
19
20      public int getStart(){
21
22         return this.start;
23     }
24
25     @XmlElement(name="start")
26     public void setStart(int started) {
27         this.start = started;
28     }
29
30
31     public int getEnd(){
32
33         return this.end;
34     }
35
```

```
36      @XmlElement(name="end")
37      public void setEnd(int ended) {
38          this.end = ended;
39      }
40
41
42      public String getText(){
43
44          return this.text;
45      }
46
47      @XmlElement(name="text")
48      public void setText(String texted) {
49          this.text = texted;
50      }
51
52
53
54  }
```

Código 3.6: Archivo Hashtag.java

3.7. Clase ImportWorker

En este apartado se especifica de forma concisa la clase ImportWorker, la cual realiza tarea de importación de datos a la base de datos del sistema en segundo plano. Esta especificación se muestra en la Tabla 3.7.

Especificación de la clase ImportWorker

Clase: *ImportWorker*

Esta clase se encarga de realizar la tarea de importación de datos, a la base de datos del sistema en segundo plano.	
Métodos	
+ getDoneFunction	Observador de doneFunction.
+ setDoneFunction	Modificador para doneFunction.
+ getPath	Observador de path.
+ setPath	Modificador de path.
+ ImportWorker	Constructor de la clase.
- doInBackground	Función que importa datos y muestra barra de progreso y mensaje del estado de la importación de los datos.
- downloadTweets	Función que descarga los datos y muestra barra de progreso y mensaje del estado de la descarga de los datos.
+ importTweets	Función que comprueba si la importación de los datos se ha realizado correctamente.
- process	Función que notifica el progreso del ImportWorker.
+ done	Función que notifica el final del ImportWorker.
+ setProgressBar	Modificador de la barra de progreso.
+ setStatusMessage	Modificador del mensaje de estado.

Tabla 3.7: Especificación de los métodos de la clase ImportWorker

3.7.1. Fichero ImportWorker.java

```
1
```

```
2  package baseDeDatos;
3
4  import java.io.File;
5  import java.io.IOException;
6  import java.nio.file.Files;
7  import java.nio.file.Paths;
8  import java.text.SimpleDateFormat;
9  import java.util.Arrays;
10 import java.util.Date;
11 import java.util.List;
12 import java.util.function.Function;
13 import java.util.logging.Level;
14 import java.util.logging.Logger;
15 import javax.swing.JLabel;
16 import javax.swing.JOptionPane;
17 import javax.swing.JProgressBar;
18 import javax.swing.SwingWorker;
19 import javax.xml.bind.JAXBContext;
20 import javax.xml.bind.JAXBException;
21 import javax.xml.bind.Unmarshaller;
22 import twitter4j.TwitterException;
23 import twittermanager.TwitterManager;
24 import java.util.regex.*;
25
26 /**
27 * Clase ImportWorker realiza la tarea de importación de datos a
      la base de datos del sistema en segundo plano.
28 *
29 * @author Antonio Luis Ríos
30 * @version 20180626
31 */
32 public class ImportWorker extends SwingWorker<Integer, Integer>
      {
33
```

```
34      /**
35       * @return the doneFunction
36       */
37      public Function<Void,Void> getDoneFunction() {
38          return doneFunction;
39      }
40
41      /**
42       * @param doneFunction the doneFunction to set
43       */
44      public void setDoneFunction(Function<Void,Void>
    doneFunction) {
45          this.doneFunction = doneFunction;
46      }
47
48      /**
49       * @return the path
50       */
51      public String getPath() {
52          return path;
53      }
54
55      /**
56       * @param path the path to set
57       */
58      public void setPath(String path) {
59          this.path = path;
60      }
61
62      private JProgressBar progress;
63      private JLabel statusMessage;
64      private String accountName;
65      private String path;
66      private Function<Void,Void> doneFunction;
```

```
67
68
69
70      public ImportWorker(String accountName, String path){
71          this.accountName = accountName;
72          this.path=path;
73      }
74
75
76
77      @Override
78      protected Integer doInBackground() {
79          progress.setVisible(true);
80
81          if (!"".equals(accountName))
82          {
83              downloadTweets(getPath());
84          }
85          int imported=importTweets(getPath());
86
87          progress.setVisible(false);
88          statusMessage.setText("");
89          return imported;
90      }
91
92
93      private void downloadTweets(String path){
94
95          statusMessage.setText("Se están descargando los Tweets.
        En breve comenzará la importación a la base de datos... ");
96          progress.setMinimum(1);
97          progress.setMaximum(16);
98          progress.setValue(1);
99
```

```
100         TwitterManager tw = new TwitterManager();
101 
102 
103         try {
104 
105             Files.createDirectories(Paths.get(path));
106         } catch (IOException ex) {
107             Logger.getLogger(ImportWorker.class.getName()).log(
    Level.SEVERE, null, ex);
108         }
109 
110 
111 
112         for(int i=1;i<=16;i++){
113            try{
114                 tw.obtenerTweetsDeUsuario(accountName, i, 200,
    Paths.get(path,accountName+i+".xml").toString());
115                 publish(i);
116            }catch(TwitterException e){
117 
118            }
119        }
120 
121     }
122 
123 
124     private int importTweets(String path){
125 
126 
127         if (!Tweet.tableExists()) {
128             Tweet.createTable();
129         }
130 
131         File dir = new File(path);
```

