ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

EPSC

UNIVERSIDAD DE CÓRDOBA
ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA INFORMÁTICA

ESPECIALIDAD EN COMPUTACIÓN

TRABAJO DE FIN DE GRADO

ANÁLISIS DE LAS EMOCIONES EN ALUMNOS MIENTRAS INTERACTÚAN CON UN SISTEMA DE APRENDIZAJE CON TUTOR VIRTUAL

- MANUAL DE CÓDIGO -

Autor:

Ángel Murcia Díaz

Director:

D. Cristóbal Romero Morales

Córdoba, 22 de enero de 2021

Publicado en enero 2021 por

Ángel Murcia Díaz

i52mudia@uco.es

D. Cristóbal Romero Morales,
profesor del departamento de informática y análisis numérico de la Escuela Politécnica Superior de la Universidad de Córdoba.

INFORMA

Que el presente trabajo fin de grado titulado *Análisis de las emociones en alumnos mientras interactúan con un sistema de aprendizaje con tutor virtual*, que constituye la memoria presentada por D Ángel Murcia Díaz para aspirar al grado en Ingeniería Informática, ha sido realizado bajo mi dirección en la Escuela Politécnica Superior de la Universidad de Córdoba reuniendo, a mi juicio, las condiciones necesarias exigidas en este tipo de trabajos. Y para que así conste se expide y firma el presente informe en Córdoba, 22 de enero de 2021

EL AUTOR

D Ángel Murcia Díaz

EL DIRECTOR

Fdo: Cristóbal Romero Morales

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Introducción

En este manual se encuentra todo el código desarrollado durante el proyecto. Todo este código pertenece a la aplicación AEV(Análisis de Emociones en Vídeos).

Figura 1.1: Icono de la aplicación AEV

Este manual se dividirá en varias secciones, cada una de ellas corresponderá al código de un paquete diferente. Se indicará el nombre de cada uno de los ficheros, para posteriormente presentar el código de cada uno de ellos. Este código contendrá diversos comentarios que explicará sus características.

La realización de este documento se llevará a cabo de forma que cualquier persona con conocimiento de programación pueda comprender el código fácilmente.

El código en su totalidad esta disponible en Github, en el siguiente enlace Código en GITHUB.

El código de esta aplicación está estructurado en varios paquetes, incluyendo el paquete de las ventanas, es decir, la interfaz gráfica. A continuación se mostrará el código de cada uno de los paquetes.

Código PaquetePreAnalisis

2.1 Principal.java

```
1  //INCLUSION NECESARIA DEL PAQUETE
2  package PaquetePreAnalisis;
3
4
5  //IMPORT NECESARIOS
6  import PaqueteVentanas.VentanaDespuesDeAnalisis;
7  import PaqueteVentanas.VentanaUsuario;
8  import PaqueteVentanas.VentanaSeguimiento;
9  import PaqueteVentanas.VentanaPrincipal;
10 import PaqueteVentanas.VentanaGraficas;
11 import PaqueteVentanas.VentanaOpciones;
12 import PaqueteVentanas.VentanaResultadosFinales;
13 import PaqueteVentanas.VentanaUsuarioActualizar;
14 import PaqueteVentanas.VentanaUsuarioNuevo;
15 import PaqueteAnalisisVideo.HiloProcesoPrincipal;
16 import java.util.ArrayList;
17
18 /**
19  *
20  * @author Angel Murcia Diaz
21  */
22
23 //CLASE PRINCIPAL (MAIN)
24 public class Principal {
25
26     ////////////////////////////
27     //DECLARACION DE CONTADORES//
```

```
28      ////////////////////////////
29
30      //DECLARACION DE CONTADORES GENERALES//
31
32      //CONTADOR anger
33      public static int angerContador = 0;
34
35      //CONTADOR contempt
36      public static int contemptContador = 0;
37
38      //CONTADOR disgust
39      public static int disgustContador = 0;
40
41      //CONTADOR fear
42      public static int fearContador= 0;
43
44      //CONTADOR happiness
45      public static int happinessContador = 0;
46
47      //CONTADOR neutral
48      public static int neutralContador = 0;
49
50      //CONTADOR sadness
51      public static int sadnessContador = 0;
52
53      //CONTADOR surprise
54      public static int surpriseContador = 0;
55
56      //CONTADOR NO ANALIZADO
57      public static int noAnalizadoContador = 0;
58
59      //CONTADOR TOTAL DE FOTOGRAMAS
60      public static int nFotosAnalizadas = 0;
61
62      //CONTADOR (VECES), para la version de prueba
63      public static int veces = 0;
```

```
64
65      //CONTADORES MAYORITARIOS//
66
67      //CONTADOR anger MAYORITARIO
68      public static int angerContadorMayor = 0;
69
70      //CONTADOR contempt MAYORITARIO
71      public static int contemptContadorMayor = 0;
72
73      //CONTADOR disgust MAYORITARIO
74      public static int disgustContadorMayor = 0;
75
76      //CONTADOR fear MAYORITARIO
77      public static int fearContadorMayor = 0;
78
79      //CONTADOR happiness MAYORITARIO
80      public static int happinessContadorMayor = 0;
81
82      //CONTADOR neutral MAYORITARIO
83      public static int neutralContadorMayor = 0;
84
85      //CONTADOR sadness MAYORITARIO
86      public static int sadnessContadorMayor = 0;
87
88      //CONTADOR surprise MAYORITARIO
89      public static int surpriseContadorMayor = 0;
90
91      //CONTADORES EN PORCENTAJES//
92
93      //CONTADOR anger PORCENTAJE
94      public static double angerContadorPorcentaje = 0.0;
95
96      //CONTADOR contempt PORCENTAJE
97      public static double contemptContadorPorcentaje = 0.0;
98
99      //CONTADOR disgust PORCENTAJE
```

```
100     public static double disgustContadorPorcentaje = 0.0;
101
102     //CONTADOR fear PORCENTAJE
103     public static double fearContadorPorcentaje = 0.0;
104
105     //CONTADOR happiness PORCENTAJE
106     public static double happinessContadorPorcentaje = 0.0;
107
108     //CONTADOR neutral PORCENTAJE
109     public static double neutralContadorPorcentaje = 0.0;
110
111     //CONTADOR sadness PORCENTAJE
112     public static double sadnessContadorPorcentaje = 0.0;
113
114     //CONTADOR surprise PORCENTAJE
115     public static double surpriseContadorPorcentaje = 0.0;
116
117      ////////////////////////////////////
118     //VARIABLES (OPCIONES CONFIGURABLES)//
119     ////////////////////////////////////
120
121     //CON VALORES QUE UTILIZARA EL PROGRAMA//
122
123     //UMBRAL PARA ACEPTAR EL SENTIMIENTO
124     public static double umbralSentimiento = 0.1 ;
125
126     //SEGUNDOS ENTRE FRAMES
127     public static double segundosEntreFrames = 2;
128
129     //RUTA DONDE GUARDAR TODO
130     public static String rutaGuardar ;
131
132     //HAY RUTA?
133     public static boolean hayRutaConfirmada ;
134
135     //GUARDAR FOTOGRAMAS
```

```
136     public static boolean conservar;
137
138     //VALORES QUE INTRODUCIRA EL USUARIO//
139
140     //UMBRAL PARA ACEPTAR EL SENTIMIENTO
141     public static double valorUsuarioUmbral = 0.1;
142
143     //SEGUNDOS ENTRE FRAMES
144     public static double valorUsuarioSegundosFrame = 2;
145
146     //RUTA DONDE GUARDAR TODO
147     public static String valorUsuarioRuta = "";
148
149     //HAY RUTA?
150     public static boolean hayRutaUsuario = false;
151
152     //GUARDAR FOTOGRAMAS
153     public static boolean valorUsuarioConservar = false;
154
155     ////////////////////////////////////////////
156     //VARIABLES NECESARIAS PARA EL PROCESAMIENTO//
157     ////////////////////////////////////////////
158
159     //vector que contendra los datos de cada FRAME
160     public static ArrayList<Contenedor> contenedorFrames = new ArrayList
            <>();
161
162     //MENSAJE QUE APARECERA EN LA VENTANA DE ERROR
163     public static String error = null;
164
165     //COMPRUEBA SI EL USUARIO SE HA IDENTIFICADO O NO
166     public static boolean sesionIniciada = false;
167
168     //HILO PARA CREAR EL SUDPROCESO QUE HARA TODO EL ANALISIS DE VIDEO
169     public static HiloProcesoPrincipal hilo1=new HiloProcesoPrincipal("
            Proceso principal de analisis de video");
```

```
170
171     //STRING QUE CONTIENE EL VIDEO A ANALIZAR
172     public static String videoFile = null;
173
174     ////////////////////
175     //DATOS (USUARIO)//
176     ////////////////////
177
178     //TIPO DE VERSION DE LA API DE MICROSOFT
179     public static boolean versionPrueba = true;
180
181     //VARIABLE PARA SABER CUANDO EL PROGRAMA ESTA EN PAUSA
182     public static boolean estaEnPausa = false;
183
184     //TODOS LOS DATOS (USUARIO COMPLETO)
185     public static Usuario user = new Usuario();
186
187     ////////////
188     //VENTANAS//
189     ////////////
190
191     //CREACION VENTANA DE SEGUIMIENTO
192     public static VentanaPrincipal ventana_principal = new VentanaPrincipal
            () ;
193
194     //CREACION VENTANA DE SEGUIMIENTO
195     public static VentanaSeguimiento ventana_seguimiento = new
            VentanaSeguimiento() ;
196
197     //CREACION VENTANA DE ERROR
198     public static VentanaOpciones ventana_opciones = new VentanaOpciones();
199
200     //VENTANA DE ADMINISTRACION USUARIOS
201     public static VentanaUsuario ventana_usuarios = new VentanaUsuario();
202
203     //VENTANA NUEVO USUARIO
```

```
204     public static VentanaUsuarioNuevo ventana_usuario_nuevo = new
            VentanaUsuarioNuevo();
205
206     //VENTANA ACTUALIZAR USUARIO
207     public static VentanaUsuarioActualizar ventana_usuario_actualizar = new
             VentanaUsuarioActualizar();
208
209     //VENTANA DESPUES DE ANALISIS
210     public static VentanaDespuesDeAnalisis ventana_final = new
            VentanaDespuesDeAnalisis();
211
212     //VENTANA GRAFICAS FINALES
213     public static VentanaGraficas ventana_graficas = new VentanaGraficas();
214
215     //VENTANA RESULTADOS FINALES
216     public static VentanaResultadosFinales ventana_resultados = new
            VentanaResultadosFinales();
217
218     ////////////////////////////////
219     //VARIABLES PARA DATASET UNICO//
220     ////////////////////////////////
221
222     //NOMBRE DEL DATASET UNICO
223     static public String nombreDatasetUnico = "datasetUnico";
224
225     //MATRIZ PARA ALMACENAR LAS FILAS DEL ANTERIOR DOCUEMNTO
226     static public ArrayList<ArrayList<Double>> arrayDatosDatasetUnico = new
             ArrayList<>();
227
228     //Vector donde cada posicion es el nombre de cada fila
229     //VECTOR PARA ALMACENAR EN NOMBRE DE LOS VIDEOS DE CADA FILA
230     static public ArrayList<String> nombresDatasetUnico = new ArrayList<>()
            ;
231
232     //VECTOR PARA ALMACENAR LA POSICION QUE OCUPA CADA UNA DE LAS FILAS
233     static public ArrayList<Integer> rowsDatasetUnico = new ArrayList<>();
```

```
234
235       ////////////////
236       //MAS VARIABLES//
237       ////////////////
238
239      //NOMBRE DEL VIDEO PARA TENERLO CUANDO SE GUARDEN LOS RESULTADOS
240       public static String nombreFichero;
241
242       //VARIABLE PARA MATAR UN HILO
243       public static boolean matarHilo;
244
245       //VARIABLE QUE INDICA SI EXISTE UN ERROR DURANTE EL ANALISIS
246       public static boolean errorDuranteAnalisis;
247
248       //VARIABLE PARA INDICAR EL FORMATO DE LOS RESULTADOS
249       public static int formatoResultado;
250
251
252       //METODO PRINCIPAL(JAVA)
253       public static void main( String args[] ) throws Exception {
254
255         //MOSTRAR EL PRIMER MENU (MENU DE USUARIOS)
256         ventana_usuarios.setVisible(true);
257         //NO PERMITIR REDIMENSION DE LA VENTANA
258         ventana_usuarios.setResizable(false);
259
260     }
261
262 }
```

2.2 USUARIO.JAVA

```
1  //PAQUETE NECESARIO
2  package PaquetePreAnalisis;
3
4  /**
5   *
6   * @author Angel Murcia Diaz
7   */
8
9  //CLASE USUARIO
10 public class Usuario {
11
12     ///////////////////////
13     //VARIABLES PRIVADAS//
14     ///////////////////////
15
16     //NOMBRE DE USUARIO
17     private String nick_;
18
19     //CONTRASE A DE USUARIO
20     private String pass_;
21
22     //P.CONEXION DE USUARIO
23     private String pconex_;
24
25     //TIPO DE VERSION DE USUARIO
26     private boolean versionPrueba_;
27
28     ////////////////////
29     //MODIFICADORES//
30     ////////////////////
31
32     //MODIFICADCOR DE NICK
33     public void setNick (String nick) {nick_=nick;};
34
```

```
35      //MODIFICADOR DE PASS
36      public void setPass (String pass) {pass_=pass;};
37
38      ///MODIFICADOR DE PCONEX
39      public void setPconex (String pconex) {pconex_=pconex;};
40
41      //MODIFICADOR VERSION PRUEBA
42      public void setVersionPrueba (boolean versionPrueba) {versionPrueba_=
            versionPrueba;};
43
44      ////////////////
45      //OBSERVADORES//
46      ////////////////
47
48      //OBSERVADOR NICK
49      public String getNick () {return nick_;};
50
51      //OBSERVADOR PASS
52      public String getPass () {return pass_;};
53
54      //OBSERVADOR PCONEX
55      public String getPconex () {return pconex_;};
56
57      //OBSERVADOR VERSION PRUEBA
58      public boolean getVersionPrueba () {return versionPrueba_;};
59
60
61  }
```

2.3 CONTENEDOR.JAVA

```java
//PAQUETE NECESARIO
package PaquetePreAnalisis;

//IMPORT NECESARIO
import java.util.ArrayList;

/**
 *
 * @author Angel Murcia Diaz
 */

//CLASE CONTENEDOR
public class Contenedor {

    ////////////
    //VARIABLES//
    ////////////

    //VECTOR PARA ALMACENAR LA CLASE DE CADA FOTOGRAMA
    private ArrayList<Integer> clases_ = new ArrayList<>();

    //TIEMPO DEL VIDEO DONDE SE PRODUCE EL FOTOFRAME
    private double tiempo_;

    //CLASE CON MAYOR PORCENTAJE EN EL FOTOFRAME
    private int claseMayor_;

    //NOMBRE DEL FOTOFRAME
    private String nombreFile_;

    ////////////////
    //MODIFICADORES//
    ////////////////

```

```
35      //MODIFICADOR DEL VECTOR CLASES
36      public void setClases(ArrayList<Integer> clases) {clases_=clases;}
37
38      //MODIFICADOR DE TIEMPO
39      public void setTiempo(double tiempo) {tiempo_=tiempo;}
40
41      //MODIFICADOR CLASE MAYOR
42      public void setClaseMayor(int claseMayor) {claseMayor_=claseMayor;}
43
44      //MODIFICADOR NOMBRE DEL ARCHIVO
45      public void setNombreFile(String nombreFile){nombreFile_=nombreFile;};
46
47
48      /////////////////
49      //OBSERVADORES//
50      /////////////////
51
52      //OBSERVADOR DEL VECTOR CLASES
53      public ArrayList<Integer> getClases() {return clases_;}
54
55      //OBSERVADOR TIEMPO
56      public double getTiempo() {return tiempo_;}
57
58      //OBSERVADOR CLASE MAYOR
59      public int getClaseMayor() {return claseMayor_;}
60
61      //OBSERVADOR NOMBRE DEL ARCHIVO
62      public String getNombreFile(){return nombreFile_;};
63
64  }
```