```
132         File[] files = dir.listFiles((d, name) -> name.endsWith
    (".xml"));
133         Integer correctImports;
134         correctImports = 0;
135         int totalImports = 0;
136         int index = 0;
137         progress.setMinimum(0);
138         publish(0);
139         statusMessage.setText("Se están importando los datos,
    en breves momentos podrá consultarlos..."); //texto
140 
141         int numTweets = 0;
142 
143         for (File file : files)
144         {
145             try{
146                 JAXBContext jaxbContext = JAXBContext.
    newInstance(Tweets.class);
147                 Unmarshaller jaxbUnmarshaller = jaxbContext.
    createUnmarshaller();
148                 Tweets tweets = (Tweets) jaxbUnmarshaller.
    unmarshal(file);
149 
150                 for(Tweet tweet : tweets.getTweets()){
151                     numTweets++;
152                 }
153 
154                 System.out.println("Se ha importado el fichero
    " + file.getName() + " correctamente.");
155             }
156             catch(JAXBException e){
157                 System.out.println("Error al importar el
    fichero " + file.getName());
158             }
```

```
159         }
160
161         publish(0);
162         progress.setMaximum(numTweets);
163
164
165         for (File file : files)
166         {
167             String filename = file.getName();
168             String accName = accountName;
169
170             if ("".equals(accName))
171             {
172                 Pattern pattern = Pattern.compile("^([a-zA-Z]+)
    ");
173                 Matcher matcher = pattern.matcher(filename);
174
175                 if (matcher.find())
176                 {
177                     accName = matcher.group();
178                 }
179             }
180
181             System.out.println("Accname: <" + accName + ">");
182
183             try{
184                 JAXBContext jaxbContext = JAXBContext.
    newInstance(Tweets.class);
185                 Unmarshaller jaxbUnmarshaller = jaxbContext.
    createUnmarshaller();
186                 Tweets tweets = (Tweets) jaxbUnmarshaller.
    unmarshal(file);
187
188                 for(Tweet tweet : tweets.getTweets()){
```

```
189                     tweet.setAccountName(accName);
190                     int result = tweet.save();
191                     System.out.println(tweet.getLocation());
192                     if(result > 0)
193                         correctImports++;
194                     totalImports++;
195                     publish(totalImports);
196                     System.out.println(result);
197                 }
198 
199                 System.out.println("Se ha importado el fichero
    " + file.getName() + " correctamente.");
200             }
201             catch(JAXBException e){
202                 System.out.println("Error al importar el
    fichero " + file.getName());
203             }
204             index++;
205 
206         }
207         JOptionPane.showMessageDialog(null, "Obtención de datos
     realizada correctamente", "", JOptionPane.
    INFORMATION_MESSAGE);
208         return correctImports;
209 
210     }
211 
212 
213     @Override
214     protected void process(List<Integer> chunks) {
215         System.out.println("process() esta en el hilo "
216                 + Thread.currentThread().getName());
217         progress.setValue(chunks.get(0));
218     }
```

```
219
220     @Override
221     public void done()
222     {
223         if (doneFunction != null)
224         doneFunction.apply(null);
225     }
226
227     public void setProgressBar(JProgressBar progress)
228     {
229         this.progress = progress;
230     }
231
232       public void setStatusMessage(JLabel statusMessage)
233     {
234         this.statusMessage = statusMessage;
235     }
236
237
238
239 }
```

Código 3.7: Archivo ImportWorker.java

3.8. Clase Tweet

En este apartado se especifica de forma concisa la clase Tweet, la cual representa la información que contiene un Tweet. Esta especificación se muestra en la Tabla 3.8.

Especificación de la clase Tweet

Clase: *Tweet*

Esta clase representa la información que contiene un Tweet.	
Métodos	
+ createTable	Función para crear la tabla de la base de datos.
+ tableExists	Función que comprueba la existencia de la tabla.
+ dropTable	Función para eliminar la tabla de la base de datos.
+ save	Función para actualizar y guardar datos en la base de datos.
+ getIsFavorited	Observador de isFavorited.
+ setIsFavorited	Modificador de isFavorited.
+ getCurrentUserRetweetId	Observador de currentUserRetweetId.
+ setCurrentUserRetweetId	Modificador de currentUserRetweetId.
+ getCreatedAt	Observador de createdAt.
+ setCreatedAt	Modificador de createdAt.
+ getText	Observador de text.
+ setText	Modificador de text.
+ getIsRetweeted	Observador de isRetweeted.
+ setIsRetweeted	Modificador de isRetweeted.
+ getLang	Observador de lang.
+ setLang	Modificador de lang.
+ getIsVerified	Observador de isVerified.
+ setIsVerified	Modificador de isVerified.
+ getIsFollowRequestSent	Observador de isFollowRequestSent.
+ setIsFollowRequestSent	Modificador de isFollowRequestSent.
+ getIsContributorsEnabled	Observador de isContributorsEnabled.

+ setIsContributorsEnabled	Modificador de isContributorsEnabled.
+ getId	Observador de id.
+ setId	Modificador de id.
+ getFriendsCount	Observador de friendsCount.
+ setFriendsCount	Modificador de friendsCount.
+ getFollowersCount	Observador de friendsCount.
+ getRetweetCount	Observador de retweetCount.
+ setRetweetCount	Modificador de retweetCount.
+ getFavoriteCount	Observador de favoriteCount.
+ setFavoriteCount	Modificador de favoriteCount.
+ getHashtagEntities	Observador de hashtagEntities.
+ getHashtagsAsString	Observador de hashtagAsString.
+ getAccountName	Observador de accountName.
+ setAccountName	Modificador de accountName.
+ getUser	Observador de user.
+ setUser	Modificador de user.

Tabla 3.8: Especificación de los métodos de la clase Tweet

3.8.1. Fichero Tweet.java

```
1
2 package baseDeDatos;
3
4 import com.vdurmont.emoji.EmojiParser;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.text.DateFormat;
8 import java.text.ParseException;
9 import java.text.SimpleDateFormat;
10 import java.util.ArrayList;
11 import java.util.Date;
12 import java.util.List;
13 import java.util.Locale;
```

```
14 import java.util.logging.Level;
15 import java.util.logging.Logger;
16 import javax.xml.bind.annotation.XmlElement;
17 import org.apache.commons.lang3.StringEscapeUtils;
18
19
20 /**
21 * Clase Tweet representa la información que contiene un Tweet.
22 *
23 * @author Antonio Luis Ríos
24 * @version 20180626
25 */
26 public class Tweet {
27
28     boolean isTruncated;//Indica si el texto del tweet ha sido
    truncado por exceder de 140 caracteres.
29
30     int inReplyToUserId;//Si el Tweet representado es una
    respuesta, este campo contendrá el ID del autor del Tweet
    original.
31
32     boolean isFavorited;// Si el tweet ha sido favorito para el
     usuario autenticado.
33
34     String source;//Contiene información sobre la herramienta
    que fue usada para enviar el tweet.
35
36     int quotedStatusId;//Este campo solo aparece cuando el
    Tweet es una cita Tweet. Este campo contiene el valor entero
     Tweet ID de la cita Tweet
37
38     int currentUserRetweetId;//Detalla la ID del Tweet del
    propio retweet del usuario (si existe) de este Tweet.
39
```

```
40    String createdAt;//Fecha de alta del usuario en Twitter
41
42    String text;//Texto del tweet
43
44    boolean isRetweeted;//Si el tweet es un retweet
45
46    String lang;//Identificador del lenguaje del tweet (en
      formato BCP 47)
47
48    boolean isVerified;//Indica si la cuenta esta verificada
49
50    boolean translator;//Indica si el usuario es participante
      de la comunidad de traducción de Twitter.
51
52    boolean isFollowRequestSent;// ***¿Seguir la solicitud
      enviada?***
53
54    boolean isContributorsEnabled;/*Indica si la cuenta esta en
       modo contribución.
55    Es una colección de usuarios que contribuyeron a la autoria
       del tweet.*/
56
57    long id;//Representación en formato texto del identificador
       único del tweet.
58
59    /*boolean isGeoEnabled; Indica si el geotagging esta
      activado.
60    Geotagging, una tecnología que nos permitirá etiquetar
      nuestras imágenes con una referencia
61    al lugar donde han sido tomadas.*/
62
63    boolean isDefaultProfile;//***¿Perfil predeteminado?***
64
```

```
65      int friendsCount; //Numero de usuarios al que sigue este
    usuario
66
67      int listedCount;  //Numero de listas públicas de las que es
     miembro el usuario
68
69      //String location;  //Texto con la localización de usuario.
70
71      //int followersCount; //número de seguidores del usuario.
72
73      int retweetCount;   //Numero de veces que el tweet ha sido
    retuiteado
74
75      int favoriteCount;//numero de veces que el tweet ha sido
    marcado como favorito.
76
77      List<Hashtag> hashtagEntities;
78
79      String accountName; //nombre de la cuenta de twitter
80
81      User user; //Usuario que ha creado el tweet
82
83      private static final String tableName="tweets";
84 //-----------------------------------------------------------------


85
86      public static void createTable(){
87
88          DB.getInstance().query("CREATE TABLE "+tableName+" (" +
89              "`id` BIGINT NOT NULL PRIMARY KEY," +
90              "isTruncated boolean null," +
91              "inReplyToUserId int null," +
92              "isFavorited boolean null," +
93              "source nvarchar(3000) null," +
```

```
94              "quotedStatusId int null," +
95              "currentUserRetweetId int null," +
96              "createdAt varchar(300) null," +
97              "text nvarchar(3000) null," +
98              "isRetweeted boolean null," +
99              "lang varchar(300) null," +
100             "isVerified boolean null," +
101             "translator boolean null," +
102             "isFollowRequestSent boolean null," +
103             "isContributorsEnabled boolean null," +
104             "isGeoEnabled boolean null," +
105             "isDefaultProfile boolean null," +
106             "friendsCount int null," +
107             "listedCount int null," +
108             "location varchar(300) null," +
109             "followersCount int null," +
110             "retweetCount int null," +
111             "favoriteCount int null," +
112             "hashtagEntities varchar(300) null," +
113             "accountName varchar(300) null" +
114             ");");
115             DB.getInstance().close();
116     }
117
118
119     public static boolean tableExists(){
120
121         DB.getInstance().queryData("SHOW TABLES LIKE '"+
    tableName+"';");
122         ResultSet rs= DB.getInstance().getResultSet();
123         try {
124             boolean result = rs.next();
125             DB.getInstance().close();
126             return result;
```