Código PaqueteAnalisisVideo

3.1 Funciones.java

```
1  //INCLUSION NECESARIA DEL PAQUETE
2  package PaqueteAnalisisVideo;
3
4  //IMPORT NECESARIOS
5  import PaquetePreAnalisis.Contenedor;
6  import PaquetePreAnalisis.Principal;
7  import PaquetePreAnalisis.Principal;
8  import PaqueteVentanas.VentanaErrorParaProceso;
9  import com.xuggle.xuggler.Global;
10 import java.io.BufferedWriter;
11 import java.io.File;
12 import java.io.FileNotFoundException;
13 import java.io.FileOutputStream;
14 import java.io.FileWriter;
15 import java.io.IOException;
16 import java.io.PrintWriter;
17 import java.io.UnsupportedEncodingException;
18 import java.net.URI;
19 import java.net.URISyntaxException;
20 import java.util.ArrayList;
21 import java.util.logging.Level;
22 import java.util.logging.Logger;
23 import javax.swing.JFrame;
24 import org.apache.http.HttpEntity;
25 import org.apache.http.HttpResponse;
26 import org.apache.http.ParseException;
27 import org.apache.http.client.HttpClient;
```

```
28 import org.apache.http.client.methods.HttpPost;
29 import org.apache.http.entity.StringEntity;
30 import org.apache.http.client.utils.URIBuilder;
31 import org.apache.http.entity.FileEntity;
32 import org.apache.http.impl.client.HttpClientBuilder;
33 import org.apache.http.util.EntityUtils;
34 import org.jfree.chart.JFreeChart;
35 import org.jfree.data.category.DefaultCategoryDataset;
36 import org.json.JSONArray;
37 import org.json.JSONObject;
38 import javax.swing.JFrame;
39 import org.apache.poi.ss.usermodel.Cell;
40 import org.apache.poi.ss.usermodel.CellStyle;
41 import org.apache.poi.ss.usermodel.FillPatternType;
42 import org.apache.poi.ss.usermodel.IndexedColors;
43 import org.apache.poi.ss.usermodel.Row;
44 import org.apache.poi.ss.usermodel.Sheet;
45 import org.apache.poi.ss.usermodel.Workbook;
46 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
47 import org.jfree.chart.*;
48 import org.jfree.chart.plot.PlotOrientation;
49 import org.jfree.data.category.DefaultCategoryDataset;
50 import org.jfree.data.general.DefaultPieDataset;
51
52
53
54 /**
55  *
56  * @author Angel Murcia Diaz
57  */
58
59 //CLASE FUNCIONES
60 public class Funciones {
61
62     //CLAVE DE API FACE
63     private static String subscriptionKey = "823
```

```
        a11f5e8024d94ab8e101df6ec7c6b";
64
65      //COMO LUGAR FISICO DONDE ACTUARA API FACE (PCONEX)
66      private static String uriBase =
67      "https://westcentralus.api.cognitive.microsoft.com/face/v1.0/detect";
68
69      //ATRIBUTOS QUE SE DESEAN EXTRAER DE LA CARA
70      private static final String faceAttributes =
71      "emotion";
72
73      //PARA LA COMPROBACION DE LOS ARCHIVOS EXCEL
74      public static final Logger LOGGER = Logger.getLogger("mx.com.hash.
            newexcel.Excel00XML");
75
76      //FUNCION PARA INICIALIZAR LOS VALORES NECESARIOS PARA UN NUEVO
            ANALISIS
77      public void inicializarValoresIniciales(){
78
79          //CLAVE PARA MICROSOFT INICIALIZAR
80          subscriptionKey = Principal.user.getPass();
81
82          //COMO LUGAR FISICO DONDE ACTUARA LA faceAPI INICIALIZAR
83          uriBase = Principal.user.getPconex();
84
85          //contador anger
86          Principal.angerContador = 0;
87
88          //contador contempt
89          Principal.contemptContador = 0;
90
91          //contador disgust
92          Principal.disgustContador = 0;
93
94          //contador fear
95          Principal.fearContador= 0;
96
```

```
97          //contador happiness
98          Principal.happinessContador = 0;
99
100         //contador neutral
101         Principal.neutralContador = 0;
102
103          //contador sadness
104         Principal.sadnessContador = 0;
105
106         //contador surprise
107         Principal.surpriseContador = 0;
108
109         //contador anger
110         Principal.angerContadorMayor = 0;
111
112         //contador contempt
113         Principal.contemptContadorMayor = 0;
114
115         //contador disgust
116         Principal.disgustContadorMayor = 0;
117
118         //contador fear
119         Principal.fearContadorMayor = 0;
120
121         //contador happiness
122         Principal.happinessContadorMayor = 0;
123
124         //contador neutral
125         Principal.neutralContadorMayor = 0;
126
127          //contador sadness
128         Principal.sadnessContadorMayor = 0;
129
130         //contador surprise
131         Principal.surpriseContadorMayor = 0;
132
```

```
133         //contador del total de fotos analizadas
134         Principal.nFotosAnalizadas = 0;
135 
136         //contador de no analizadas
137         Principal.noAnalizadoContador = 0;
138 
139         //CONTADORES EN PORCENTAJES
140 
141         //CONTADOR anger
142         Principal.angerContadorPorcentaje = 0.0;
143 
144         //CONTADOR contempt
145         Principal.contemptContadorPorcentaje = 0.0;
146 
147         //CONTADOR disgust
148         Principal.disgustContadorPorcentaje = 0.0;
149 
150         //CONTADOR fear
151         Principal.fearContadorPorcentaje = 0.0;
152 
153         //CONTADOR happiness
154         Principal.happinessContadorPorcentaje = 0.0;
155 
156         //CONTADOR neutral
157         Principal.neutralContadorPorcentaje = 0.0;
158 
159         //CONTADOR sadness
160         Principal.sadnessContadorPorcentaje = 0.0;
161 
162         //CONTADOR surprise
163         Principal.surpriseContadorPorcentaje = 0.0;
164 
165         ////////////////////////
166         //CONTADOR surprise ESTE ES TANTO GENERAL COMO MAYORITARIO
167 
168         //umbral configurado por el usuario
```

```
169         Principal.umbralSentimiento = Principal.valorUsuarioUmbral ;
170
171         //segundos entre los frames del video
172         Principal.segundosEntreFrames = Principal.valorUsuarioSegundosFrame
                ;
173
174         //ruta para guardar el archivo
175         Principal.rutaGuardar = Principal.valorUsuarioRuta ;
176
177         //para saber si hay ruta
178         Principal.hayRutaUsuario = Principal.hayRutaConfirmada;
179
180         //para conservar o no fotogramas y datos
181         Principal.conservar = Principal.valorUsuarioConservar;
182
183        //Indicar de la existencia de un error durante un an lisis
184         Principal.errorDuranteAnalisis=false;
185
186         //Indica el formato en el que se muestra las soluciones
187         //1 total
188         //2 porcentaje
189         Principal.formatoResultado = 1;
190
191         //PARA PODER ANALIZAR VARIOS VIDEOS AL ABRIR UNA VEZ LA INTERFAZ
192
193         //Tiempo al final de escrbir cada frame
194         DecodeAndCaptureFrames.mLastPtsWrite = Global.NO_PTS;
195
196         //Indice de flujo de videos, necesario
197         DecodeAndCaptureFrames.mVideoStreamIndex = -1;
198
199         //nuevo
200         Principal.matarHilo=false;
201
202     }
203
```

```
204     //FUNCION PARA IMPRIMIR LOS SEGUNDOS Y LA FOTO CREADA
205     public void funcionImprimir(double segundos, File file){
206
207         System.out.printf("
                ----------------------------------------------------------------------\
                n");
208         System.out.printf("-->(SEGUNDO DE LA FOTO / ARCHIVO CREADO) %6.3f /
                 %s\n",segundos, file);
209
210     }
211
212     //FUNCION PARA ANALIZAR CADA UNO DE LOS ARCHIVOS
213     //los segundos para guardar mas facilemente
214     public void face (File file, double seconds){
215
216         //CREACION DEL CLIENTE, necesaria para el analisis
217         HttpClient httpclient = HttpClientBuilder.create().build();
218
219         //TRY NECESARIO, para evitar errores
220         try
221         {
222             //CREACION DEL OBJETO URI, necesario
223             URIBuilder builder = null;
224                 try {
225                     builder = new URIBuilder(uriBase);
226                 } catch (URISyntaxException ex) {
227                     Logger.getLogger(Funciones.class.getName()).log(Level.
                            SEVERE, null, ex);
228                 }
229
230             //APLICACION DE PARAMETROS, del analisis
231             builder.setParameter("returnFaceId", "true");
232             builder.setParameter("returnFaceLandmarks", "false");
233             builder.setParameter("returnFaceAttributes", faceAttributes);
234
235             //PREPARACION DE LA LLAMADA A LA APIFACE
```

```
236             URI uri = null;
237                 try {
238                     uri = builder.build();
239                 } catch (URISyntaxException ex) {
240                     Logger.getLogger(Funciones.class.getName()).log(Level.
                        SEVERE, null, ex);
241                 }
242             //OBEJTO NECESARIO, que contiene toda la configuracion
243             HttpPost request = new HttpPost(uri);
244 
245             //APLICACION DE MAS PARAMETROS NECESARIOS, como la clave
246             request.setHeader("Content-Type", "application/octet-stream");
247             request.setHeader("Ocp-Apim-Subscription-Key", subscriptionKey)
                ;
248 
249             //CREACION DEL FILEENTITY Y ENTRADA COMO PARAMETRO EN EL
                REQUEST(objeto con los datos necesarios para analisis)
250             FileEntity reqEntity = new FileEntity(file);
251             request.setEntity(reqEntity);
252 
253             //EJECUCION O PETICION A LA APIFACE
254             HttpResponse response = null;
255                 try {
256                     response = httpclient.execute(request);
257                 } catch (IOException ex) {
258                     Logger.getLogger(Funciones.class.getName()).log(Level.
                        SEVERE, null, ex);
259                 }
260             HttpEntity entity = response.getEntity();
261 
262             //COMPROBACION, para ver si devuelve algo la llamada
263             if (entity != null)
264                 {
265                     //CABEZERA PARA MOSTRAR LA RESPUESTA DEL SERVICIO DE
                        FACEAPI.
266                     System.out.println("REST Response (respuesta de la API
```

```
267
268                 //STRING QUE CONTENDRA LA RESPUES DE LA API
269                 String jsonString = null;
270
271                 //TRY NECESARIO, INSERCION DE LA RESPUESTA EN EL STRING
                        ANTERIOR
272                 try {
273                     jsonString = EntityUtils.toString(entity).trim();
274                 } catch (IOException ex) {
275                     Logger.getLogger(Funciones.class.getName()).log(
                            Level.SEVERE, null, ex);
276                 } catch (ParseException ex) {
277                     Logger.getLogger(Funciones.class.getName()).log(
                            Level.SEVERE, null, ex);
278                 }
279                 if (jsonString.charAt(0) == '[') {
280                     JSONArray jsonArray = new JSONArray(jsonString);
281                     System.out.println(jsonArray.toString(2));
282                     System.out.println("1");
283                 }else if (jsonString.charAt(0) == '{') {
284                     JSONObject jsonObject = new JSONObject(jsonString);
285                     System.out.println(jsonObject.toString(2));
286                     System.out.println("2");
287                 } else {
288                     System.out.println(jsonString);
289                     System.out.println("3");
290                 }
291
292
293                 //LLAMA A LA FUNCION CLASE, para ver a que clase
                        pertenece cada foto
294                 clase (jsonString,seconds,file);
295
296             }
297         }
```

```
298             catch (Exception e)
299             {
300                 // Display error message.
301                 System.out.println(e.getMessage());
302             }
303 
304         }
305 
306         //FUNCION CLASE PARA VER EL PORCENTAJE MAS ALTO Y DECIR A LA CLASE
                QUE PERTENECE
307         //segundos para guardarlso facilmente en el vector
308         public void clase (String jsonString, double seconds, File file){
309 
310             //STRING QUE CONTIENE EL NOMBRE DEL ARCHIVO PARA PODER IR
                    ALMACENANDOLO JUNTO CON SUS RESULTADOS
311             String nombreFile = file.getName();
312 
313             if(jsonString.length() == 2){
314 
315                 //STRING AUXILIAR PARA IR EXTRAYENDO LOS POCENTAJES DE CADA
                        CLASE, se corta la cadena original
316                 String cortadoComprobacion=jsonString.substring(jsonString.
                        lastIndexOf("["), jsonString.length());
317                 System.out.println(cortadoComprobacion);
318 
319                 //COMPROBACION DE QUE ESTA ANALIZANDO EL FOTOGRMAA
320                 if("[]".equals(cortadoComprobacion)){
321 
322                     //System.out.println("holiiiiiiiiiiiii");
323                     Principal.noAnalizadoContador++;
324 
325                     //AUMENTAN LAS FOTOS ANALIZADAS
326                     Principal.nFotosAnalizadas++;
327 
328                     //VECTOR AUXILIAR NECESARIO PARA INTRODUCIR UN VECTOR
                            DE CLASES A INSERTAR
```

```
329                 ArrayList<Integer> vectorAux = new ArrayList<> ();
330
331                 vectorAux.add(8);
332
333                 //GUARDADO EN VECTOR
334                 //meto manualmente la clase8:No analizado
335                 insertarElementoVector(8,vectorAux,seconds, nombreFile)
                        ;
336
337             }
338
339         }else{
340
341             //STRING AUXILIAR PARA IR EXTRAYENDO LOS POCENTAJES DE CADA
                    CLASE, se corta la cadena original
342             String cortadoError=jsonString.substring(2, 7);
343             System.out.println("|"+cortadoError+"|");
344
345
346
347            if("error".equals(cortadoError)){
348
349                 System.out.println("Error locacalizado en el proceso de
                        an lisis");
350                 //cerrar ventana povisional
351                 //abrir menu principal
352                 //ventana error con fallo
353
354                 if(Principal.errorDuranteAnalisis == false){
355
356                     //ERROR DESEADO
357                     Principal.error = "ERROR EN DURANTE EL AN LISIS";
358
359                     //CREACI N VENTANA DE ERROR
360                     VentanaErrorParaProceso ventana_error_para_proceso
                            = new VentanaErrorParaProceso();
```

```
361
362                     //PONER VISIBLE LA VENTANA
363                     ventana_error_para_proceso.setVisible(true);
364
365                     //PONER LA VENTANA DE TAMA O FIJO
366                     ventana_error_para_proceso.setResizable(false);
367
368                     Principal.errorDuranteAnalisis=true;
369
370                 }
371              }
372          }
373
374          //STRING AUXILIAR PARA IR EXTRAYENDO LOS POCENTAJES DE CADA
                CLASE, se corta la cadena original
375          String cortado=jsonString.substring(jsonString.lastIndexOf("\"
                anger\":"), jsonString.lastIndexOf("}}}]"));
376
377          //OTRO STRING AUXILIAR EN EL CUAL SE IRA CORTANDO EL TROZO
                CORTADO
378          String aux;
379
380          //PARA OBTENER EL PROCENTAJE DE ANGER
381          aux=cortado.substring(8, cortado.lastIndexOf(",\"contempt\"") )
                ;
382
383          //VARIABLE, para almacenar el porcentaje de anger
384          float angerPorcentaje = Float.parseFloat(aux);
385
386          //PARA OBTENER EL PROCENTAJE DE CONTEMPT
387          aux=cortado.substring(cortado.lastIndexOf(",\"contempt\"")+12,
                cortado.lastIndexOf(",\"disgust\"") );
388
389          //VARIABLE, para almacenar el porcentaje de contempt
390          float contemptPorcentaje = Float.parseFloat(aux);
391
```

```
392             //PARA OBTENER EL PROCENTAJE DE DISGUST
393             aux=cortado.substring(cortado.lastIndexOf(",\"disgust\"")+11,
                    cortado.lastIndexOf(",\"fear\"") );
394
395             //VARIABLE, para almacenar el porcentaje de disgust
396             float disgustPorcentaje = Float.parseFloat(aux);
397
398             //PARA OBTENER EL PROCENTAJE DE FEAR
399             aux=cortado.substring(cortado.lastIndexOf(",\"fear\"")+8,
                    cortado.lastIndexOf(",\"happiness\"") );
400
401             //VARIABLE, para almacenar el porcentaje de fear
402             float fearPorcentaje = Float.parseFloat(aux);
403
404             //PARA OBTENER EL PROCENTAJE DE HAPPINESS
405             aux=cortado.substring(cortado.lastIndexOf(",\"happiness\"")+13,
                     cortado.lastIndexOf(",\"neutral\"") );
406
407             //VARIABLE, para almacenar el porcentaje de happiness
408             float happinessPorcentaje = Float.parseFloat(aux);
409
410             //PARA OBTENER EL PROCENTAJE DE NEUTRAL
411             aux=cortado.substring(cortado.lastIndexOf(",\"neutral\"")+11,
                    cortado.lastIndexOf(",\"sadness\"") );
412
413             //VARIABLE, para almacenar el porcentaje de neutral
414             float neutralPorcentaje = Float.parseFloat(aux);
415
416             //PARA OBTENER EL PROCENTAJE DE SADNESS
417             aux=cortado.substring(cortado.lastIndexOf(",\"sadness\"")+11,
                    cortado.lastIndexOf(",\"surprise\"") );
418
419             //VARIABLE, para almacenar el porcentaje de sadness
420             float sadnessPorcentaje = Float.parseFloat(aux);
421
422             //PARA OBTENER EL PROCENTAJE DE SURPRISE
```

```
423             aux=cortado.substring(cortado.lastIndexOf(",\"surprise\"")+12,
                    cortado.length() );
424
425             //VARIABLE, para almacenar el porcentaje de surprise
426             float surprisePorcentaje = Float.parseFloat(aux);
427
428             //RESUMEN DE CADA FOTO CON SUS RESPECTIVOS PORCENTAJES
429
430             System.out.println("----------------RESUMEN DE PORCENTAJES
                    ------------");
431
432             System.out.print("anger:");
433             System.out.println(angerPorcentaje);
434
435             System.out.print("contempt: ");
436             System.out.println(contemptPorcentaje);
437
438             System.out.print("disgust: ");
439             System.out.println(disgustPorcentaje);
440
441             System.out.print("fear: ");
442             System.out.println(fearPorcentaje);
443
444             System.out.print("happiness: ");
445             System.out.println(happinessPorcentaje);
446
447             System.out.print("neutral: ");
448             System.out.println(neutralPorcentaje);
449
450             System.out.print("sadness: ");
451             System.out.println(sadnessPorcentaje);
452
453             System.out.print("surprise: ");
454             System.out.println(surprisePorcentaje);
455
456             ////////////////////////////////////////////////////////////
```

```
457
458             //VARIABLE PARA ALMACENAR LA CLASE A LA QUE PERTENECE CADA FOTO
459             int claseMayor=0;
460
461             //VARIABLE que guardara el porcentaje mayor
462             float mayor=angerPorcentaje;
463
464             //COMPROBACIONES PARA VER EL PORCENTAJE MAYOR (por tanto a la
                    clase que pertenece)
465
466             if(mayor<contemptPorcentaje){
467
468                 mayor=contemptPorcentaje;
469                 claseMayor=1;
470
471             }
472
473             if(mayor<disgustPorcentaje){
474
475                 mayor=disgustPorcentaje;
476                 claseMayor=2;
477
478             }
479
480             if(mayor<fearPorcentaje){
481
482                 mayor=fearPorcentaje;
483                 claseMayor=3;
484
485             }
486
487             if(mayor<happinessPorcentaje){
488
489                 mayor=happinessPorcentaje;
490                 claseMayor=4;
491
```

```
492             }
493
494             if(mayor<neutralPorcentaje){
495
496                 mayor=neutralPorcentaje;
497                 claseMayor=5;
498
499             }
500
501             if(mayor<sadnessPorcentaje){
502
503                 mayor=sadnessPorcentaje;
504                 claseMayor=6;
505
506             }
507
508             if(mayor<surprisePorcentaje){
509
510                 mayor=surprisePorcentaje;
511                 claseMayor=7;
512
513             }
514
515             //SUMA EN EL CONTADOR DE LA VARIABLE CORRECTA DEPENDIENDO A LA
                    CLASE QUE PERTENEZCA LA FOTO
516             System.out.print("1.Clase mayor: ");
517             System.out.println(claseMayor);
518
519             // declaraci n de switch
520             switch (claseMayor) {
521                 // declaraci n case
522                 // los valores deben ser del mismo tipo de la expresi n
523                 case 0:
524                     // Declaraciones
525                     Principal.angerContadorMayor++;
526                     break; // break es opcional
```

```
527
528                 case 1:
529                     // Declaraciones
530                     Principal.contemptContadorMayor++;
531                     break; // break es opcional
532
533                 case 2:
534                     // Declaraciones
535                     Principal.disgustContadorMayor++;
536                     break; // break es opcional
537
538                 case 3:
539                     // Declaraciones
540                     Principal.fearContadorMayor++;
541                     break; // break es opcional
542
543                 case 4:
544                     // Declaraciones
545                     Principal.happinessContadorMayor++;
546                     break; // break es opcional
547
548                 case 5:
549                     // Declaraciones
550                     Principal.neutralContadorMayor++;
551                     break; // break es opcional
552
553                 case 6:
554                     // Declaraciones
555                     Principal.sadnessContadorMayor++;
556                     break;
557
558                 case 7:
559                     // Declaraciones
560                     Principal.surpriseContadorMayor++;
561                     break; // break es opcional
562
```

```
563                 // Podemos tener cualquier n mero de declaraciones de
                        casos o case
564                 // debajo se encuentra la declaraci n predeterminada, que
                        se usa cuando ninguno de los casos es verdadero.
565                 // No se necesita descanso en el case default
566                 default:
567                 // Declaraciones
568             }
569
570             //VECTOR AUXILIAR PARA ALMACENAR LAS CLASES A LAS QUE PERTENECE
571             ArrayList<Integer> clases = new ArrayList<>();
572
573             //ANALISIS DE LAS DISTINTAS CATEGORIAS QUE PERTENECE (LAS QUE
                    SUPERAN EL UMBRAL)
574
575             if(angerPorcentaje >= Principal.umbralSentimiento){
576
577                 clases.add(0);
578                 Principal.angerContador++;
579
580
581             }
582
583             if(contemptPorcentaje >= Principal.umbralSentimiento){
584
585                 clases.add(1);
586                 Principal.contemptContador++;
587
588
589             }
590
591             if(disgustPorcentaje >= Principal.umbralSentimiento){
592
593                  clases.add(2);
594                  Principal.disgustContador++;
595
```

```
596
597             }
598
599             if(fearPorcentaje >= Principal.umbralSentimiento){
600
601                  clases.add(3);
602                  Principal.fearContador++;
603
604
605             }
606
607             if(happinessPorcentaje >= Principal.umbralSentimiento){
608
609                 clases.add(4);
610                 Principal.happinessContador++;
611
612
613             }
614
615             if(neutralPorcentaje >= Principal.umbralSentimiento ){
616
617                 clases.add(5);
618                 Principal.neutralContador++;
619
620
621             }
622
623             if(sadnessPorcentaje >= Principal.umbralSentimiento){
624
625                 clases.add(6);
626                 Principal.sadnessContador++;
627
628
629             }
630
631             if(surprisePorcentaje >= Principal.umbralSentimiento){
```

```
632
633                 clases.add(7);
634                 Principal.surpriseContador++;
635
636
637             }
638
639
640             //GUARDADO EN VECTOR
641             insertarElementoVector(claseMayor, clases, seconds, nombreFile)
                    ;
642
643             //IMPRESION DE LA CLASES A LA QUE PERTENECE LA FOTO
644             System.out.print("2.Clases existentes: ");
645
646             for(int i=0; i<clases.size(); i++){
647
648                 System.out.print(clases.get(i)+ "  ");
649
650             }
651             System.out.println("");
652
653                             //AUMENTAN LAS FOTOS ANALIZADAS
654                   Principal.nFotosAnalizadas++;
655
656             //imprimir ("DESPUES DE ANALIZAR UNA FOTO:", "Mayoritario");
657             //imprimir ("DESPUES DE ANALIZAR UNA FOTO:", "Generales");
658
659     }
660
661     //FUNCION PARA IMPRIMIR LOS RESULTADOS
662     public void imprimir (String descripcion, String tipo){
663
664
665         System.out.println("------------------------RESUMEN
                ------------------------");
```

```
666         System.out.println(descripcion);
667         System.out.println("------------");
668 
669         if("Mayoritario".equals(tipo)){
670 
671             System.out.println("------------");
672             System.out.println(tipo);
673             System.out.println("------------");
674 
675             System.out.print("anger:");
676             System.out.println(Principal.angerContadorMayor);
677 
678             System.out.print("contempt: ");
679             System.out.println(Principal.contemptContadorMayor);
680 
681             System.out.print("disgust: ");
682             System.out.println(Principal.disgustContadorMayor);
683 
684             System.out.print("fear: ");
685             System.out.println(Principal.fearContadorMayor);
686 
687             System.out.print("happiness: ");
688             System.out.println(Principal.happinessContadorMayor);
689 
690             System.out.print("neutral: ");
691             System.out.println(Principal.neutralContadorMayor);
692 
693             System.out.print("sadness: ");
694             System.out.println(Principal.sadnessContadorMayor);
695 
696             System.out.print("surprise: ");
697             System.out.println(Principal.surpriseContadorMayor);
698 
699             System.out.print("Fotos NO analizadas: ");
700             System.out.println(Principal.noAnalizadoContador);
701 
```

```
702             System.out.print("Fotos analizadas: ");
703             System.out.println(Principal.nFotosAnalizadas);
704 
705         }else if("Porcentajes".equals(tipo)){
706 
707             System.out.println("------------");
708             System.out.println(tipo);
709             System.out.println("------------");
710 
711             System.out.print("anger:");
712             System.out.println(Principal.angerContadorPorcentaje);
713 
714             System.out.print("contempt: ");
715             System.out.println(Principal.contemptContadorPorcentaje);
716 
717             System.out.print("disgust: ");
718             System.out.println(Principal.disgustContadorPorcentaje);
719 
720             System.out.print("fear: ");
721             System.out.println(Principal.fearContadorPorcentaje);
722 
723             System.out.print("happiness: ");
724             System.out.println(Principal.happinessContadorPorcentaje);
725 
726             System.out.print("neutral: ");
727             System.out.println(Principal.neutralContadorPorcentaje);
728 
729             System.out.print("sadness: ");
730             System.out.println(Principal.sadnessContadorPorcentaje);
731 
732             System.out.print("surprise: ");
733             System.out.println(Principal.surpriseContadorPorcentaje);
734 
735             System.out.print("Fotos NO analizadas: ");
736             System.out.println(Principal.noAnalizadoContador);
737 
```

```
738             System.out.print("Fotos analizadas: ");
739             System.out.println(Principal.nFotosAnalizadas);
740 
741 
742         }else{
743 
744             System.out.println("
                    ----------------------------------------------------------");
745             System.out.println(tipo);
746             System.out.println("
                    ----------------------------------------------------------");
747 
748             System.out.print("anger:");
749             System.out.println(Principal.angerContador);
750 
751             System.out.print("contempt: ");
752             System.out.println(Principal.contemptContador);
753 
754             System.out.print("disgust: ");
755             System.out.println(Principal.disgustContador);
756 
757             System.out.print("fear: ");
758             System.out.println(Principal.fearContador);
759 
760             System.out.print("happiness: ");
761             System.out.println(Principal.happinessContador);
762 
763             System.out.print("neutral: ");
764             System.out.println(Principal.neutralContador);
765 
766             System.out.print("sadness: ");
767             System.out.println(Principal.sadnessContador);
768 
769             System.out.print("surprise: ");
770             System.out.println(Principal.surpriseContador);
771 
```

```
772             System.out.print("Fotos NO analizadas: ");
773             System.out.println(Principal.noAnalizadoContador);
774 
775             System.out.print("Fotos analizadas: ");
776             System.out.println(Principal.nFotosAnalizadas);
777 
778 
779 
780         }
781 
782 
783     }
784 
785     ////////////////////////////////////////////////
786     //FUNCIONES DE CONTENEDOR
787 
788     //FUNCION PARA A ADIR UN ELEMENTO AL VECTOR
789     public void insertarElementoVector(int claseMayor, ArrayList<Integer>
            clases, double tiempo, String nombreFile){
790 
791         //CREACION DEL CONTENEDOR AUXILIAR
792         Contenedor aux = new Contenedor();
793 
794         //RELLENAR CLASE EN AUXILIAR
795         aux.setClases(clases);
796 
797         //RELLENAR CLASE MAYOR EN AUXILIAR
798         aux.setClaseMayor(claseMayor);
799 
800         //RELLENAR TIEMPO EN AUXILIAR
801         aux.setTiempo(tiempo);
802 
803         //RELLENAR EL NOMBRE DLE ARCHIVO EN EL AUXILIAR
804         aux.setNombreFile(nombreFile);
805 
806         //A ADIR AL VECTOR
```

```
807             Principal.contenedorFrames.add(aux);
808
809         }
810
811         public void listarContenedor(){
812
813             //PARA RECOGER TODOS LOS ELEMENTOS DEL VECTOR O CONTENEDOR
814             for( int i=0 ; i<Principal.contenedorFrames.size() ; i++ ){
815
816                 //IMPRESION DE ELEMENTOS
817                 System.out.print("El frame analizado: ");
818                 System.out.println(i);
819
820                 System.out.print("El archivo se llama: ");
821                 System.out.println(Principal.contenedorFrames.get(i).
                        getNombreFile());
822
823                 System.out.print("pertenece al segundo: ");
824                 System.out.println(Principal.contenedorFrames.get(i).getTiempo
                        ());
825
826                 //PROCEDIMIENTO PARA IMPRIMIR CADENA EN LUGAR DEL NUMERO, mas
                        bonito
827                 int clase = Principal.contenedorFrames.get(i).getClaseMayor();
828                 String claseCadena = claseEnteroACadena(clase);
829
830                 System.out.print("y es de la clase (MAYORITARIA): ");
831                 System.out.println(claseCadena);
832
833
834                 //IMPRESION DE TODAS LAS CLASES PERTENECIENTES
835                 ArrayList<Integer> clases = new ArrayList<>();
836                 clases = Principal.contenedorFrames.get(i).getClases();
837
838                 System.out.print("Las clases a la que pertenece la foto son: ")
                        ;
```

```
839
840             for(int j=0; j<clases.size(); j++){
841
842                 clase = clases.get(j);
843
844                 claseCadena = claseEnteroACadena(clase);
845
846                 System.out.println(claseCadena + "    ");
847
848
849             }
850
851             System.out.println("");
852         }
853
854     }
855
856     public String claseEnteroACadena(int clase){
857
858         String claseCadena = null;
859
860             if(clase == 0){
861
862                 claseCadena="anger";
863
864             }
865
866             if(clase == 1){
867
868                 claseCadena="contempt";
869
870             }
871
872             if(clase == 2){
873
874                 claseCadena="disgust";
```

```
875
876             }
877
878             if(clase == 3){
879
880                 claseCadena="fear";
881
882             }
883
884             if(clase == 4){
885
886                 claseCadena="happiness";
887
888             }
889
890             if(clase == 5){
891
892                claseCadena="neutral";
893
894             }
895
896             if(clase == 6){
897
898                 claseCadena="sadness";
899
900             }
901
902             if(clase == 7){
903
904                 claseCadena="surprise";
905
906             }
907
908             if(clase == 8){
909
910                 claseCadena="no analizado";
```

```
911
912                 }
913
914             return claseCadena;
915
916         }
917
918
919
920
921 }
```

3.2 HiloProcesoPrincipal.java

```
1  //PAQUETE NECESARIO
2  package PaqueteAnalisisVideo;
3
4  //IMPORT NECESARIOS
5  import PaqueteAnalisisVideo.Funciones;
6  import PaquetePostAnalisis.FuncionesResultados;
7  import PaquetePreAnalisis.Principal;
8
9  //CLASE HILOPROCESO PRINCIPAL (HILO)
10 public class HiloProcesoPrincipal extends Thread {
11
12     //CONSTRUCTOR
13     public HiloProcesoPrincipal(String nombre) {
14         super(nombre);
15     }
16
17     @Override
18     public void run(){
19
20         Funciones aux = new Funciones();
21         //VARIABLE PARA MEDIR EL TIEMPO
22         long inicio = System.currentTimeMillis();
23
24         System.out.print("umbralSentimiento: ");
25         System.out.println(Principal.umbralSentimiento);
26
27         System.out.print("segundosEntreFrames: ");
28         System.out.println(Principal.segundosEntreFrames);
29
30         System.out.print("rutaGuardar: ");
31         System.out.println(Principal.rutaGuardar);
32
33         System.out.print("conservar: ");
34         System.out.println(Principal.conservar);
```

```
35
36          System.out.print("--------------------------------");
37          System.out.print("------COMIENZO DEL ANALISIS-----");
38          System.out.print("--------------------------------");
39
40          DecodeAndCaptureFrames decode = new DecodeAndCaptureFrames(
                Principal.videoFile);
41
42          boolean comprobar = decode.getMatarHilo();
43
44          if(comprobar==false){
45
46              //VARIABLE PARA MEDIR EL TIEMPO
47              long fin = System.currentTimeMillis();
48
49              //CALCULO DEL TIEMPO UTILIZANDO LAS VARIABLES ANTERIORES
50              double tiempo = (double) ((fin - inicio)/1000);
51
52              //IMPRESION FINAL DE LOS RESULTADOS DE LA CLASE MAYOR
53              aux.imprimir ("FINALES", "Mayoritario");
54
55              //IMPRESION FINAL DE LOS RESULTADOS GENERALES
56              aux.imprimir ("FINALES", "Generales");
57
58              //IMPRESION DE VARIABLES, de tiempo completo
59              System.out.println("Esta proceso ha tardado:" + tiempo + "
                    segundos");
60
61              //PONER LA VENTANA VISIBLE
62              // Principal.ventana_principal.setVisible(true);
63              Principal.ventana_seguimiento.setVisible(false);
64
65              Principal.ventana_final.setVisible(true);
66              Principal.ventana_final.setResizable(false);
67
68              if(Principal.conservar == true){
```

```
69
70                      new FuncionesResultados().crearDatasetPorFotograma();
71
72                  }
73              }
74          }
75  }
```

3.3 DecodeAndFrames.java

```
1  //PAQUETE NECESARIO
2  package PaqueteAnalisisVideo;
3
4  //IMPORT NECESARIOS
5  import PaqueteAnalisisVideo.Funciones;
6  import PaquetePreAnalisis.Principal;
7  import PaquetePreAnalisis.Principal;
8  import javax.imageio.ImageIO;
9  import java.io.File;
10 import java.awt.image.BufferedImage;
11 import com.xuggle.mediatool.IMediaReader;
12 import com.xuggle.mediatool.MediaListenerAdapter;
13 import com.xuggle.mediatool.ToolFactory;
14 import com.xuggle.mediatool.event.IVideoPictureEvent;
15 import com.xuggle.xuggler.Global;
16 import java.awt.Desktop;
17 import java.util.Scanner;
18 import java.util.logging.Level;
19 import java.util.logging.Logger;
20
21
22 //CLASE DECODEANDFRAMES
23 public class DecodeAndCaptureFrames extends MediaListenerAdapter{
24
25     //NUMERO DE SEGUNDOS entre cada frame a capturar esta a 2
26     public static final double SECONDS_BETWEEN_FRAMES = Principal.
           segundosEntreFrames;
27
28     //NUMERO DE MICROSEGUNDOS entre cada frame, puesto que la libreria
           trabaja con este dato
29     public static final long MICRO_SECONDS_BETWEEN_FRAMES =
30     (long)(Global.DEFAULT_PTS_PER_SECOND * SECONDS_BETWEEN_FRAMES);
31
32     //TIEMPO AL FINAL DE ESCRIBIR CADA FRAME
```

```
33      public static long mLastPtsWrite = Global.NO_PTS;
34
35      //INDICE DE FLUJO, NECESARIOS PARA ANALIZAR UN VIDEO
36      public static int mVideoStreamIndex = -1;
37
38      //VARIABLE PARA MATAR EL HILO
39      private boolean matarHilo = false;
40
41      //FUNCION PARA FRAGMENTAR EL VIDEO EN FRAMES, es decir fotos
42      //filename nombre de los ficheros de imagen a crear
43      public DecodeAndCaptureFrames(String filename){
44
45          //CREACION DEL OBJETO MEDIA, para que posteriormente se pueda
                trabajar con el video
46          IMediaReader reader = ToolFactory.makeReader(filename);
47
48          //CONVERSION de la imagen a un espacio de BGR
49          reader.setBufferedImageTypeToGenerate(BufferedImage.TYPE_3BYTE_BGR)
                ;
50
51          //SE DECLARA COMO OLLENTE
52          reader.addListener(this);
53
54          //AQUI LEE EL CONTENIDO DE LOS ARCHIVOS
55          //AQUI ESTA A LA ESPERA DE LOS EVENTOS SUCESIDOS EN LA OTRA FUNCION
                (espera de eventos)
56          while (reader.readPacket() == null){
57
58              if(matarHilo==false){
59
60                  do {} while(false);
61
62              }else{
63
64                  return;
65
```

```
66              }
67
68          }
69
70      }
71
72      //FUNCION QUE A TRAVES DEL VIDEO GRABA TANTAS IMAGENES COMO SE DESEE
73      public void onVideoPicture(IVideoPictureEvent event)
74      {
75          //TRY NECESARIO
76          try
77          {
78              if(Principal.matarHilo==true){
79
80                  setMatarHilo(true);
81
82                  return;
83              }
84
85              //para que no se siga ejecutando si el usuario quiere terminar
86              if(getMatarHilo()==false){
87
88                  //PARA EVITAR LLEGAR A 20 EN EL MINUTO
89                  if(Principal.veces >= 19 && Principal.versionPrueba == true
                        ){
90
91                      System.out.println("PAUSA");
92                      System.out.print("Fotos analizadas: ");
93                      System.out.println(Principal.nFotosAnalizadas);
94
95                      Principal.estaEnPausa = true;
96
97                      if(Principal.versionPrueba == true){
98                      funcionPausa(60000);
99                      //try{
100                     // Thread.sleep(60000);
```

```
101                     //}catch(InterruptedException e ) {
102                     // }
103                     }
104
105                     Principal.estaEnPausa = false;
106                     Principal.veces=0;
107                 }
108
109                 //SE COMPRUEBAN LOS EVENTOS CAPTURADOS
110                 if (event.getStreamIndex() != mVideoStreamIndex)
111                 {
112                     if (-1 == mVideoStreamIndex)
113                       mVideoStreamIndex = event.getStreamIndex();
114
115                     else
116                       return;
117                 }
118
119                 //SE COMPRUEBA EL TIEMPO PARA EL SIGUIENTE
120                 if (mLastPtsWrite == Global.NO_PTS)
121
122                   mLastPtsWrite = event.getTimeStamp() -
                        MICRO_SECONDS_BETWEEN_FRAMES;
123
124                 //SE COMPRUEBA SI HAY TIEMPO PARA EL SIGUIENTE
125                 if (event.getTimeStamp() - mLastPtsWrite >=
                    MICRO_SECONDS_BETWEEN_FRAMES)
126                 {
127                     //Nombre del archivo Temporal a crear posteriormente
128                     String savedFile = "fotograma";
129
130                     //Ruta para la creacion del fichero (Temporal)
131                     //nombreFichero, extension, directorioParaGuardar
132                     // File file = File.createTempFile(savedFile, ".png",
                           new File("C:\\Users\\angel\\Desktop\\prueba"));
133                     File file = File.createTempFile(savedFile, ".png", new
```

```
                File(Principal.rutaGuardar));
134
135             //Creacion del fichero (Temporal)
136             ImageIO.write(event.getImage(), "png", file);
137
138             //Impresion por consola de los archivos creados a
                    traves del video
139             double seconds = ((double)event.getTimeStamp())
140               / Global.DEFAULT_PTS_PER_SECOND;
141
142             //System.out.printf("at elapsed time of %6.3f seconds
                    wrote: %s\n", seconds, file);
143
144             //IMPRESION FOTO TIEMPO, de lo que se ha guardando
145             new Funciones().funcionImprimir(seconds,file);
146
147             //SE CARGA EL ARCHIVO EN LA FUNCION FACE, para realizar
                     el analisis
148             System.out.print("Analizando Frame: ");
149             System.out.println(file.getName());
150
151             new Funciones().face(file,seconds);
152
153             //Eliminar al acabar el programa, LOS ARCHIVOS
                    TEMPORALES
154             if(Principal.conservar == false){
155                 file.delete();
156                 file.deleteOnExit();
157                 System.out.println("archivo eliminado del sistema")
                        ;
158             }
159
160             //AUMENTA LA VARIABLE NUMERO DE VECES, para las
                    versiuones de prueba de faceAPI
161             Principal.veces++;
162
```

```
163                     //new Funciones().imprimir("principal--_>", "
                            Mayoritario");
164
165                     new Funciones().imprimir("principal--_>", "General");
166
167                     //ACTUALIZA LA ULTIMA ESCRITURA, para la lectura del
                            video
168                     mLastPtsWrite += MICRO_SECONDS_BETWEEN_FRAMES;
169
170                 }
171
172             }
173
174         }
175         catch (Exception e)
176         {
177           e.printStackTrace();
178         }
179     }
180
181
182     //FUNCION PARA PAUSAR EL PROGRAMA
183     //numero tiempo en microsegundos
184     public void funcionPausa(int numero){
185
186         try {
187             Thread.sleep(numero);
188         } catch (InterruptedException ex) {
189             Logger.getLogger(DecodeAndCaptureFrames.class.getName()).log(
                    Level.SEVERE, null, ex);
190         }
191
192     }
193
194     //FUNCION OBSERVADOR DE MATAR HILO
195     public boolean getMatarHilo(){
```

```
196
197
198             return matarHilo;
199
200         }
201
202         //FUNCION MODIFICADOR DE MATAR HILO
203         public void setMatarHilo(boolean activar){
204
205             matarHilo = activar;
206
207         }
208
209
210
211     }
```