```
127         } catch (SQLException ex) {
128             DB.getInstance().close();
129             return false;
130         }
131 
132     }
133 
134     public static void dropTable(){
135 
136         DB.getInstance().query("DROP TABLE '"+tableName+"';");
137         DB.getInstance().close();
138     }
139 
140 
141     public int save(){
142 
143         DateFormat format = new SimpleDateFormat("MMM dd, yyyy"
    , Locale.ENGLISH);
144         Date date=new Date();
145         try {
146             date = format.parse(createdAt);
147         } catch (ParseException ex) {
148             Logger.getLogger(Tweet.class.getName()).log(Level.
    SEVERE, null, ex);
149         }
150         System.out.println(date);
151 
152         DateFormat newformat=new SimpleDateFormat("
    yyyyMMddhhmmss");
153         String newDate = newformat.format(date);
154 
155        int result;
156 
```

```
157         DB.getInstance().setPreparedStatement(String.format("
    SELECT id FROM %s WHERE id=?", tableName));
158          try {
159              DB.getInstance().getPreparedStatement().setLong(1,
    id);
160 
161          } catch (SQLException ex) {
162              Logger.getLogger(Tweet.class.getName()).log(Level.
    SEVERE, null, ex);
163 
164          }
165         System.out.println(DB.getInstance().getPreparedStatement
    ());
166         DB.getInstance().queryData();
167         ResultSet rs= DB.getInstance().getResultSet();
168          try {
169              boolean exists=rs.next();
170              DB.getInstance().close();
171              if(exists){
172 
173              DB.getInstance().setPreparedStatement(String.format
    ("Update %s Set id = ?, isTruncated = ?, inReplyToUserId =
    ?, isFavorited = ?, source = ?, quotedStatusId = ?,
    currentUserRetweetId = ?, createdAt = ?, text = ?,
    isRetweeted = ?, lang = ?, isVerified = ?, translator = ?,
    isFollowRequestSent = ?, isContributorsEnabled = ?,
    isGeoEnabled = ?, isDefaultProfile = ?, friendsCount = ?,
    listedCount = ?, location = ?, followersCount = ?,
    retweetCount = ?, favoriteCount = ?, hashtagEntities = ?,
    accountName= ? where id = ?",tableName));
174              DB.getInstance().getPreparedStatement().setLong(1,
    id);
175              DB.getInstance().getPreparedStatement().setBoolean
    (2, isTruncated);
```

```
176             DB.getInstance().getPreparedStatement().setInt(3,
    inReplyToUserId);
177             DB.getInstance().getPreparedStatement().setBoolean
    (4, isFavorited);
178             DB.getInstance().getPreparedStatement().setNString
    (5, EmojiParser.removeAllEmojis(source));
179             DB.getInstance().getPreparedStatement().setInt(6,
    quotedStatusId);
180             DB.getInstance().getPreparedStatement().setInt(7,
    currentUserRetweetId);
181             DB.getInstance().getPreparedStatement().setString
    (8, newDate);
182             DB.getInstance().getPreparedStatement().setNString
    (9, EmojiParser.removeAllEmojis(text));
183             DB.getInstance().getPreparedStatement().setBoolean
    (10, isRetweeted);
184             DB.getInstance().getPreparedStatement().setString
    (11, lang);
185             DB.getInstance().getPreparedStatement().setBoolean
    (12, isVerified);
186             DB.getInstance().getPreparedStatement().setBoolean
    (13, translator);
187             DB.getInstance().getPreparedStatement().setBoolean
    (14, isFollowRequestSent);
188             DB.getInstance().getPreparedStatement().setBoolean
    (15, isContributorsEnabled);
189             DB.getInstance().getPreparedStatement().setBoolean
    (16, getIsGeoEnabled());
190             DB.getInstance().getPreparedStatement().setBoolean
    (17, isDefaultProfile);
191             DB.getInstance().getPreparedStatement().setInt(18,
    friendsCount);
192             DB.getInstance().getPreparedStatement().setInt(19,
    listedCount);
```

```
193             DB.getInstance().getPreparedStatement().setString
    (20, getLocation());
194             DB.getInstance().getPreparedStatement().setInt(21,
    getFollowersCount());
195             DB.getInstance().getPreparedStatement().setInt(22,
    retweetCount);
196             DB.getInstance().getPreparedStatement().setInt(23,
    favoriteCount);
197             DB.getInstance().getPreparedStatement().setString
    (24, getHashtagsAsString());
198             DB.getInstance().getPreparedStatement().setString
    (25, accountName);
199             DB.getInstance().getPreparedStatement().setLong(26,
     id);
200             System.out.println(DB.getInstance().
    getPreparedStatement());
201             result = DB.getInstance().query();
202             System.out.println("estoy en el update");
203             DB.getInstance().close();
204             return result;
205             }
206             else{
207                 DB.getInstance().setPreparedStatement(String.
    format("Insert into %s (id, isTruncated, inReplyToUserId,
    isFavorited, source, quotedStatusId, currentUserRetweetId,
    createdAt, text, isRetweeted, lang, isVerified, translator,
    isFollowRequestSent, isContributorsEnabled, isGeoEnabled,
    isDefaultProfile, friendsCount, listedCount, location,
    followersCount, retweetCount, favoriteCount, hashtagEntities
    , accountName) values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
     ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,?)", tableName));
208                 DB.getInstance().getPreparedStatement().setLong
    (1, id);
```