Código PaquetePostAnalisis

4.1 FuncionesResultados.java

```
1  //PAQUETE NECESARIO
2  package PaquetePostAnalisis;
3
4  //IMPORT NECESARIOS
5  import PaqueteAnalisisVideo.Funciones;
6  import PaquetePreAnalisis.Principal;
7  import java.io.File;
8  import java.io.FileInputStream;
9  import java.io.FileNotFoundException;
10 import java.io.FileOutputStream;
11 import java.io.FileWriter;
12 import java.io.IOException;
13 import java.io.PrintWriter;
14 import java.nio.file.Files;
15 import java.util.ArrayList;
16 import java.util.Iterator;
17 import java.util.logging.Level;
18 import org.apache.poi.ss.usermodel.Cell;
19 import org.apache.poi.ss.usermodel.CellStyle;
20 import org.apache.poi.ss.usermodel.DateUtil;
21 import org.apache.poi.ss.usermodel.FillPatternType;
22 import org.apache.poi.ss.usermodel.IndexedColors;
23 import org.apache.poi.ss.usermodel.Row;
24 import org.apache.poi.ss.usermodel.Sheet;
25 import org.apache.poi.ss.usermodel.Workbook;
26 import org.apache.poi.xssf.usermodel.XSSFSheet;
27 import org.apache.poi.xssf.usermodel.XSSFWorkbook;
```

```
28
29 /**
30  *
31  * @author Angel Murcia Diaz
32  */
33
34 //CLASE FUNCIONESRESULTADOS
35 public class FuncionesResultados {
36
37     Funciones aux = new Funciones();
38
39     public void crearResultadosPorcentajes(){
40
41         //total util
42         double totalUtil = Principal.nFotosAnalizadas - Principal.
               noAnalizadoContador;
43
44         //evitar indeterminaciones
45         if(totalUtil!=0){
46
47             //evitar una indeterminacion
48             if(Principal.angerContador != 0){
49
50                 //CONTADOR anger
51                 Principal.angerContadorPorcentaje = (Principal.
                       angerContador * 100.0) / totalUtil;
52
53             }else{
54
55                 Principal.angerContadorPorcentaje = 0;
56
57             }
58
59             //evitar una indeterminacion
60             if(Principal.contemptContador != 0){
61
```

```
62                  //CONTADOR anger
63                  Principal.contemptContadorPorcentaje = (Principal.
                        contemptContador * 100.0) / totalUtil;
64
65              }else{
66
67                  Principal.contemptContadorPorcentaje = 0;
68
69              }
70
71              //evitar una indeterminacion
72              if(Principal.disgustContador != 0){
73
74                  //CONTADOR anger
75                  Principal.disgustContadorPorcentaje = (Principal.
                        disgustContador * 100.0) / totalUtil;
76
77              }else{
78
79                  Principal.disgustContadorPorcentaje = 0;
80
81              }
82
83              //evitar una indeterminacion
84              if(Principal.fearContador != 0){
85
86                  //CONTADOR anger
87                  Principal.fearContadorPorcentaje = (Principal.fearContador
                        * 100.0) / totalUtil;
88
89              }else{
90
91                  Principal.fearContadorPorcentaje = 0;
92
93              }
94
```

```
95              //evitar una indeterminacion
96              if(Principal.happinessContador != 0){
97
98                  //CONTADOR anger
99                  Principal.happinessContadorPorcentaje = (Principal.
                        happinessContador *100.0) / totalUtil;
100
101             }else{
102
103                 Principal.happinessContadorPorcentaje = 0;
104
105             }
106
107             //evitar una indeterminacion
108             if(Principal.neutralContador != 0){
109
110                 //CONTADOR anger
111                 Principal.neutralContadorPorcentaje = (Principal.
                        neutralContador * 100.0) / totalUtil;
112                 System.out.println("llego");
113
114             }else{
115
116                 Principal.neutralContadorPorcentaje = 0;
117
118             }
119
120             //evitar una indeterminacion
121             if(Principal.sadnessContador != 0){
122
123                 //CONTADOR anger
124                 Principal.sadnessContadorPorcentaje = (Principal.
                        sadnessContador * 100.0) / totalUtil;
125
126             }else{
127
```

```
128                 Principal.sadnessContadorPorcentaje = 0;
129
130             }
131
132             //evitar una indeterminacion
133             if(Principal.surpriseContador != 0){
134
135                 //CONTADOR anger
136                 Principal.surpriseContadorPorcentaje = (Principal.
                        surpriseContador * 100.0) / totalUtil;
137
138             }else{
139
140                 Principal.surpriseContadorPorcentaje = 0;
141
142             }
143
144         }else{
145
146             //CONTADOR anger
147             Principal.angerContadorPorcentaje = 0.0;
148
149             //CONTADOR contempt
150             Principal.contemptContadorPorcentaje = 0.0;
151
152             //CONTADOR disgust
153             Principal.disgustContadorPorcentaje = 0.0;
154
155             //CONTADOR fear
156             Principal.fearContadorPorcentaje = 0.0;
157
158             //CONTADOR happiness
159             Principal.happinessContadorPorcentaje = 0.0;
160
161             //CONTADOR neutral
162             Principal.neutralContadorPorcentaje = 0.0;
```

```
163
164             //CONTADOR sadness
165             Principal.sadnessContadorPorcentaje = 0.0;
166
167             //CONTADOR surprise
168             Principal.surpriseContadorPorcentaje = 0.0;
169         }
170
171     }
172
173
174     public void resultadosTXT(int formato) throws IOException{
175
176             System.out.println(Principal.nombreFichero);
177
178             String ruta = null;
179
180             if(formato==1){
181
182              ruta = Principal.rutaGuardar + "\\" + Principal.nombreFichero
                    + ".txt" ;
183
184             }else if (formato==2){
185
186              ruta = Principal.rutaGuardar + "\\" + Principal.nombreFichero
                    + "Porcentajes.txt" ;
187
188             }
189
190             FileWriter fichero = null;
191             PrintWriter pw = null;
192
193         try
194         {
195             fichero = new FileWriter(ruta);
196             pw = new PrintWriter(fichero);
```

```
197
198                 if(formato==1){
199
200                     pw.println("-----------------------RESUMEN EN FICHERO
                        -------------------------");
201                     pw.println("-------------");
202
203                     pw.println("anger:");
204                     pw.println(Principal.angerContador);
205
206                     pw.println("contempt: ");
207                     pw.println(Principal.contemptContador);
208
209                     pw.println("disgust: ");
210                     pw.println(Principal.disgustContador);
211
212                     pw.println("fear: ");
213                     pw.println(Principal.fearContador);
214
215                     pw.println("happiness: ");
216                     pw.println(Principal.happinessContador);
217
218                     pw.println("neutral: ");
219                     pw.println(Principal.neutralContador);
220
221                     pw.println("sadness: ");
222                     pw.println(Principal.sadnessContador);
223
224                     pw.println("surprise: ");
225                     pw.println(Principal.surpriseContador);
226
227                     pw.println("Fotos NO analizadas: ");
228                     pw.println(Principal.noAnalizadoContador);
229
230                     pw.println("Fotos analizadas: ");
231                     pw.println(Principal.nFotosAnalizadas);
```

```
232
233
234                 }else if(formato==2){
235
236                     pw.println("-----------------------RESUMEN EN FICHERO
                            EN PORCENTAJE-------------------------");
237                     pw.println("-------------");
238
239                     pw.println("Porcentaje anger:");
240                     pw.println(Principal.angerContadorPorcentaje);
241
242                     pw.println("Porcentaje contempt: ");
243                     pw.println(Principal.contemptContadorPorcentaje);
244
245                     pw.println("Porcentaje disgust: ");
246                     pw.println(Principal.disgustContadorPorcentaje);
247
248                     pw.println("Porcentaje fear: ");
249                     pw.println(Principal.fearContadorPorcentaje);
250
251                     pw.println("Porcentaje happiness: ");
252                     pw.println(Principal.happinessContadorPorcentaje);
253
254                     pw.println("Porcentaje neutral: ");
255                     pw.println(Principal.neutralContadorPorcentaje);
256
257                     pw.println("Porcentaje sadness: ");
258                     pw.println(Principal.sadnessContadorPorcentaje);
259
260                     pw.println("Porcentaje surprise: ");
261                     pw.println(Principal.surpriseContadorPorcentaje);
262
263                     pw.println("Fotos  tiles : ");
264                     pw.println(Principal.nFotosAnalizadas-Principal.
                            noAnalizadoContador);
265
```

```
266                 }
267
268         } catch (Exception e) {
269             e.printStackTrace();
270         } finally {
271             try {
272                 // Nuevamente aprovechamos el finally para
273                 // asegurarnos que se cierra el fichero.
274                 if (null != fichero)
275                     fichero.close();
276             } catch (Exception e2) {
277                 e2.printStackTrace();
278             }
279         }
280
281
282
283
284         }
285
286        //nuevo
287     public void crearDataset(int formato){
288
289         String nombreFichero = null;
290
291         if(formato==1){
292
293             nombreFichero = Principal.rutaGuardar + "\\" + Principal.
                    nombreFichero + ".xlsx" ;
294
295         }else if(formato==2){
296
297             nombreFichero = Principal.rutaGuardar + "\\" + Principal.
                    nombreFichero + "Porcentajes.xlsx" ;
298
299         }
```

```
300
301         // Creamos el archivo donde almacenaremos la hoja
302         // de calculo, recuerde usar la extension correcta,
303         // en este caso .xlsx
304         File archivo = new File(nombreFichero);
305
306
307         // Creamos el libro de trabajo de Excel formato OOXML
308         Workbook workbook = new XSSFWorkbook();
309
310         // La hoja donde pondremos los datos
311         Sheet pagina = workbook.createSheet("Resumen de las emociones del
                sujeto");
312
313         // Creamos el estilo paga las celdas del encabezado
314         CellStyle style = workbook.createCellStyle();
315         // Indicamos que tendra un fondo azul aqua
316         // con patron solido del color indicado
317         style.setFillForegroundColor(IndexedColors.BLUE.getIndex());
318         style.setFillPattern(FillPatternType.SOLID_FOREGROUND);
319
320         if(formato==1){
321
322             String[] titulos = {"anger", "contempt",
323                 "disgust", "fear", "happiness", "neutral", "sadness", "
                        surprise", "not analyzed", "total"};
324             Integer[] datos = {Principal.angerContador, Principal.
                    contemptContador, Principal.disgustContador, Principal.
                    fearContador, Principal.happinessContador,
325                  Principal.neutralContador, Principal.sadnessContador,
                        Principal.surpriseContador, Principal.
                        noAnalizadoContador, Principal.nFotosAnalizadas};
326
327             // Creamos una fila en la hoja en la posicion 0
328             Row fila = pagina.createRow(0);
329
```

```
330             // Creamos el encabezado
331             for (int i = 0; i < titulos.length; i++) {
332                 // Creamos una celda en esa fila, en la posicion
333                 // indicada por el contador del ciclo
334                 Cell celda = fila.createCell(i);
335 
336                 // Indicamos el estilo que deseamos
337                 // usar en la celda, en este caso el unico
338                 // que hemos creado
339                 //celda.setCellStyle(style);
340                 celda.setCellValue(titulos[i]);
341             }
342 
343             // Ahora creamos una fila en la posicion 1
344             fila = pagina.createRow(1);
345 
346             // Y colocamos los datos en esa fila
347             for (int i = 0; i < datos.length; i++) {
348                 // Creamos una celda en esa fila, en la
349                 // posicion indicada por el contador del ciclo
350                 Cell celda = fila.createCell(i);
351                 celda.setCellValue(datos[i]);
352                 pagina.autoSizeColumn(i);
353 
354             }
355 
356 
357         }else if(formato==2){
358 
359             String[] titulos = {"anger", "contempt", "disgust", "fear", "
                    happiness", "neutral", "sadness", "surprise", "useful"};
360             Double[] datos = {Principal.angerContadorPorcentaje, Principal.
                    contemptContadorPorcentaje,Principal.
                    disgustContadorPorcentaje, Principal.fearContadorPorcentaje,
                     Principal.happinessContadorPorcentaje, Principal.
                    neutralContadorPorcentaje, Principal.
```

```
                sadnessContadorPorcentaje, Principal.
                surpriseContadorPorcentaje, Double.valueOf(Principal.
                nFotosAnalizadas - Principal.noAnalizadoContador)};
361
362             // Creamos una fila en la hoja en la posicion 0
363             Row fila = pagina.createRow(0);
364
365             // Creamos el encabezado
366             for (int i = 0; i < titulos.length; i++) {
367                 // Creamos una celda en esa fila, en la posicion
368                 // indicada por el contador del ciclo
369                 Cell celda = fila.createCell(i);
370
371                 // Indicamos el estilo que deseamos
372                 // usar en la celda, en este caso el unico
373                 // que hemos creado
374                 //celda.setCellStyle(style);
375                 celda.setCellValue(titulos[i]);
376             }
377
378             // Ahora creamos una fila en la posicion 1
379             fila = pagina.createRow(1);
380
381             // Y colocamos los datos en esa fila
382             for (int i = 0; i < datos.length; i++) {
383                 // Creamos una celda en esa fila, en la
384                 // posicion indicada por el contador del ciclo
385                 Cell celda = fila.createCell(i);
386
387                 celda.setCellValue(datos[i]);
388
389                   pagina.autoSizeColumn(i);
390
391             }
392
393         }
```

```
394
395         // Ahora guardaremos el archivo
396         try {
397             // Creamos el flujo de salida de datos,
398             // apuntando al archivo donde queremos
399             // almacenar el libro de Excel
400             FileOutputStream salida = new FileOutputStream(archivo);
401
402             // Almacenamos el libro de
403             // Excel via ese
404             // flujo de datos
405             workbook.write(salida);
406
407             // Cerramos el libro para concluir operaciones
408             workbook.close();
409
410             aux.LOGGER.log(Level.INFO, "Archivo creado existosamente en {0}
                    ", archivo.getAbsolutePath());
411
412         } catch (FileNotFoundException ex) {
413             aux.LOGGER.log(Level.SEVERE, "Error con la creacion del archivo
                    ");
414         } catch (IOException ex) {
415             aux.LOGGER.log(Level.SEVERE, "Error de entrada/salida");
416         }
417
418
419     }
420
421
422     public void crearDatasetPorFotograma(){
423
424         String nombreFichero = Principal.rutaGuardar + "\\DatosPorFotograma
                -" + Principal.nombreFichero + ".xlsx" ;
425
426         // Creamos el archivo donde almacenaremos la hoja
```

```
427         // de calculo, recuerde usar la extension correcta,
428         // en este caso .xlsx
429         File archivo = new File(nombreFichero);
430 
431         // Creamos el libro de trabajo de Excel formato OOXML
432         Workbook workbook = new XSSFWorkbook();
433 
434         // La hoja donde pondremos los datos
435         Sheet pagina = workbook.createSheet("Resumen de las emociones del
                sujeto");
436 
437         String[] titulos = {"name", "second",
438             "emotion principal", "all emotion"};
439 
440         // Creamos una fila en la hoja en la posicion 0
441         Row filaCabezera = pagina.createRow(0);
442 
443         // Creamos el encabezado
444         for (int i = 0; i < titulos.length; i++) {
445             // Creamos una celda en esa fila, en la posicion
446             // indicada por el contador del ciclo
447             Cell celda = filaCabezera.createCell(i);
448 
449             // Indicamos el estilo que deseamos
450             // usar en la celda, en este caso el unico
451             // que hemos creado
452             //celda.setCellStyle(style);
453             celda.setCellValue(titulos[i]);
454         }
455 
456         //PARA RECOGER TODOS LOS ELEMENTOS DEL VECTOR O CONTENEDOR
457         for( int i=0 ; i<Principal.contenedorFrames.size() ; i++ ){
458 
459              // Ahora creamos una fila en la posicion 1
460             Row fila = pagina.createRow(i+1);
461 
```

```
462             // posicion indicada por el contador del ciclo
463             Cell celda = fila.createCell(0);
464 
465             //IMPRESION DE ELEMENTOS
466 
467             //System.out.print("El archivo se llama: ");
468             celda.setCellValue(Principal.contenedorFrames.get(i).
                    getNombreFile());
469 
470             // posicion indicada por el contador del ciclo
471             Cell celda2 = fila.createCell(1);
472 
473             //System.out.print("pertenece al segundo: ");
474             celda2.setCellValue(Principal.contenedorFrames.get(i).getTiempo
                    ());
475 
476             //PROCEDIMIENTO PARA IMPRIMIR CADENA EN LUGAR DEL NUMERO, mas
                    bonito
477             int clase = Principal.contenedorFrames.get(i).getClaseMayor();
478             String claseCadena = aux.claseEnteroACadena(clase);
479 
480             // posicion indicada por el contador del ciclo
481             Cell celda3 = fila.createCell(2);
482 
483             //System.out.print("y es de la clase (MAYORITARIA): ");
484             celda3.setCellValue(claseCadena);
485 
486             //IMPRESION DE TODAS LAS CLASES PERTENECIENTES
487             ArrayList<Integer> clases = new ArrayList<>();
488             clases = Principal.contenedorFrames.get(i).getClases();
489 
490             String clasesCadena = null;
491 
492             claseCadena = null;
493 
494             for(int j=0; j<clases.size(); j++){
```

```
495
496                 clase = clases.get(j);
497
498                 claseCadena = aux.claseEnteroACadena(clase);
499
500                 if(j==0){
501
502                     clasesCadena = claseCadena;
503
504                 }else{
505
506                     clasesCadena= clasesCadena + ", " + claseCadena;
507
508                 }
509
510             }
511
512             // posicion indicada por el contador del ciclo
513             Cell celda4 = fila.createCell(3);
514
515
516             pagina.autoSizeColumn(i);
517
518         }
519
520         // Ahora guardaremos el archivo
521         try {
522             // Creamos el flujo de salida de datos,
523             // apuntando al archivo donde queremos
524             // almacenar el libro de Excel
525             FileOutputStream salida = new FileOutputStream(archivo);
526
527             // Almacenamos el libro de
528             // Excel via ese
529             // flujo de datos
530             workbook.write(salida);
```

```
531
532             // Cerramos el libro para concluir operaciones
533             workbook.close();
534
535             aux.LOGGER.log(Level.INFO, "Archivo creado existosamente en {0}
                    ", archivo.getAbsolutePath());
536
537         } catch (FileNotFoundException ex) {
538             aux.LOGGER.log(Level.SEVERE, "Error con la creacion del archivo
                    ");
539         } catch (IOException ex) {
540             aux.LOGGER.log(Level.SEVERE, "Error de entrada/salida");
541         }
542
543
544     }
545
546     //crear datasetUNICO
547
548     public static void crearDatasetUnico(ArrayList<Double> nuevosDatos,
            String nombre) throws FileNotFoundException, IOException{
549
550             String nombreFichero = Principal.rutaGuardar + "\\" + Principal
                    .nombreDatasetUnico + ".xlsx" ;
551
552             File archivo = new File(nombreFichero);
553
554             Workbook workbook = new XSSFWorkbook();
555
556             Sheet pagina = workbook.createSheet("Resumen de las emociones
                    de los sujetos");
557
558             CellStyle style = workbook.createCellStyle();
559
560
561             String[] titulos = { "archivo", "anger", "contempt", "disgust",
```

```
                "fear", "happiness",
562                                 "neutral", "sadness", "surprise", "% anger
                                       ", "% contempt", "%disgust",
563                                 "% fear", "% happiness", "% neutral", "%
                                       sadness", "% surprise", "analizadas",
564                                 "no analizadas", "utiles"};
565             //Integer[] datos = {};
566
567             Row fila = pagina.createRow(0);
568
569             for (int i = 0; i < titulos.length; i++) {
570
571                 Cell celda = fila.createCell(i);
572                 celda.setCellValue(titulos[i]);
573                 pagina.autoSizeColumn(i);
574             }
575
576             fila = pagina.createRow(1);
577
578             Cell celda = fila.createCell(0);
579             celda.setCellValue(nombre);
580             pagina.autoSizeColumn(0);
581
582
583             for (int i = 0; i < nuevosDatos.size(); i++) {
584
585                 celda = fila.createCell(i+1);
586                 celda.setCellValue(nuevosDatos.get(i));
587                 pagina.autoSizeColumn(i+1);
588             }
589
590             FileOutputStream salida = new FileOutputStream(archivo);
591
592             workbook.write(salida);
593
594             workbook.close();
```

```
595
596
597     }
598
599     //ver si existe dataset unico
600     public static boolean existeDatasetUnico(){
601
602         boolean resultado = false;
603
604         String rutaFile = Principal.rutaGuardar + "\\" + Principal.
                nombreDatasetUnico + ".xlsx" ;
605
606         //COMPROBACION, de la ruta de video elegido
607         System.out.println("La ruta del video elegido es: ");
608         System.out.println(rutaFile);
609
610         //PARA PROBAR CONTENIDO
611
612         //declaracion del File para la comprobaci n
613         File archivo = new File(rutaFile);
614
615         //COMPROBAR SI EXISTE
616         resultado = Files.exists(archivo.toPath()) ;
617
618         return resultado;
619
620     }
621
622     //cargar dataset ecxistente
623
624     public static void cargarDatasetUnico() throws IOException{
625
626         String nombreFichero = Principal.rutaGuardar + "\\" + Principal.
                nombreDatasetUnico + ".xlsx" ;
627
628         FileInputStream file = new FileInputStream(new File(nombreFichero))
```

```
            ;
629
630         XSSFWorkbook workbook = new XSSFWorkbook(file);
631
632         XSSFSheet sheet = workbook.getSheetAt(0);
633         Iterator<Row> rowIterator = sheet.iterator();
634
635         Row row;
636             //para saltar 1
637             row = rowIterator.next();
638
639         while (rowIterator.hasNext()){
640
641             row = rowIterator.next();
642
643             Iterator<Cell> cellIterator = row.cellIterator();
644
645             Cell celda;
646
647             ArrayList<Double> aux = new ArrayList<>();
648
649                 while (cellIterator.hasNext()){
650
651                 celda = cellIterator.next();
652
653                 switch(celda.getCellType()) {
654
655                 case Cell.CELL_TYPE_NUMERIC:
656
657                     if( DateUtil.isCellDateFormatted(celda) ){
658
659                       // System.out.print(celda.getDateCellValue()+" | ");
660
661
662                     }else{
663
```

```
664                         //System.out.print(celda.getNumericCellValue()+" |
                                ");
665                             aux.add((Double)celda.getNumericCellValue())
                                    ;
666
667                     }
668
669                     break;
670
671                 case Cell.CELL_TYPE_STRING:
672
673                    // System.out.print(celda.getStringCellValue()+" | ");
674                             Principal.nombresDatasetUnico.add(celda.
                                getStringCellValue());
675
676                     break;
677
678                 case Cell.CELL_TYPE_BOOLEAN:
679
680                     //System.out.print(celda.getBooleanCellValue()+" | ");
681
682                     break;
683
684                 }
685
686             }
687             Principal.arrayDatosDatasetUnico.add(aux);
688             // System.out.println(row.getRowNum());
689
690             Principal.rowsDatasetUnico.add(row.getRowNum());
691
692         }
693
694         workbook.close();
695
696     }
```

```
697
698
699     //agregarA datasetUNICO
700     public static void agregarADatasetUnico(ArrayList<Double> nuevosDatos,
            String nombre) throws FileNotFoundException, IOException{
701
702         int ultimo = 0;
703
704         String nombreFichero = Principal.rutaGuardar + "\\" + Principal.
                nombreDatasetUnico + ".xlsx" ;
705
706         File archivo = new File(nombreFichero);
707
708         Workbook workbook = new XSSFWorkbook();
709
710         Sheet pagina = workbook.createSheet("Resumen de las emociones del
                sujeto");
711
712         CellStyle style = workbook.createCellStyle();
713
714         String[] titulos = { "archivo", "anger", "contempt", "disgust", "
                fear", "happiness", "neutral", "sadness", "surprise", "% anger",
                 "% contempt", "%disgust", "% fear", "% happiness", "% neutral",
                 "% sadness", "% surprise", "analizadas", "no analizadas", "
                utiles"};
715
716         Row fila = pagina.createRow(0);
717
718
719         for (int i = 0; i < titulos.length; i++) {
720             Cell celda = fila.createCell(i);
721             celda.setCellValue(titulos[i]);
722             pagina.autoSizeColumn(i);
723         }
724
725
```

```
726         for(int i=0; i< Principal.rowsDatasetUnico.size(); i++){
727
728             fila = pagina.createRow(Principal.rowsDatasetUnico.get(i));
729             ultimo = Principal.rowsDatasetUnico.get(i);
730             Cell celda = fila.createCell(0);
731             celda.setCellValue(Principal.nombresDatasetUnico.get(i));
732             pagina.autoSizeColumn(0);
733
734             for (int j = 0; j < Principal.arrayDatosDatasetUnico.get(i).
                    size() ; j++) {
735
736                 celda = fila.createCell(j+1);
737                 celda.setCellValue(Principal.arrayDatosDatasetUnico.get(i).
                        get(j));
738                 pagina.autoSizeColumn(j+1);
739
740             }
741
742         }
743
744
745         fila = pagina.createRow(ultimo+1);
746
747         Cell celda = fila.createCell(0);
748         celda.setCellValue(nombre);
749         pagina.autoSizeColumn(0);
750
751
752         for (int i = 0; i < nuevosDatos.size(); i++) {
753
754             celda = fila.createCell(i+1);
755             celda.setCellValue(nuevosDatos.get(i));
756             pagina.autoSizeColumn(i+1);
757         }
758
759         FileOutputStream salida = new FileOutputStream(archivo);
```

```
760
761             workbook.write(salida);
762
763             workbook.close();
764
765
766         }
767
768
769
770         //para rellenar el vector de los nuevos datos que posteriormente se
                introducira en el dataset unico
771         public static  ArrayList<Double>  rellenarVectorDatasetUnico() {
772
773            ArrayList<Double> nuevosDatos = new ArrayList<>();
774
775             nuevosDatos.add(Double.valueOf(Principal.angerContador));
776             nuevosDatos.add(Double.valueOf(Principal.contemptContador));
777             nuevosDatos.add(Double.valueOf(Principal.disgustContador));
778             nuevosDatos.add(Double.valueOf(Principal.fearContador));
779             nuevosDatos.add(Double.valueOf(Principal.happinessContador));
780             nuevosDatos.add(Double.valueOf(Principal.neutralContador));
781             nuevosDatos.add(Double.valueOf(Principal.sadnessContador));
782             nuevosDatos.add(Double.valueOf(Principal.surpriseContador));
783
784             //PORCENTAJES
785             nuevosDatos.add(Principal.angerContadorPorcentaje);
786             nuevosDatos.add(Principal.contemptContadorPorcentaje);
787             nuevosDatos.add(Principal.disgustContadorPorcentaje);
788             nuevosDatos.add(Principal.fearContadorPorcentaje);
789             nuevosDatos.add(Principal.happinessContadorPorcentaje);
790             nuevosDatos.add(Principal.neutralContadorPorcentaje);
791             nuevosDatos.add(Principal.sadnessContadorPorcentaje);
792             nuevosDatos.add(Principal.surpriseContadorPorcentaje);
793
794             nuevosDatos.add(Double.valueOf(Principal.nFotosAnalizadas));
```

```
795             nuevosDatos.add(Double.valueOf(Principal.noAnalizadoContador));
796             nuevosDatos.add(Double.valueOf(Principal.nFotosAnalizadas)-Double.
                    valueOf(Principal.noAnalizadoContador));
797
798             return nuevosDatos;
799
800         }
801
802
803     }
```