```
209                 DB.getInstance().getPreparedStatement().
    setBoolean(2, isTruncated);
210                 DB.getInstance().getPreparedStatement().setInt
    (3, inReplyToUserId);
211                 DB.getInstance().getPreparedStatement().
    setBoolean(4, isFavorited);
212                 DB.getInstance().getPreparedStatement().
    setNString(5, EmojiParser.removeAllEmojis(source));
213                 DB.getInstance().getPreparedStatement().setInt
    (6, quotedStatusId);
214                 DB.getInstance().getPreparedStatement().setInt
    (7, currentUserRetweetId);
215                 DB.getInstance().getPreparedStatement().
    setString(8, newDate);
216                 DB.getInstance().getPreparedStatement().
    setNString(9, EmojiParser.removeAllEmojis(text));
217                 DB.getInstance().getPreparedStatement().
    setBoolean(10, isRetweeted);
218                 DB.getInstance().getPreparedStatement().
    setString(11, lang);
219                 DB.getInstance().getPreparedStatement().
    setBoolean(12, isVerified);
220                 DB.getInstance().getPreparedStatement().
    setBoolean(13, translator);
221                 DB.getInstance().getPreparedStatement().
    setBoolean(14, isFollowRequestSent);
222                 DB.getInstance().getPreparedStatement().
    setBoolean(15, isContributorsEnabled);
223                 DB.getInstance().getPreparedStatement().
    setBoolean(16, getIsGeoEnabled());
224                 DB.getInstance().getPreparedStatement().
    setBoolean(17, isDefaultProfile);
225                 DB.getInstance().getPreparedStatement().setInt
    (18, friendsCount);
```

```
226                 DB.getInstance().getPreparedStatement().setInt
    (19, listedCount);
227                 DB.getInstance().getPreparedStatement().
    setString(20, getLocation());
228                 DB.getInstance().getPreparedStatement().setInt
    (21, getFollowersCount());
229                 DB.getInstance().getPreparedStatement().setInt
    (22, retweetCount);
230                 DB.getInstance().getPreparedStatement().setInt
    (23, favoriteCount);
231                 DB.getInstance().getPreparedStatement().
    setString(24, getHashtagsAsString());
232                 DB.getInstance().getPreparedStatement().
    setString(25, accountName);
233                 System.out.println(DB.getInstance().
    getPreparedStatement());
234                 result = DB.getInstance().query();
235                 System.out.println("estoy en el insert");
236                 DB.getInstance().close();
237                 return result;
238
239             }
240
241
242         } catch (SQLException ex) {
243             System.out.println("estoy en el catch");
244             DB.getInstance().close();
245             return -1;
246
247         }
248
249
250
251
```

```
252     }
253
254
255     //Indica si el texto del tweet ha sido truncado por exceder
        de 140 caracteres.
256     public boolean getIsTruncated(){
257
258         return this.isTruncated;
259     }
260
261     @XmlElement(name="isTruncated")
262     public void setIsTruncated(boolean truncated) {
263         this.isTruncated = truncated;
264     }
265
266
267
268     //Si el Tweet representado es una respuesta, este campo
      contendrá el ID del autor del Tweet original.
269     public int getInReplyToUserId(){
270
271         return this.inReplyToUserId;
272     }
273
274     @XmlElement(name="inReplyToUserId")
275     public void setInReplyToUserId(int replyToUserId) {
276         this.inReplyToUserId = replyToUserId;
277     }
278
279
280     // Si el tweet ha sido favorito para el usuario autenticado
      .
281      public boolean getIsFavorited(){
282
```

```
283         return this.isFavorited;
284     }
285 
286     @XmlElement(name="isFavorited")
287     public void setIsFavorited(boolean favorited) {
288         this.isFavorited = favorited;
289     }
290 
291 
292     //Contiene información sobre la herramienta que fue usada
    para enviar el tweet.
293     public String getSource(){
294 
295         return this.source;
296     }
297 
298     @XmlElement(name="source")
299     public void setSource(String sourced) {
300         this.source = sourced;
301     }
302 
303 
304 
305     //Este campo solo aparece cuando el Tweet es una cita Tweet
    . Contiene el valor entero Tweet ID de la cita Tweet
306     public int getQuotedStatusId(){
307 
308         return this.quotedStatusId;
309     }
310 
311     @XmlElement(name="quotedStatusId")
312     public void setQuotedStatusId(int quoted) {
313         this.quotedStatusId = quoted;
314     }
```

```
315
316
317     //Detalla la ID del Tweet del propio retweet del usuario (
    si existe) de este Tweet.
318     public int getCurrentUserRetweetId(){
319
320         return this.currentUserRetweetId;
321     }
322
323     @XmlElement(name="currentUserRetweetId")
324     public void setCurrentUserRetweetId(int currented) {
325         this.currentUserRetweetId = currented;
326     }
327
328
329     //Fecha de alta del usuario en Twitter
330      public String getCreatedAt(){
331
332         return this.createdAt;
333     }
334
335     @XmlElement(name="createdAt")
336     public void setCreatedAt(String created) {
337         this.createdAt = created;
338     }
339
340
341
342     //Texto del tweet
343      public String getText(){
344
345         return this.text;
346     }
347
```

```
348     @XmlElement(name="text")
349     public void setText(String texted) {
350         this.text = texted;
351     }
352 
353     //Si el tweet es un retweet
354     public boolean getIsRetweeted(){
355 
356         return this.isRetweeted;
357     }
358 
359     @XmlElement(name="isRetweeted")
360     public void setIsRetweeted(boolean retweeted) {
361         this.isRetweeted = retweeted;
362     }
363 
364 
365     //Identificador del lenguaje del tweet (en formato BCP 47)
366      public String getLang(){
367 
368         return this.lang;
369     }
370 
371     @XmlElement(name="lang")
372     public void setLang(String langed) {
373         this.lang = langed;
374     }
375 
376 
377     //Indica si la cuenta esta verificada
378      public boolean getIsVerified(){
379 
380         return this.isVerified;
381     }
```

```
382
383     @XmlElement(name="isVerified")
384     public void setIsVerified(boolean verified) {
385         this.isVerified = verified;
386     }
387
388
389
390     //Indica si el usuario es participante de la comunidad de
    traducción de Twitter.
391     public boolean getTranslator(){
392
393         return this.translator;
394     }
395
396     @XmlElement(name="translator")
397     public void setTranslator(boolean translatored) {
398         this.translator = translatored;
399     }
400
401
402     // ¿Seguir la solicitud enviada?
403
404      public boolean getIsFollowRequestSent(){
405
406         return this.isFollowRequestSent;
407     }
408
409     @XmlElement(name="isFollowRequestSent")
410     public void setIsFollowRequestSent(boolean requested) {
411         this.isFollowRequestSent = requested;
412     }
413
414
```

```
415     /*Indica si la cuenta esta en modo contribución.
416     Es una colección de usuarios que contribuyeron a la autoria
        del tweet.*/
417     public boolean getIsContributorsEnabled(){
418
419         return this.isContributorsEnabled;
420     }
421
422     @XmlElement(name="isContributorsEnabled")
423     public void setIsContributorsEnabled(boolean contributored)
        {
424         this.isContributorsEnabled = contributored;
425     }
426
427
428     //Representación en formato texto del identificador único
      del tweet.
429      public long getId(){
430
431         return this.id;
432     }
433
434     @XmlElement(name="id")
435     public void setId(long ided) {
436         this.id = ided;
437     }
438
439
440     /*Indica si el geotagging esta activado.
441     Geotagging, una tecnología que nos permitirá etiquetar
      nuestras imágenes con una referencia
442     al lugar donde han sido tomadas.*/
443     /*public boolean getIsGeoEnabled(){
444
```

```
445         return this.isGeoEnabled;
446     }
447 
448     @XmlElement(name="user.isGeoEnabled")
449     public void setIsGeoEnabled(boolean geoed) {
450         this.isGeoEnabled = geoed;
451     }*/
452 
453     public boolean getIsGeoEnabled()
454     {
455         return this.user.getIsGeoEnabled();
456     }
457 
458 
459     //***¿Perfil predeteminado?***
460      public boolean getIsDefaultProfile(){
461 
462         return this.isDefaultProfile;
463     }
464 
465     @XmlElement(name="isDefaultProfile")
466     public void setIsDefaultProfile(boolean defaulted) {
467         this.isDefaultProfile = defaulted;
468     }
469 
470     // Numero de usuarios al que sigue este usuario
471      public int getFriendsCount(){
472 
473         return this.friendsCount;
474     }
475 
476     @XmlElement(name="friendsCount")
477     public void setFriendsCount(int friended) {
478         this.friendsCount = friended;
```

```
479     }
480
481     //Numero de listas públicas de las que es miembro el
    usuario.
482     public int getListedCount(){
483
484         return this.listedCount;
485     }
486
487     @XmlElement(name="listedCount")
488     public void setListedCount(int listeded) {
489         this.listedCount = listeded;
490     }
491
492     //Texto con la localización de usuario.
493     /*public String getLocation(){
494
495         return this.location;
496     }
497
498     @XmlElement(name="user.location")
499     public void setLocation(String locationed) {
500         this.location = locationed;
501     }*/
502
503     public String getLocation()
504     {
505         return this.user.getLocation();
506     }
507
508
509
510
511     public int getFollowersCount()
```