4.2 FUNCIONESGRAFICAS.JAVA

```
1  //PAQUETE NECESARIO
2  package PaquetePostAnalisis;
3
4  //IMPORT NECESARIOS
5  import PaqueteAnalisisVideo.Funciones;
6  import PaquetePreAnalisis.Principal;
7  import java.awt.BasicStroke;
8  import java.awt.Color;
9  import java.io.File;
10 import java.io.FileWriter;
11 import java.io.IOException;
12 import java.io.PrintWriter;
13 import java.util.ArrayList;
14 import java.util.logging.Level;
15 import java.util.logging.Logger;
16 import javax.swing.JFrame;
17 import org.jfree.chart.ChartFactory;
18 import org.jfree.chart.ChartPanel;
19 import org.jfree.chart.ChartUtilities;
20 import org.jfree.chart.JFreeChart;
21 import org.jfree.chart.plot.PlotOrientation;
22 import org.jfree.chart.plot.XYPlot;
23 import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
24 import org.jfree.data.category.DefaultCategoryDataset;
25 import org.jfree.data.general.DefaultPieDataset;
26 import org.jfree.data.xy.XYSeries;
27 import org.jfree.data.xy.XYSeriesCollection;
28
29 /**
30  *
31  * @author Angel Murcia Diaz
32  */
33
34 //CLASE FUNCIONESGRAFICAS
```

```
35 public class FuncionesGraficas {
36
37     //FUNCION HACER GRAFICA TIPO PASTEL
38     public void hacerGraficaFinalPastel(){
39
40         //DECLARACION DEL GRAFICO
41         JFreeChart grafico = null;
42         //DECLARACION DEL DEFAULT CATEGORY DATASET
43         DefaultCategoryDataset datos = new DefaultCategoryDataset();
44
45         //PARA PASAR POR TODO EL CONTENEDOR
46         for( int i=0 ; i<Principal.contenedorFrames.size() ; i++ ){
47
48             //CLASE DE CADA UNO DE LOS MIEMBROS DEL CONTENEDOR
49             int clase = Principal.contenedorFrames.get(i).getClaseMayor();
50
51             //
52             double tiempo = Principal.contenedorFrames.get(i).getTiempo();
53
54             String tempo = Double.toString(tiempo);
55
56             datos.addValue( clase , "Sujeto" , tempo);//Principal.
                   contenedorFrames.get(i).getTiempo() );
57
58         }
59
60         String tipoGrafica = "Pastel";
61         if(tipoGrafica.equals("Barras")){
62             grafico = ChartFactory.createBarChart("Grafica Prueba", "Eje X"
                   , "Eje Y",datos ,PlotOrientation.VERTICAL, true, true, false
                   );
63         }
64         if(tipoGrafica.equals("Lineal")){
65             grafico = ChartFactory.createLineChart("Grafica del an lisis
                   de emociones", "Segundo del fotograma", "Tipo de emocion",
                   datos ,PlotOrientation.VERTICAL, true, true, false);
```

```
66          }
67          if(tipoGrafica.equals("Pastel")){
68              DefaultPieDataset datosPie = new DefaultPieDataset();
69             datosPie.setValue("anger", Principal.angerContador);
70             datosPie.setValue("contempt", Principal.contemptContador);
71             datosPie.setValue("disgust", Principal.disgustContador);
72             datosPie.setValue("fear", Principal.fearContador);
73             datosPie.setValue("happiness", Principal.happinessContador);
74             datosPie.setValue("neutral", Principal.neutralContador);
75             datosPie.setValue("sadness", Principal.sadnessContador);
76             datosPie.setValue("neutral", Principal.neutralContador);
77             datosPie.setValue("surprise", Principal.surpriseContador);
78
79             datosPie.setValue("no analizadas", Principal.noAnalizadoContador
                   );
80
81              grafico = ChartFactory.createPieChart("Grafico Pastel con la
                   pertenencia de los fotogramas a las clases", datosPie, true,
                    true, false);
82          }
83
84          ChartPanel cPanel = new ChartPanel(grafico);
85          JFrame informacion = new JFrame("Grafica");
86          informacion.getContentPane().add(cPanel);
87          informacion.pack();
88
89          informacion.setVisible(true);
90
91
92          try {
93              ChartUtilities.saveChartAsPNG(new File("C:\\Users\\angel\\
                   Desktop\\prueba\\GraficaPastel.png"), grafico, 800, 800);
94          } catch (IOException ex) {
95              Logger.getLogger(Funciones.class.getName()).log(Level.SEVERE,
                   null, ex);
96          }
```

```
97
98
99      }
100
101     public void hacerGraficaFinalBarras(){
102
103
104         JFreeChart grafico = null;
105         DefaultCategoryDataset datos = new DefaultCategoryDataset();
106
107         datos.addValue( Principal.angerContador, "anger", "Grafica");
108         datos.addValue( Principal.contemptContador, "contempt", "Grafica");
109         datos.addValue( Principal.disgustContador, "disgrust", "Grafica");
110         datos.addValue( Principal.fearContador, "fear", "Grafica");
111         datos.addValue( Principal.happinessContador, "happiness", "Grafica"
                );
112         datos.addValue( Principal.neutralContador, "neutral", "Grafica");
113         datos.addValue( Principal.sadnessContador, "sadness", "Grafica");
114         datos.addValue( Principal.surpriseContador, "surprise", "Grafica");
115         datos.addValue( Principal.noAnalizadoContador, "NO analizados", "
                Grafica");
116
117         String tipoGrafica = "Barras";
118         if(tipoGrafica.equals("Barras")){
119             grafico = ChartFactory.createBarChart("Grafica del an lisis de
                    emociones", "Segundo del fotograma", "Tipo de emocion",
                    datos ,PlotOrientation.VERTICAL, true, true, false);
120         }
121         if(tipoGrafica.equals("Lineal")){
122             grafico = ChartFactory.createLineChart("Grafica del an lisis
                    de emociones", "Segundo del fotograma", "Tipo de emocion",
                    datos ,PlotOrientation.VERTICAL, true, true, false);
123         }
124
125         ChartPanel cPanel = new ChartPanel(grafico);
126         JFrame informacion = new JFrame("Grafica");
```

```
127         informacion.getContentPane().add(cPanel);
128         informacion.pack();
129 
130         informacion.setVisible(true);
131 
132         try {
133             ChartUtilities.saveChartAsPNG(new File("C:\\Users\\angel\\
                Desktop\\prueba\\GraficaBarras.png"), grafico, 800, 800);
134         } catch (IOException ex) {
135             Logger.getLogger(Funciones.class.getName()).log(Level.SEVERE,
                null, ex);
136         }
137     }
138 
139 
140     //INTENTO DE FUNCION UNICA PARA SACAR TODAS LAS GRAFICAS
141     public void hacerGraficaEmocionTiempoGenerica(int tipo){
142 
143         //CRECION DE JFreeChart objeto para las graficas
144         JFreeChart grafico = null;
145         //crear un archivador de datos para nuestro GRAFICO
146         //DefaultCategoryDataset datos = new DefaultCategoryDataset();
147 
148         XYSeries unDato = new XYSeries("XDDeito");
149 
150         XYSeriesCollection datos = new XYSeriesCollection();
151 
152         //DECLARACION DE LA CLASE FUNCIONES PARA UTILIZARLA COMO AUXILIAR
153         Funciones funAux = new Funciones();
154 
155         //RECORRER TODOS LOS FRAMES almacenados en el vector
156         for( int i=0 ; i<Principal.contenedorFrames.size() ; i++ ){
157 
158             //CREACION ARRAY AUXILIAR para introducir el de cada FRAME
159             ArrayList<Integer> clases = new ArrayList<>();
160             //INTRODUCIR EL DE CADA FRAME
```

```
161             clases = Principal.contenedorFrames.get(i).getClases();
162
163             //PARA EL TIEMPO
164             //INTRODUCION DEL TIEMPO DEL FRAME NE VARIABLE AUXILIAR
165             double tiempo = Principal.contenedorFrames.get(i).getTiempo();
166
167             //PASO A ENTERO
168             int tiempoEntero = (int) tiempo;
169
170             //PASO A STRING
171             String tempo = Integer.toString(tiempoEntero);
172
173             //CLASE AUXILIAR
174             int clase;
175
176             //VARIABLE AUXILIAR PARA ACTIVAR O DESACTIVAR dependiendo si se
                    encuentra la clase buscada o no
177             boolean esta = false;
178
179             //RECORRER TODO EL VECTOR DE CLASES
180             for(int j=0; j<clases.size(); j++){
181
182                 //INTRODUCION DE LA CLASE EN LA variable auxiliar
183                 clase = clases.get(j);
184
185                 //COMPROBACION DE LA CLASE DESEADA
186                 if(clase == tipo){
187
188                     //PONER BOOLEANO COPMO VERDADERO
189                     esta = true;
190
191                 }
192
193             }
194
195             //SI EL BOOLEANO ESTA EN VERDADERO
```

```
196             if(esta == true){
197 
198                 //INTRODUCION DE DATOS DE GRAFICA en el archivador de datos
199                 unDato.add(tiempoEntero, 1.0);
200 
201             }else{
202 
203                 //INTRODUCION DE DATOS DE GRAFICA en el archivador de datos
204                 //con el valor a o, ES DECIR NO EXISTE ESA EMOCION EN ESE
                        FRAME
205                 unDato.add(tiempoEntero, 0.0);
206 
207             }
208         }
209 
210 
211         datos.addSeries(unDato);
212 
213 
214         grafico = ChartFactory.createXYLineChart("Grafica de la emocion " +
                 funAux.claseEnteroACadena(tipo), "Segundos transcurridos",
                funAux.claseEnteroACadena(tipo), datos, PlotOrientation.VERTICAL
                , false, true, false);
215 
216 
217         XYPlot plot = grafico.getXYPlot();
218 
219         XYLineAndShapeRenderer rendered = new XYLineAndShapeRenderer();
220 
221 
222         switch(tipo) {
223             case 0:
224               rendered.setSeriesPaint(0, Color.RED);
225               break;
226             case 1:
227               rendered.setSeriesPaint(0, Color.MAGENTA);
```

```
228                 break;
229             case 2:
230                 rendered.setSeriesPaint(0, Color.GREEN);
231                 break;
232             case 3:
233                 rendered.setSeriesPaint(0, Color.BLACK);
234                 break;
235             case 4:
236                 rendered.setSeriesPaint(0, Color.YELLOW);
237                 break;
238             case 5:
239                 rendered.setSeriesPaint(0, Color.CYAN);
240                 break;
241             case 6:
242                 rendered.setSeriesPaint(0, Color.BLUE);
243                 break;
244             case 7:
245                 rendered.setSeriesPaint(0, Color.ORANGE);
246                 break;
247             case 8:
248                 rendered.setSeriesPaint(0, Color.DARK_GRAY);
249                 break;
250             default:
251                 rendered.setSeriesPaint(0, Color.DARK_GRAY);
252         }
253 
254         rendered.setSeriesStroke(0, new BasicStroke(0.6f));
255 
256          rendered.setSeriesShapesVisible(0, false);
257 
258 
259 
260         plot.setRenderer(rendered);
261 
262 
263         //CREACION DEL PANEL PARA MOSTRRAR (VENTANA)
```

```
264             ChartPanel cPanel = new ChartPanel(grafico);
265             //NOMBRE DE PANEL
266             JFrame informacion = new JFrame("Grafica de la emocion" + funAux.
                    claseEnteroACadena(tipo));
267             //INTRODUCIR NOMBRE
268             informacion.getContentPane().add(cPanel);
269             informacion.pack();
270 
271             //PONER VISIBLE EL PANEL
272             informacion.setVisible(true);
273         }
274     }
```