```
512     {
513         return this.user.getFollowersCount();
514     }
515 
516 
517     //Numero de veces que el tweet ha sido retuiteado
518     public int getRetweetCount(){
519 
520         return this.retweetCount;
521     }
522 
523     @XmlElement(name="retweetCount")
524     public void setRetweetCount(int retweeted) {
525         this.retweetCount = retweeted;
526     }
527 
528 
529 
530     //numero de veces que el tweet ha sido marcado como
    favorito
531       public int getFavoriteCount(){
532 
533         return this.favoriteCount;
534     }
535 
536     @XmlElement(name="favoriteCount")
537     public void setFavoriteCount(int favourited) {
538         this.favoriteCount = favourited;
539     }
540 
541 
542     public List<Hashtag> getHashtagEntities(){
543 
544         return this.hashtagEntities;
```

```
545     }
546
547     public String getHashtagsAsString()
548     {
549         String hashtags = "";
550         if (this.hashtagEntities == null) return "";
551         for(Hashtag h : this.hashtagEntities)
552         {
553             hashtags += "#" + h.getText();
554         }
555
556         return hashtags;
557     }
558
559     @XmlElement(name="hashtagEntities")
560     public void setHashtagEntities(List<Hashtag>
    hashtagEntitiesed) {
561         this.hashtagEntities = hashtagEntitiesed;
562     }
563
564     public String getAccountName()
565     {
566
567         return accountName;
568     }
569
570     public void setAccountName(String text) {
571         this.accountName = text;
572     }
573
574     public User getUser()
575     {
576         return this.user;
577     }
```

```
578
579     @XmlElement(name="user")
580     public void setUser(User user) {
581         this.user = user;
582     }
583
584
585
586
587 }
```

Código 3.8: Archivo Tweet.java

3.9. Clase Tweets

En este apartado se especifica de forma concisa la clase Tweets, la cual representa un conjunto de Tweets. Esta especificación se muestra en la Tabla 3.9.

Especificación de la clase Tweets

Clase: *Tweets*	
Esta clase representa un conjunto de Tweets.	
Métodos	
+ getTweets	Observador de tweets.
+ setTweet	Modificador de tweets.

Tabla 3.9: Especificación de los métodos de la clase Tweets

3.9.1. Fichero Tweets.java

```
1
2 package baseDeDatos;
```

```
3
4  import java.util.List;
5  import javax.xml.bind.annotation.XmlAccessType;
6  import javax.xml.bind.annotation.XmlAccessorType;
7  import javax.xml.bind.annotation.XmlElement;
8  import javax.xml.bind.annotation.XmlRootElement;
9
10 /**
11 * Clase Tweets representa un conjunto de tweets.
12 *
13 * @author Antonio Luis Ríos
14 * @version 20180626
15 */
16
17
18
19 @XmlRootElement(name="tweets")
20 @XmlAccessorType(XmlAccessType.FIELD)
21 public class Tweets {
22 //Convierte los ficheros XML a objetos tweets
23    @XmlElement(name="tweet")
24     List<Tweet> tweets;
25
26     public List<Tweet> getTweets(){
27
28         return this.tweets;
29     }
30
31
32     public void setTweet(List<Tweet> tweetsed) {
33         this.tweets = tweetsed;
34     }
35
36
```

```
37 }
```

Código 3.9: Archivo Tweets.java

3.10. Clase User

En este apartado se especifica de forma concisa la clase User, la cual representa un usuario de Twitter. Esta especificación se muestra en la Tabla 3.10.

Especificación de la clase User

Clase: *User*	
Esta clase representa a un usuario de Twitter.	
Métodos	
+ getFollowersCount	Observador de followersCount.
+ setFollowersCount	Modificador del followersCount.
+ getIsGeoEnabled	Observador de IsGeoEnabled.
+ setIsGeoEnabled	Modificador de IsGeoEnabled.
+ getLocation	Observador de getLocation.
+ setLocation	Modificador de setLocation.

Tabla 3.10: Especificación de los métodos de la clase User

3.10.1. Fichero User.java

```
1
2 package baseDeDatos;
3 import javax.xml.bind.annotation.XmlElement;
4
5 /**
6 * Clase User representa un usuario de Twitter.
7 *
8 * @author Antonio Luis Ríos
```

```
9  * @version 20180626
10 */
11
12 public class User {
13
14     private String location;
15
16     private int followersCount;
17
18     private boolean isGeoEnabled;
19
20     public int getFollowersCount(){
21
22         return this.followersCount;
23     }
24
25     @XmlElement(name="followersCount")
26     public void setFollowersCount(int followersCount) {
27         this.followersCount = followersCount;
28     }
29
30     /*Indica si el geotagging esta activado.
31     Geotagging, una tecnología que nos permitirá etiquetar
       nuestras imágenes con una referencia
32     al lugar donde han sido tomadas.*/
33     public boolean getIsGeoEnabled(){
34
35         return this.isGeoEnabled;
36     }
37
38     @XmlElement(name="isGeoEnabled")
39     public void setIsGeoEnabled(boolean geoed) {
40         this.isGeoEnabled = geoed;
41     }
```