Código PaqueteVentanas

5.1 VentanaPrincipal.java

```
1  //INCLUSION NECESARIA DEL PAQUETE
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIOS
5  import PaqueteAnalisisVideo.HiloProcesoPrincipal;
6  import PaqueteAnalisisVideo.Funciones;
7  import PaquetePreAnalisis.Principal;
8  import java.io.File;
9  import java.io.IOException;
10 import java.nio.file.Files;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13 import javax.swing.ImageIcon;
14 import javax.swing.JFileChooser;
15 import javax.swing.filechooser.FileNameExtensionFilter;
16
17
18
19 /**
20  *
21  * @author Angel Murcia Diaz
22  */
23
24 //CLASE VENTANA PRINCIPAL
25 public class VentanaPrincipal extends javax.swing.JFrame {
26
27     //CONSTRUCTOR
```

```
28      public VentanaPrincipal() {
29          initComponents();
30          //icono de la aplicacion
31          //setIconImage(new ImageIcon(getClass().getResource("/
                PaqueteImagenes/instrumentos32.png")).getImage());
32      }
33
34      @SuppressWarnings("unchecked")
35      // <editor-fold defaultstate="collapsed" desc="Generated Code">
36      private void initComponents() {
37
38          BotonAnalizar = new javax.swing.JButton();
39          BotonOpciones = new javax.swing.JButton();
40          BotonUsuarios = new javax.swing.JButton();
41
42          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
                ;
43          setTitle("MENU PRINCIPAL");
44
45          BotonAnalizar.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
46          BotonAnalizar.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/play32.png"))); // NOI18N
47          BotonAnalizar.setText("ANALIZAR VIDEO");
48          BotonAnalizar.setToolTipText("Clica aqu  para empezar a analizar
                el video.");
49          BotonAnalizar.setDefaultCapable(false);
50          BotonAnalizar.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
51          BotonAnalizar.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
52          BotonAnalizar.addActionListener(new java.awt.event.ActionListener()
                {
53              public void actionPerformed(java.awt.event.ActionEvent evt) {
54                  BotonAnalizarActionPerformed(evt);
55              }
```

```
56          });
57
58          BotonOpciones.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
59          BotonOpciones.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/usuario32.png"))); // NOI18N
60          BotonOpciones.setText("MENU USUARIO");
61          BotonOpciones.setToolTipText("Clic aqu  para cambiar a un nuevo
                usuario o en su defecto para crear un nuevo usuario, donde se
                introduzca una clave de Microsoft API.");
62          BotonOpciones.setDefaultCapable(false);
63          BotonOpciones.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
64          BotonOpciones.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
65          BotonOpciones.addActionListener(new java.awt.event.ActionListener()
                 {
66              public void actionPerformed(java.awt.event.ActionEvent evt) {
67                  BotonOpcionesActionPerformed(evt);
68              }
69          });
70
71          BotonUsuarios.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
72          BotonUsuarios.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/instrumentos32.png"))); // NOI18N
73          BotonUsuarios.setText("OPCIONES");
74          BotonUsuarios.setToolTipText("Clic aqu  para cambiar las opciones
                deseadas que se aplicar n al posterior analisis de emociones en
                 un archivo de video.");
75          BotonUsuarios.setDefaultCapable(false);
76          BotonUsuarios.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
77          BotonUsuarios.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
78          BotonUsuarios.addActionListener(new java.awt.event.ActionListener()
```

```
            {
79              public void actionPerformed(java.awt.event.ActionEvent evt) {
80                  BotonUsuariosActionPerformed(evt);
81              }
82          });
83
84          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
85          getContentPane().setLayout(layout);
86          layout.setHorizontalGroup(
87              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
88              .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
                    createSequentialGroup()
89                  .addContainerGap()
90                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.TRAILING)
91                      .addComponent(BotonAnalizar, javax.swing.GroupLayout.
                            DEFAULT_SIZE, 326, Short.MAX_VALUE)
92                      .addGroup(layout.createSequentialGroup()
93                          .addComponent(BotonOpciones, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 160, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
94                          .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
95                          .addComponent(BotonUsuarios, javax.swing.
                                GroupLayout.DEFAULT_SIZE, javax.swing.
                                GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
96                  .addContainerGap())
97          );
98          layout.setVerticalGroup(
99              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
100             .addGroup(layout.createSequentialGroup()
101                 .addContainerGap()
102                 .addGroup(layout.createParallelGroup(javax.swing.
```

```
                    GroupLayout.Alignment.LEADING)
103                  .addComponent(BotonOpciones, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 75, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
104                  .addComponent(BotonUsuarios, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 75, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
105              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                    .RELATED)
106              .addComponent(BotonAnalizar, javax.swing.GroupLayout.
                    DEFAULT_SIZE, 85, Short.MAX_VALUE)
107              .addContainerGap())
108      );
109
110      pack();
111  }// </editor-fold>
112
113  //BOTON ANALIZAR
114  private void BotonAnalizarActionPerformed(java.awt.event.ActionEvent
        evt) {
115
116      //COMPROBAR SI ESTA LA SESION INICIADA
117      if(Principal.sesionIniciada == true){
118
119          if(Principal.hayRutaConfirmada == true){
120
121              //LLAMA A LA FUNCION CARGAR, para la seleccion manual del
                    archivo de video
122              Principal.videoFile = cargar();
123
124              //COMPROBACION, de la ruta de video elegido
125              System.out.println("La ruta del video elegido es: ");
126              System.out.println(Principal.videoFile);
127
128              //PARA PROBAR CONTENIDO
129
```

```
130                 //declaracion del File para la comprobaci n
131                 File archivo = new File(Principal.videoFile);
132 
133                 //String para la comprobacion
134                 String tipodeArchivo = null;
135                 //probar el archivo
136                 try {
137                     tipodeArchivo = Files.probeContentType(archivo.toPath()
                            );
138                 } catch (IOException ex) {
139                     Logger.getLogger(VentanaPrincipal.class.getName()).log(
                            Level.SEVERE, null, ex);
140                 }
141 
142 
143                 //System.out.println(tipodeArchivo);
144 
145                 //manipulacion de la cadena para comprobar facilmente si es
                        de tipo video o no
146                 String tipodeArchivoComparar = tipodeArchivo.substring(0, 5
                        );
147 
148                 System.out.println("El archivo elegido es: ");
149                 System.out.println("|"+tipodeArchivoComparar+"|");
150 
151                 //comprobacion de si es o no un archivo de video
152                 if("video".equals(tipodeArchivoComparar)){
153 
154                     //PONER LA VENTANA NO VISIBLE
155                     this.setVisible(false);
156 
157                     //resetear valores iniciales de los contadores
158                     new Funciones().inicializarValoresIniciales();
159 
160                     //SE REINICIA EL VECTOR(incluir en ionicializar)
161                     Principal.contenedorFrames.clear();
```

```
162
163                 //VENTANAAAAAA DE SEGUIMIENTO
164                 //PONER VISIBLE LA VENTANA
165                 Principal.ventana_seguimiento.setVisible(true);
166
167                 //PONER LA VENTANA DE TAMA O FIJO
168                 Principal.ventana_seguimiento.setResizable(false);
169
170                 //HILO PARA CREAR EL SUDPROCESO QUE HARA TODO EL
                        ANALISIS DE VIDEO
171                 HiloProcesoPrincipal hilo1=new HiloProcesoPrincipal("
                        Proceso principal de analisis de video");
172
173                 //Principal.hilo1.start();
174                 hilo1.start();
175
176             //ELSE: ERROR no es archivo de video
177             }else{
178
179                 System.out.println("Seleccione un archivo de video");
180
181             }
182
183         }else{
184
185             //ERROR DESEADO
186             Principal.error = "ES NECESARIO SELECCIONAR UNA RUTA PARA
                    LOS ARCHIVOS TEMPORALES, HAGA CLIC EN EL BOTON DE
                    OPCIONES Y POSTERIORMENTE SELECCIONE UNA RUTA";
187
188             //CREACI N VENTANA DE ERROR
189             VentanaError ventana_error = new VentanaError();
190
191             //PONER VISIBLE LA VENTANA
192             ventana_error.setVisible(true);
193
```

```
194                 //PONER LA VENTANA DE TAMA O FIJO
195                 ventana_error.setResizable(false);
196 
197             }
198 
199         }else{
200 
201             //ERROR DESEADO
202             Principal.error = "SE DEBE DE IDENTIFICAR EL USUARIO ANTES DE
                    PODER ANALIZAR CUALQUIER VIDEO";
203 
204             //CREACI N VENTANA DE ERROR
205             VentanaError ventana_error = new VentanaError();
206 
207             //PONER VISIBLE LA VENTANA
208             ventana_error.setVisible(true);
209 
210             //PONER LA VENTANA DE TAMA O FIJO
211             ventana_error.setResizable(false);
212         }
213 
214     }
215 
216     //FUNCION BOTON OPCIONES
217     private void BotonOpcionesActionPerformed(java.awt.event.ActionEvent
            evt) {
218 
219         //PONER VISIBLE LA VENTANA
220         Principal.ventana_usuarios.setVisible(true);
221 
222         //PONER LA VENTANA DE TAMA O FIJO
223         Principal.ventana_usuarios.setResizable(false);
224 
225     }
226 
227     //FUNCION BOTON USUARIOS
```

```
228     private void BotonUsuariosActionPerformed(java.awt.event.ActionEvent
            evt) {
229
230         System.out.println("El usuario que se utilizara tiene los
                siguientes datos: ");
231         System.out.println("nick: "+ Principal.user.getNick());
232         System.out.println("contrase a: "+  Principal.user.getPass());
233         System.out.println("punto de conexion: " + Principal.user.getPconex
                ());
234         System.out.println("Version de Prueba: " + Principal.user.
                getVersionPrueba());
235
236         //PONER VISIBLE LA VENTANA
237         Principal.ventana_opciones.setVisible(true);
238
239         //PONER LA VENTANA DE TAMA O FIJO
240         Principal.ventana_opciones.setResizable(false);
241
242     }
243
244     //FUNCION PRINCIPAL
245     public static void main(String args[]) {
246
247         try {
248             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
249                 if ("Nimbus".equals(info.getName())) {
250                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
251                     break;
252                 }
253             }
254         } catch (ClassNotFoundException ex) {
255             java.util.logging.Logger.getLogger(VentanaPrincipal.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
256         } catch (InstantiationException ex) {
```

```
257             java.util.logging.Logger.getLogger(VentanaPrincipal.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
258         } catch (IllegalAccessException ex) {
259             java.util.logging.Logger.getLogger(VentanaPrincipal.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
260         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
261             java.util.logging.Logger.getLogger(VentanaPrincipal.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
262         }
263 
264         java.awt.EventQueue.invokeLater(new Runnable() {
265             public void run() {
266                 new VentanaPrincipal().setVisible(true);
267             }
268         });
269     }
270 
271     //FUNCION CARGAR
272     public String cargar(){
273 
274         //fichero seleccionado
275         JFileChooser fichero = new JFileChooser();
276 
277         //atajos
278         fichero.setFileFilter(new FileNameExtensionFilter("Archivos de
                    video", "mkv", "mp4", "wmv"));
279 
280         //ventana madre para mostrar la ventana de abrir
281         int option = fichero.showDialog(this, "Abrir");
282 
283         //pasar el nombre del video
284         String aux;
285         aux = fichero.getSelectedFile().toString();
286         Principal.nombreFichero = aux.substring( aux.lastIndexOf("\\")+1 ,
                    aux.lastIndexOf(".") );
287 
```

```
288             System.out.print("Nombre del archivo a analizar: ");
289             System.out.println(Principal.nombreFichero);
290 
291             //devuelve directamente el fichero especificado
292             return fichero.getSelectedFile().toString();
293 
294         }
295 
296 
297         // Variables declaration - do not modify
298         private javax.swing.JButton BotonAnalizar;
299         private javax.swing.JButton BotonOpciones;
300         private javax.swing.JButton BotonUsuarios;
301         // End of variables declaration
302     }
303 
304     //YA NO ES UTIL
305     class EjemploHilo extends Thread
306     {
307         public void run()
308         {
309             // C digo del hilo
310         }
311     }
```