```
42
43      public String getLocation(){
44
45          return this.location;
46      }
47
48      @XmlElement(name="location")
49      public void setLocation(String locationed) {
50          this.location = locationed;
51      }
52  }
```

CÓDIGO 3.10: ARCHIVO USER.JAVA

3.11. Clase TwitterManager

En este apartado se especifica de forma concisa la clase TwitterManager, la cual representa las conexiones con la API de Twitter. Esta especificación se muestra en la Tabla 3.11.

Especificación de la clase TwitterManager

Clase: *TwitterManager*	
Esta clase contendrá las conexiones con la API de Twitter.	
Métodos	
+ copyToJsonString	Función para convertir un JSON a string
+ obtenerTodosTweets	Función para obtener el total de tweets máximos permitidos en una sola descarga.
+ obtenerTweetsDeUsuario	Función para obtener el tweet de un usuario.

Tabla 3.11: Especificación de los métodos de la clase TwitterManager

3.11.1. Fichero TwitterManager.java

```
1
2  package twittermanager;
3
4  import com.google.gson.Gson;
5  import java.io.BufferedWriter;
6  import java.io.FileWriter;
7  import java.util.ArrayList;
8  import java.util.List;
9  import java.util.logging.Level;
10 import java.util.logging.Logger;
11 import org.json.JSONObject;
12 import org.json.XML;
13 import twitter4j.IDs;
14
15 import twitter4j.Paging;
16 import twitter4j.Query;
17 import twitter4j.QueryResult;
18 import twitter4j.Status;
19 import twitter4j.Twitter;
20 import twitter4j.TwitterException;
21 import twitter4j.TwitterFactory;
22 import twitter4j.conf.ConfigurationBuilder;
23 import twitter4j.json.DataObjectFactory;
24
25 /**
26 * Clase TwitterManager representa las conexiones con la API de
     Twitter.
27 *
```

```
28 * @author Antonio Luis Ríos
29 * @version 20180626
30 */
31 public class TwitterManager {
32 
33     /**
34      * @param args the command line arguments
35      */
36    private final String consumerKeys_="
   n9eCWVojQsBODYXZoTH0gRufJ";
37    private final String consumerSecret_="
   wIiwPTYMaGyMb6mHsXzbQuOZSJ1Bfs50dodg7evNv4B0ehFCgX";
38    private final String accessToken_="963083620587057152-0
   ZC5imnMy5RAHd2n4JkISZR6IHaspei";
39    private final String accessTokenSecret_="
   vYGhXANmeVB5Fpryyx4PaccfbTiiFGUghcZFTMuH1rQYe";
40    private ConfigurationBuilder CB_=new ConfigurationBuilder()
41            .setDebugEnabled(true)
42            .setOAuthConsumerKey(consumerKeys_)
43            .setOAuthConsumerSecret(consumerSecret_)
44            .setOAuthAccessToken(accessToken_)
45            .setOAuthAccessTokenSecret(accessTokenSecret_);
46    private final Twitter tw_=new TwitterFactory(CB_.build()).
   getInstance();
47 
48 
49 
50    public void escribirTweets() throws TwitterException{
51 
52        String tweet="Sublime text";
53        tw_.updateStatus(tweet);
54    }
55 
56 
```

```
57    public static String copyToJsonString(Object modelObject ){
58     String data="";
59     try {
60         Gson gson= new Gson();
61         data=gson.toJson(modelObject);
62
63     } catch (Exception e) {
64     }
65     return data;
66 }
67
68    public void obtenerTodosTweets(String nombreUsuario,String
      prefijo){
69
70        for(int i=1;i<=16;i++){
71            try{
72            obtenerTweetsDeUsuario(nombreUsuario, i, 200,
      prefijo+i+".xml");
73            }catch(TwitterException e){
74
75            }
76        }
77    }
78
79
80
81    public void obtenerTweetsDeUsuario(String nombreUsuario, int
       numPagina,int numTweets,String nombFich) throws
      TwitterException{
82 //First param of Paging() is the page number, second is the
      number per page (this is capped around 200 I think.
83 Paging paging = new Paging(numPagina,numTweets);
84 List<Status> statuses = tw_.getUserTimeline(nombreUsuario,
      paging);
```

```
85 String TweetsXML="<?xml version=\"1.0\" encoding=\"UTF-8\"
       standalone=\"no\" ?><tweets>";
86 for (Status status : statuses) {
87
88     String JsonString=copyToJsonString(status);
89
90     JSONObject json = new JSONObject(JsonString);
91     String xml = XML.toString(json);
92     TweetsXML+="<tweet>"+xml+"</tweet>\n";
93
94
95        }
96
97         TweetsXML+="</tweets>";
98
99         int i=0;
100        int pos=0;
101        do{
102            pos=TweetsXML.indexOf("<"+i+">");
103            TweetsXML=TweetsXML.replace("<"+i+">", "<item>");
104            TweetsXML=TweetsXML.replace("</"+i+">", "</item>");
105            i++;
106        }while(pos!=-1);
107
108
109
110        System.out.println(TweetsXML);
111        System.out.println(TweetsXML.length());
112
113 BufferedWriter writer = null;
114        try {
115
116            writer = new BufferedWriter(new FileWriter(nombFich
       ));
```

```
117                 writer.write(TweetsXML);
118             } catch (Exception e ) {
119             } finally {
120                 try {
121                     // Close the writer regardless of what happens
    ...
122                     if(writer!=null)
123                         writer.close();
124                 } catch (Exception e) {
125                 }
126             }
127 
128 
129     }
130 
131 
132 
133 
134 }
```

Código 3.11: Archivo TwitterManager.java