5.2 VentanaAviso.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIO
5  import PaquetePreAnalisis.Principal;
6
7  /**
8   *
9   * @author Angel Murcia Diaz
10  */
11
12 //VENTANA: VENTANAAVISO
13 public class VentanaAviso extends javax.swing.JFrame {
14
15     //CONSTRUCTOR
16     public VentanaAviso() {
17         initComponents();
18         etiquetaDescripcionError.setText(Principal.error);
19         etiquetaDescripcionError.setEditable(false);
20     }
21
22
23     @SuppressWarnings("unchecked")
24     // <editor-fold defaultstate="collapsed" desc="Generated Code">
25     private void initComponents() {
26
27         Imagen = new javax.swing.JLabel();
28         etiquetaError = new javax.swing.JLabel();
29         panelDeslizador = new javax.swing.JScrollPane();
30         etiquetaDescripcionError = new javax.swing.JTextPane();
31
32         setDefaultCloseOperation(javax.swing.WindowConstants.
               DISPOSE_ON_CLOSE);
33         setTitle("VENTANA DE AVISO");
```

```
34
35          Imagen.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                PaqueteImagenes/comprobado128.png"))); // NOI18N
36
37          etiquetaError.setFont(new java.awt.Font("Tahoma", 1, 14)); //
                NOI18N
38          etiquetaError.setText("AVISO");
39
40          etiquetaDescripcionError.setFont(new java.awt.Font("Tahoma", 1, 11)
                ); // NOI18N
41          panelDeslizador.setViewportView(etiquetaDescripcionError);
42
43          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
44          getContentPane().setLayout(layout);
45          layout.setHorizontalGroup(
46              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
47              .addGroup(layout.createSequentialGroup()
48                  .addGap(4, 4, 4)
49                  .addComponent(Imagen)
50                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .RELATED)
51                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
52                      .addGroup(layout.createSequentialGroup()
53                          .addComponent(panelDeslizador)
54                          .addContainerGap())
55                      .addGroup(layout.createSequentialGroup()
56                          .addComponent(etiquetaError, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 148, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
57                          .addGap(34, 90, Short.MAX_VALUE))))
58          );
59          layout.setVerticalGroup(
60              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
```

```
                LEADING)
61              .addGroup(layout.createSequentialGroup()
62                  .addContainerGap()
63                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
64                      .addGroup(layout.createSequentialGroup()
65                          .addComponent(Imagen, javax.swing.GroupLayout.
                                PREFERRED_SIZE, 139, javax.swing.GroupLayout.
                                PREFERRED_SIZE)
66                          .addGap(0, 0, Short.MAX_VALUE))
67                      .addGroup(layout.createSequentialGroup()
68                          .addComponent(etiquetaError, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 32, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
69                          .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
70                          .addComponent(panelDeslizador)))
71                  .addContainerGap())
72          );
73
74          pack();
75      }// </editor-fold>
76
77      //FUNCION PRINCIPAL
78      public static void main(String args[]) {
79          try {
80              for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
81                  if ("Nimbus".equals(info.getName())) {
82                      javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
83                      break;
84                  }
85              }
86          } catch (ClassNotFoundException ex) {
87              java.util.logging.Logger.getLogger(VentanaAviso.class.getName()
```

```
                ).log(java.util.logging.Level.SEVERE, null, ex);
88          } catch (InstantiationException ex) {
89              java.util.logging.Logger.getLogger(VentanaAviso.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
90          } catch (IllegalAccessException ex) {
91              java.util.logging.Logger.getLogger(VentanaAviso.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
92          } catch (javax.swing.UnsupportedLookAndFeelException ex) {
93              java.util.logging.Logger.getLogger(VentanaAviso.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
94          }
95
96          java.awt.EventQueue.invokeLater(new Runnable() {
97              public void run() {
98                  new VentanaAviso().setVisible(true);
99              }
100         });
101     }
102     // Variables declaration - do not modify
103     private javax.swing.JLabel Imagen;
104     private javax.swing.JTextPane etiquetaDescripcionError;
105     private javax.swing.JLabel etiquetaError;
106     private javax.swing.JScrollPane panelDeslizador;
107     // End of variables declaration
108 }
```

5.3 VentanaDespuesDeAnalisis.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIO
5  import PaquetePreAnalisis.Principal;
6
7  /**
8   *
9   * @author Angel Murcia Diaz
10  */
11
12 //VENTANA: VENTANADESPUESDEANALISIS
13 public class VentanaDespuesDeAnalisis extends javax.swing.JFrame {
14
15     //CONSTRUCTOR
16     public VentanaDespuesDeAnalisis() {
17         initComponents();
18     }
19
20
21     @SuppressWarnings("unchecked")
22     // <editor-fold defaultstate="collapsed" desc="Generated Code">
23     private void initComponents() {
24
25         botonResultadosFinales = new javax.swing.JButton();
26         botonGraficas = new javax.swing.JButton();
27         botonVolverMenuPrincipal = new javax.swing.JButton();
28
29         setDefaultCloseOperation(javax.swing.WindowConstants.
               DISPOSE_ON_CLOSE);
30         setTitle("VENTANA DESPUES DE ANALIZAR");
31
32         botonResultadosFinales.setFont(new java.awt.Font("Candara", 1, 14))
               ; // NOI18N
```

```
33          botonResultadosFinales.setIcon(new javax.swing.ImageIcon(getClass()
                .getResource("/PaqueteImagenes/resultados32.png"))); // NOI18N
34          botonResultadosFinales.setText("RESULTADOS");
35          botonResultadosFinales.setHorizontalTextPosition(javax.swing.
                SwingConstants.CENTER);
36          botonResultadosFinales.setVerticalTextPosition(javax.swing.
                SwingConstants.BOTTOM);
37          botonResultadosFinales.addActionListener(new java.awt.event.
                ActionListener() {
38              public void actionPerformed(java.awt.event.ActionEvent evt) {
39                  botonResultadosFinalesActionPerformed(evt);
40              }
41          });
42
43          botonGraficas.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
44          botonGraficas.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/graficas32.png"))); // NOI18N
45          botonGraficas.setText("GRAFICAS");
46          botonGraficas.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
47          botonGraficas.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
48          botonGraficas.addActionListener(new java.awt.event.ActionListener()
                {
49              public void actionPerformed(java.awt.event.ActionEvent evt) {
50                  botonGraficasActionPerformed(evt);
51              }
52          });
53
54          botonVolverMenuPrincipal.setFont(new java.awt.Font("Candara", 1,
                14)); // NOI18N
55          botonVolverMenuPrincipal.setIcon(new javax.swing.ImageIcon(getClass
                ().getResource("/PaqueteImagenes/atras32.png"))); // NOI18N
56          botonVolverMenuPrincipal.setText("AL INICIO");
57          botonVolverMenuPrincipal.setHorizontalTextPosition(javax.swing.
```

```
            SwingConstants.CENTER);
58          botonVolverMenuPrincipal.setVerticalTextPosition(javax.swing.
            SwingConstants.BOTTOM);
59          botonVolverMenuPrincipal.addActionListener(new java.awt.event.
            ActionListener() {
60              public void actionPerformed(java.awt.event.ActionEvent evt) {
61                  botonVolverMenuPrincipalActionPerformed(evt);
62              }
63          });
64
65          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
            getContentPane());
66          getContentPane().setLayout(layout);
67          layout.setHorizontalGroup(
68              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
69              .addGroup(layout.createSequentialGroup()
70                  .addGap(6, 6, 6)
71                  .addComponent(botonResultadosFinales, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 125, javax.swing.GroupLayout
                    .PREFERRED_SIZE)
72                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                    .UNRELATED)
73                  .addComponent(botonGraficas, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 125, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
74                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                    .RELATED)
75                  .addComponent(botonVolverMenuPrincipal, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 125, javax.swing.GroupLayout
                    .PREFERRED_SIZE)
76                  .addContainerGap(18, Short.MAX_VALUE))
77          );
78          layout.setVerticalGroup(
79              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
```

```
80              .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
                    createSequentialGroup()
81                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE)
82                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.TRAILING)
83                      .addComponent(botonVolverMenuPrincipal, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 86, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
84                      .addComponent(botonResultadosFinales, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 86, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
85                      .addComponent(botonGraficas, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 86, javax.swing.GroupLayout.
                            PREFERRED_SIZE))
86                  .addContainerGap())
87          );
88
89          pack();
90      }// </editor-fold>
91
92      //FUNCION BOTON RESULTADOS FINALES
93      private void botonResultadosFinalesActionPerformed(java.awt.event.
            ActionEvent evt) {
94
95          Principal.ventana_resultados.setVisible(false);
96          Principal.ventana_resultados.actualizar();
97          Principal.ventana_resultados.setVisible(true);
98          Principal.ventana_resultados.setResizable(false);
99
100     }
101
102     //FUNCION BOTON GRAFICAS
103     private void botonGraficasActionPerformed(java.awt.event.ActionEvent
            evt) {
104         // TODO add your handling code here:
```

```
105
106         Principal.ventana_graficas.setVisible(true);
107         Principal.ventana_graficas.setResizable(false);
108     }
109
110     //FUNCION BOTON VOLVER MENU PRINCIPAL
111     private void botonVolverMenuPrincipalActionPerformed(java.awt.event.
            ActionEvent evt) {
112
113         Principal.ventana_final.setVisible(false);
114         Principal.ventana_principal.setVisible(true);
115         Principal.ventana_principal.setResizable(false);
116
117     }
118
119     //FUNCION PRINCIPAL
120     public static void main(String args[]) {
121         try {
122             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
123                 if ("Nimbus".equals(info.getName())) {
124                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
125                     break;
126                 }
127             }
128         } catch (ClassNotFoundException ex) {
129             java.util.logging.Logger.getLogger(VentanaDespuesDeAnalisis.
                    class.getName()).log(java.util.logging.Level.SEVERE, null,
                    ex);
130         } catch (InstantiationException ex) {
131             java.util.logging.Logger.getLogger(VentanaDespuesDeAnalisis.
                    class.getName()).log(java.util.logging.Level.SEVERE, null,
                    ex);
132         } catch (IllegalAccessException ex) {
133             java.util.logging.Logger.getLogger(VentanaDespuesDeAnalisis.
```

```
                class.getName()).log(java.util.logging.Level.SEVERE, null,
                ex);
134         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
135             java.util.logging.Logger.getLogger(VentanaDespuesDeAnalisis.
                class.getName()).log(java.util.logging.Level.SEVERE, null,
                ex);
136         }
137
138         java.awt.EventQueue.invokeLater(new Runnable() {
139             public void run() {
140                 new VentanaDespuesDeAnalisis().setVisible(true);
141             }
142         });
143     }
144
145     // Variables declaration - do not modify
146     private javax.swing.JButton botonGraficas;
147     private javax.swing.JButton botonResultadosFinales;
148     private javax.swing.JButton botonVolverMenuPrincipal;
149     // End of variables declaration
150 }
```

5.4 VentanaError.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIO
5  import PaquetePreAnalisis.Principal;
6
7  /**
8   *
9   * @author Angel Murcia Diaz
10  */
11
12 //VENTANA: VENTANAERROR
13 public class VentanaError extends javax.swing.JFrame {
14
15     //CONSTRUCTOR
16     public VentanaError() {
17         initComponents();
18         etiquetaDescripcionError.setText(Principal.error);
19         etiquetaDescripcionError.setEditable(false);
20
21     }
22
23
24     @SuppressWarnings("unchecked")
25     // <editor-fold defaultstate="collapsed" desc="Generated Code">
26     private void initComponents() {
27
28         Imagen = new javax.swing.JLabel();
29         etiquetaError = new javax.swing.JLabel();
30         panelDeslizador = new javax.swing.JScrollPane();
31         etiquetaDescripcionError = new javax.swing.JTextPane();
32
33         setDefaultCloseOperation(javax.swing.WindowConstants.
               DISPOSE_ON_CLOSE);
```

```
34          setTitle("VENTANA DE ERROR");
35
36          Imagen.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                PaqueteImagenes/error128.png"))); // NOI18N
37
38          etiquetaError.setFont(new java.awt.Font("Tahoma", 1, 14)); //
                NOI18N
39          etiquetaError.setText("AVISO DE ERROR");
40
41          etiquetaDescripcionError.setFont(new java.awt.Font("Tahoma", 1, 11)
                ); // NOI18N
42          panelDeslizador.setViewportView(etiquetaDescripcionError);
43
44          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
45          getContentPane().setLayout(layout);
46          layout.setHorizontalGroup(
47              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
48              .addGroup(layout.createSequentialGroup()
49                  .addGap(4, 4, 4)
50                  .addComponent(Imagen)
51                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .RELATED)
52                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
53                      .addGroup(layout.createSequentialGroup()
54                          .addComponent(panelDeslizador)
55                          .addContainerGap())
56                      .addGroup(layout.createSequentialGroup()
57                          .addComponent(etiquetaError, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 148, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
58                          .addGap(34, 90, Short.MAX_VALUE))))
59          );
60          layout.setVerticalGroup(
```

```
61              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
62              .addGroup(layout.createSequentialGroup()
63                  .addContainerGap()
64                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
65                      .addGroup(layout.createSequentialGroup()
66                          .addComponent(Imagen, javax.swing.GroupLayout.
                                PREFERRED_SIZE, 139, javax.swing.GroupLayout.
                                PREFERRED_SIZE)
67                          .addGap(0, 0, Short.MAX_VALUE))
68                      .addGroup(layout.createSequentialGroup()
69                          .addComponent(etiquetaError, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 32, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
70                          .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
71                          .addComponent(panelDeslizador)))
72                  .addContainerGap())
73          );
74
75          pack();
76      }// </editor-fold>
77
78      //FUNCION PRINCIPAL
79      public static void main(String args[]) {
80
81          try {
82              for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
83                  if ("Nimbus".equals(info.getName())) {
84                      javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
85                      break;
86                  }
87              }
```

```
88          } catch (ClassNotFoundException ex) {
89              java.util.logging.Logger.getLogger(VentanaError.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
90          } catch (InstantiationException ex) {
91              java.util.logging.Logger.getLogger(VentanaError.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
92          } catch (IllegalAccessException ex) {
93              java.util.logging.Logger.getLogger(VentanaError.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
94          } catch (javax.swing.UnsupportedLookAndFeelException ex) {
95              java.util.logging.Logger.getLogger(VentanaError.class.getName()
                    ).log(java.util.logging.Level.SEVERE, null, ex);
96          }
97
98          java.awt.EventQueue.invokeLater(new Runnable() {
99              public void run() {
100                 new VentanaError().setVisible(true);
101             }
102         });
103     }
104
105     // Variables declaration - do not modify
106     private javax.swing.JLabel Imagen;
107     private javax.swing.JTextPane etiquetaDescripcionError;
108     private javax.swing.JLabel etiquetaError;
109     private javax.swing.JScrollPane panelDeslizador;
110     // End of variables declaration
111 }
```

5.5 VentanaErrorParaProceso.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIOS
5  import PaqueteAnalisisVideo.Funciones;
6  import PaquetePreAnalisis.Principal;
7
8  /**
9   *
10  * @author Angel Murcia Diaz
11  */
12
13 //VENTANA: VENTANAERRORPARAPROCESO
14 public class VentanaErrorParaProceso extends javax.swing.JFrame {
15
16     //CONSTRUCTOR
17     public VentanaErrorParaProceso() {
18         initComponents();
19         etiquetaDescripcionError.setText(Principal.error);
20         etiquetaDescripcionError.setEditable(false);
21     }
22
23
24     @SuppressWarnings("unchecked")
25     // <editor-fold defaultstate="collapsed" desc="Generated Code">
26     private void initComponents() {
27
28         Imagen = new javax.swing.JLabel();
29         etiquetaError = new javax.swing.JLabel();
30         panelDeslizador = new javax.swing.JScrollPane();
31         etiquetaDescripcionError = new javax.swing.JTextPane();
32         botonParaProceso = new javax.swing.JButton();
33
34         setDefaultCloseOperation(javax.swing.WindowConstants.
```

```
            DISPOSE_ON_CLOSE);
35          setTitle("VENTANA DE ERROR");
36
37          Imagen.setIcon(new javax.swing.ImageIcon(getClass().getResource("/
                PaqueteImagenes/error128.png"))); // NOI18N
38
39          etiquetaError.setFont(new java.awt.Font("Tahoma", 1, 14)); //
                NOI18N
40          etiquetaError.setText("AVISO DE ERROR");
41
42          etiquetaDescripcionError.setFont(new java.awt.Font("Tahoma", 1, 11)
                ); // NOI18N
43          panelDeslizador.setViewportView(etiquetaDescripcionError);
44
45          botonParaProceso.setText("TERMINAR");
46          botonParaProceso.addActionListener(new java.awt.event.
                ActionListener() {
47              public void actionPerformed(java.awt.event.ActionEvent evt) {
48                  botonParaProcesoActionPerformed(evt);
49              }
50          });
51
52          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
53          getContentPane().setLayout(layout);
54          layout.setHorizontalGroup(
55              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
56              .addGroup(layout.createSequentialGroup()
57                  .addGap(4, 4, 4)
58                  .addComponent(Imagen)
59                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .RELATED)
60                  .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING, false)
61                      .addComponent(etiquetaError, javax.swing.GroupLayout.
```

```
                    PREFERRED_SIZE, 148, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
62                  .addComponent(panelDeslizador, javax.swing.GroupLayout.
                    Alignment.TRAILING)
63                  .addComponent(botonParaProceso, javax.swing.GroupLayout
                    .Alignment.TRAILING, javax.swing.GroupLayout.
                    DEFAULT_SIZE, 398, Short.MAX_VALUE))
64              .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                Short.MAX_VALUE))
65          );
66          layout.setVerticalGroup(
67              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
68              .addGroup(layout.createSequentialGroup()
69                  .addContainerGap()
70                  .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.TRAILING)
71                      .addGroup(layout.createSequentialGroup()
72                          .addComponent(etiquetaError, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 32, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
73                          .addPreferredGap(javax.swing.LayoutStyle.
                            ComponentPlacement.RELATED)
74                          .addComponent(panelDeslizador, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 101, javax.swing.
                            GroupLayout.PREFERRED_SIZE))
75                      .addComponent(Imagen, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 139, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
76                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                    .UNRELATED)
77                  .addComponent(botonParaProceso, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 63, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
78                  .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE))
```

```
79          );
80
81          pack();
82      }// </editor-fold>
83
84      //FUNCION BOTON PARAR PROCESO
85      private void botonParaProcesoActionPerformed(java.awt.event.ActionEvent
            evt) {
86
87          Principal.matarHilo=true;
88          Principal.ventana_seguimiento.setVisible(false);
89          Principal.ventana_principal.setVisible(true);
90          Principal.ventana_principal.setResizable(false);
91          System.out.printf("Proceso terminado.");
92          new Funciones().imprimir("Porcentajes","Porcentajes");
93
94      }
95
96      //FUNCION PRINCIPAL
97      public static void main(String args[]) {
98
99          try {
100             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                UIManager.getInstalledLookAndFeels()) {
101                 if ("Nimbus".equals(info.getName())) {
102                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                        ());
103                     break;
104                 }
105             }
106         } catch (ClassNotFoundException ex) {
107             java.util.logging.Logger.getLogger(VentanaErrorParaProceso.
                class.getName()).log(java.util.logging.Level.SEVERE, null,
                ex);
108         } catch (InstantiationException ex) {
109             java.util.logging.Logger.getLogger(VentanaErrorParaProceso.
```

```
                class.getName()).log(java.util.logging.Level.SEVERE, null,
                ex);
110         } catch (IllegalAccessException ex) {
111             java.util.logging.Logger.getLogger(VentanaErrorParaProceso.
                class.getName()).log(java.util.logging.Level.SEVERE, null,
                ex);
112         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
113             java.util.logging.Logger.getLogger(VentanaErrorParaProceso.
                class.getName()).log(java.util.logging.Level.SEVERE, null,
                ex);
114         }
115 
116         java.awt.EventQueue.invokeLater(new Runnable() {
117             public void run() {
118                 new VentanaErrorParaProceso().setVisible(true);
119             }
120         });
121     }
122 
123     // Variables declaration - do not modify
124     private javax.swing.JLabel Imagen;
125     private javax.swing.JButton botonParaProceso;
126     private javax.swing.JTextPane etiquetaDescripcionError;
127     private javax.swing.JLabel etiquetaError;
128     private javax.swing.JScrollPane panelDeslizador;
129     // End of variables declaration
130 }
```

5.6 VentanaGraficas.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIOS
5  import PaquetePostAnalisis.FuncionesGraficas;
6  import PaquetePreAnalisis.Principal;
7
8  /**
9   *
10  * @author Angel Murcia Diaz
11  */
12
13 //VENTANA: VENTANAGRAFICAS
14 public class VentanaGraficas extends javax.swing.JFrame {
15
16     //CONSTRUCTOR
17     public VentanaGraficas() {
18         initComponents();
19     }
20
21     @SuppressWarnings("unchecked")
22     // <editor-fold defaultstate="collapsed" desc="Generated Code">
23     private void initComponents() {
24
25         botonContempt = new javax.swing.JButton();
26         botonDisgust = new javax.swing.JButton();
27         botonFear = new javax.swing.JButton();
28         botonHappiness = new javax.swing.JButton();
29         botonNeutral = new javax.swing.JButton();
30         botonSadness = new javax.swing.JButton();
31         botonSurprise = new javax.swing.JButton();
32         botonNoAnalizado = new javax.swing.JButton();
33         botonAnger = new javax.swing.JButton();
34         botonPastel = new javax.swing.JButton();
```

```
35          botonBarras = new javax.swing.JButton();
36          botonAtras = new javax.swing.JButton();
37
38          setDefaultCloseOperation(javax.swing.WindowConstants.
                DISPOSE_ON_CLOSE);
39          setTitle("VENTANA GRAFICAS");
40
41          botonContempt.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
42          botonContempt.setText("CONTEMPT");
43          botonContempt.addActionListener(new java.awt.event.ActionListener()
                 {
44              public void actionPerformed(java.awt.event.ActionEvent evt) {
45                  botonContemptActionPerformed(evt);
46              }
47          });
48
49          botonDisgust.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
50          botonDisgust.setText("DISGUST");
51          botonDisgust.addActionListener(new java.awt.event.ActionListener()
                {
52              public void actionPerformed(java.awt.event.ActionEvent evt) {
53                  botonDisgustActionPerformed(evt);
54              }
55          });
56
57          botonFear.setFont(new java.awt.Font("Candara", 1, 14)); // NOI18N
58          botonFear.setText("FEAR");
59          botonFear.addActionListener(new java.awt.event.ActionListener() {
60              public void actionPerformed(java.awt.event.ActionEvent evt) {
61                  botonFearActionPerformed(evt);
62              }
63          });
64
65          botonHappiness.setFont(new java.awt.Font("Candara", 1, 14)); //
```

```
            NOI18N
66          botonHappiness.setText("HAPPINESS");
67          botonHappiness.addActionListener(new java.awt.event.ActionListener
                () {
68              public void actionPerformed(java.awt.event.ActionEvent evt) {
69                  botonHappinessActionPerformed(evt);
70              }
71          });
72
73          botonNeutral.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
74          botonNeutral.setText("NEUTRAL");
75          botonNeutral.addActionListener(new java.awt.event.ActionListener()
                {
76              public void actionPerformed(java.awt.event.ActionEvent evt) {
77                  botonNeutralActionPerformed(evt);
78              }
79          });
80
81          botonSadness.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
82          botonSadness.setText("SADNESS");
83          botonSadness.addActionListener(new java.awt.event.ActionListener()
                {
84              public void actionPerformed(java.awt.event.ActionEvent evt) {
85                  botonSadnessActionPerformed(evt);
86              }
87          });
88
89          botonSurprise.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
90          botonSurprise.setText("SURPRISE");
91          botonSurprise.addActionListener(new java.awt.event.ActionListener()
                {
92              public void actionPerformed(java.awt.event.ActionEvent evt) {
93                  botonSurpriseActionPerformed(evt);
```

```
94              }
95          });
96
97          botonNoAnalizado.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
98          botonNoAnalizado.setText("NO ANALIZADO");
99          botonNoAnalizado.addActionListener(new java.awt.event.
                ActionListener() {
100             public void actionPerformed(java.awt.event.ActionEvent evt) {
101                 botonNoAnalizadoActionPerformed(evt);
102             }
103         });
104
105         botonAnger.setFont(new java.awt.Font("Candara", 1, 14)); // NOI18N
106         botonAnger.setText("ANGER");
107         botonAnger.addActionListener(new java.awt.event.ActionListener() {
108             public void actionPerformed(java.awt.event.ActionEvent evt) {
109                 botonAngerActionPerformed(evt);
110             }
111         });
112
113         botonPastel.setFont(new java.awt.Font("Candara", 1, 14)); // NOI18N
114         botonPastel.setText("G. PASTEL");
115         botonPastel.addActionListener(new java.awt.event.ActionListener() {
116             public void actionPerformed(java.awt.event.ActionEvent evt) {
117                 botonPastelActionPerformed(evt);
118             }
119         });
120
121         botonBarras.setFont(new java.awt.Font("Candara", 1, 14)); // NOI18N
122         botonBarras.setText("G. BARRAS");
123         botonBarras.addActionListener(new java.awt.event.ActionListener() {
124             public void actionPerformed(java.awt.event.ActionEvent evt) {
125                 botonBarrasActionPerformed(evt);
126             }
127         });
```

```
128
129         botonAtras.setFont(new java.awt.Font("Candara", 1, 14)); // NOI18N
130         botonAtras.setIcon(new javax.swing.ImageIcon(getClass().getResource
                ("/PaqueteImagenes/atras32.png"))); // NOI18N
131         botonAtras.setText("ATRAS");
132         botonAtras.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
133         botonAtras.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
134         botonAtras.addActionListener(new java.awt.event.ActionListener() {
135             public void actionPerformed(java.awt.event.ActionEvent evt) {
136                 botonAtrasActionPerformed(evt);
137             }
138         });
139
140         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
141         getContentPane().setLayout(layout);
142         layout.setHorizontalGroup(
143             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
144             .addGroup(layout.createSequentialGroup()
145                 .addContainerGap()
146                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
147                     .addGroup(layout.createSequentialGroup()
148                         .addComponent(botonFear, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 132, javax.swing.GroupLayout.
                            PREFERRED_SIZE)
149                         .addGap(18, 18, 18)
150                         .addComponent(botonHappiness, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 132, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
151                         .addGap(18, 18, 18)
152                         .addComponent(botonNeutral, javax.swing.GroupLayout
                            .PREFERRED_SIZE, 132, javax.swing.GroupLayout.
```

```
                PREFERRED_SIZE))
153             .addGroup(layout.createSequentialGroup()
154                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
155                     .addGroup(layout.createSequentialGroup()
156                         .addComponent(botonAnger, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 132, javax.
                                swing.GroupLayout.PREFERRED_SIZE)
157                         .addGap(18, 18, 18)
158                         .addComponent(botonContempt, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 132, javax.
                                swing.GroupLayout.PREFERRED_SIZE))
159                     .addGroup(layout.createSequentialGroup()
160                         .addComponent(botonSadness, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 132, javax.
                                swing.GroupLayout.PREFERRED_SIZE)
161                         .addGap(18, 18, 18)
162                         .addComponent(botonSurprise, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 132, javax.
                                swing.GroupLayout.PREFERRED_SIZE)))
163                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
164                     .addGroup(layout.createSequentialGroup()
165                         .addGap(18, 18, 18)
166                         .addComponent(botonDisgust, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 132, javax.
                                swing.GroupLayout.PREFERRED_SIZE))
167                     .addGroup(layout.createSequentialGroup()
168                         .addGap(18, 18, 18)
169                         .addComponent(botonNoAnalizado, javax.swing
                                .GroupLayout.PREFERRED_SIZE, 132, javax.
                                swing.GroupLayout.PREFERRED_SIZE))))
170             .addGroup(layout.createSequentialGroup()
171                 .addGap(69, 69, 69)
172                 .addComponent(botonPastel, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 132, javax.swing.GroupLayout.
```

```
                        PREFERRED_SIZE)
173                     .addGap(36, 36, 36)
174                     .addComponent(botonBarras, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 132, javax.swing.GroupLayout.
                        PREFERRED_SIZE)))
175             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE))
176         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
                createSequentialGroup()
177             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)
178             .addComponent(botonAtras, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 125, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
179             .addGap(157, 157, 157))
180     );
181     layout.setVerticalGroup(
182         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
183         .addGroup(layout.createSequentialGroup()
184             .addContainerGap()
185             .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.LEADING, false)
186                 .addComponent(botonDisgust, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 42, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
187                 .addComponent(botonAnger, javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE)
188                 .addComponent(botonContempt, javax.swing.GroupLayout.
                        Alignment.TRAILING, javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
189             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                    .UNRELATED)
190             .addGroup(layout.createParallelGroup(javax.swing.
```

```
                GroupLayout.Alignment.LEADING)
191              .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.BASELINE)
192                  .addComponent(botonFear, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 44, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
193                  .addComponent(botonHappiness, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 46, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
194              .addComponent(botonNeutral, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 45, javax.swing.GroupLayout.
                    PREFERRED_SIZE))
195          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                .UNRELATED)
196          .addGroup(layout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.LEADING, false)
197              .addComponent(botonSadness, javax.swing.GroupLayout.
                    DEFAULT_SIZE, 43, Short.MAX_VALUE)
198              .addComponent(botonNoAnalizado, javax.swing.GroupLayout
                    .DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                     Short.MAX_VALUE)
199              .addComponent(botonSurprise, javax.swing.GroupLayout.
                    DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE))
200          .addGap(18, 18, 18)
201          .addGroup(layout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.BASELINE)
202              .addComponent(botonPastel, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 44, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
203              .addComponent(botonBarras, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 44, javax.swing.GroupLayout.
                    PREFERRED_SIZE))
204          .addGap(18, 18, 18)
205          .addComponent(botonAtras, javax.swing.GroupLayout.
                PREFERRED_SIZE, 67, javax.swing.GroupLayout.
```

```
                    PREFERRED_SIZE)
206                 .addContainerGap(25, Short.MAX_VALUE))
207         );
208
209         pack();
210     }// </editor-fold>
211
212     private void botonAngerActionPerformed(java.awt.event.ActionEvent evt)
        {
213         // TODO add your handling code here:
214             // new FuncionesGraficas().hacerGraficaAngerTotal();
215               new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(0)
                  ;
216     }
217
218     private void botonContemptActionPerformed(java.awt.event.ActionEvent
        evt) {
219         // TODO add your handling code here:
220              // new FuncionesGraficas().hacerGraficaContemptTotal();
221               new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(1)
                  ;
222     }
223
224     private void botonDisgustActionPerformed(java.awt.event.ActionEvent evt
        ) {
225         // TODO add your handling code here:
226             //  new FuncionesGraficas().hacerGraficaDisgustTotal();
227               new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(2)
                  ;
228     }
229
230     private void botonFearActionPerformed(java.awt.event.ActionEvent evt) {
231         // TODO add your handling code here:
232             //  new FuncionesGraficas().hacerGraficaFearTotal();
233               new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(3)
                  ;
```

```
234     }
235
236     private void botonHappinessActionPerformed(java.awt.event.ActionEvent
            evt) {
237         // TODO add your handling code here:
238               //  new FuncionesGraficas().hacerGraficaHappinessTotal();
239                 new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(4)
                        ;
240     }
241
242     private void botonNeutralActionPerformed(java.awt.event.ActionEvent evt
            ) {
243         // TODO add your handling code here:
244               // new FuncionesGraficas().hacerGraficaNeutralTotal();
245                 new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(5)
                        ;
246     }
247
248     private void botonSadnessActionPerformed(java.awt.event.ActionEvent evt
            ) {
249         // TODO add your handling code here:
250               // new FuncionesGraficas().hacerGraficaSadnessTotal();
251                 new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(6)
                        ;
252     }
253
254     private void botonSurpriseActionPerformed(java.awt.event.ActionEvent
            evt) {
255         // TODO add your handling code here:
256               // new FuncionesGraficas().hacerGraficaSurpriseTotal();
257                 new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(7)
                        ;
258     }
259
260     private void botonNoAnalizadoActionPerformed(java.awt.event.ActionEvent
             evt) {
```

```
261         // TODO add your handling code here:
262             //  new FuncionesGraficas().hacerGraficaNoAnalizadoTotal();
263                 new FuncionesGraficas().hacerGraficaEmocionTiempoGenerica(8)
                        ;
264     }
265 
266     private void botonPastelActionPerformed(java.awt.event.ActionEvent evt)
            {
267         // TODO add your handling code here:
268           new FuncionesGraficas().hacerGraficaFinalPastel();
269     }
270 
271     private void botonBarrasActionPerformed(java.awt.event.ActionEvent evt)
            {
272         // TODO add your handling code here:
273           new FuncionesGraficas().hacerGraficaFinalBarras();
274     }
275 
276     private void botonAtrasActionPerformed(java.awt.event.ActionEvent evt)
           {
277         // TODO add your handling code here:
278 
279         Principal.ventana_graficas.setVisible(false);
280 
281 
282     }
283 
284     //FUNCION PRINCIPAL
285     public static void main(String args[]) {
286 
287         try {
288             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
289                 if ("Nimbus".equals(info.getName())) {
290                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
```

```
291                     break;
292                 }
293             }
294         } catch (ClassNotFoundException ex) {
295             java.util.logging.Logger.getLogger(VentanaGraficas.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
296         } catch (InstantiationException ex) {
297             java.util.logging.Logger.getLogger(VentanaGraficas.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
298         } catch (IllegalAccessException ex) {
299             java.util.logging.Logger.getLogger(VentanaGraficas.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
300         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
301             java.util.logging.Logger.getLogger(VentanaGraficas.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
302         }
303 
304         java.awt.EventQueue.invokeLater(new Runnable() {
305             public void run() {
306                 new VentanaGraficas().setVisible(true);
307             }
308         });
309     }
310 
311     // Variables declaration - do not modify
312     private javax.swing.JButton botonAnger;
313     private javax.swing.JButton botonAtras;
314     private javax.swing.JButton botonBarras;
315     private javax.swing.JButton botonContempt;
316     private javax.swing.JButton botonDisgust;
317     private javax.swing.JButton botonFear;
318     private javax.swing.JButton botonHappiness;
319     private javax.swing.JButton botonNeutral;
320     private javax.swing.JButton botonNoAnalizado;
321     private javax.swing.JButton botonPastel;
322     private javax.swing.JButton botonSadness;
```

```
323     private javax.swing.JButton botonSurprise;
324     // End of variables declaration
325 }
```

5.7 VentanaOpciones.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIOS
5  import PaquetePreAnalisis.Principal;
6  import java.io.File;
7  import java.io.IOException;
8  import java.nio.file.Files;
9  import java.util.logging.Level;
10 import java.util.logging.Logger;
11 import javax.swing.JFileChooser;
12 import javax.swing.filechooser.FileNameExtensionFilter;
13
14 /**
15  *
16  * @author Angel Murcia Diaz
17  */
18
19 //VENTANA: VENTANAOPCIONES
20 public class VentanaOpciones extends javax.swing.JFrame {
21
22     //CONSTRUCTOR
23     public VentanaOpciones() {
24         initComponents();
25     }
26
27     //VARIABLE PRIVADA CON LA RUTA
28     public String ruta=null;
29
30     @SuppressWarnings("unchecked")
31     // <editor-fold defaultstate="collapsed" desc="Generated Code">
32     private void initComponents() {
33
34         etiquetaUmbral = new javax.swing.JLabel();
```

```
35          etiquetaSegundos = new javax.swing.JLabel();
36          etiquetaOpciones = new javax.swing.JLabel();
37          botonAtras = new javax.swing.JButton();
38          botonAceptar = new javax.swing.JButton();
39          etiquetaConservacion = new javax.swing.JLabel();
40          etiquetaRuta = new javax.swing.JLabel();
41          botonSeleccionar = new javax.swing.JButton();
42          valorConservacion = new javax.swing.JComboBox<>();
43          valorUmbral = new javax.swing.JTextField();
44          valorSegundos = new javax.swing.JTextField();
45
46          setDefaultCloseOperation(javax.swing.WindowConstants.
                DISPOSE_ON_CLOSE);
47          setTitle("OPCIONES");
48
49          etiquetaUmbral.setText("Umbral de aceptaci n:");
50
51          etiquetaSegundos.setText("Capturar cada (s):");
52
53          etiquetaOpciones.setFont(new java.awt.Font("Candara", 1, 18)); //
                NOI18N
54          etiquetaOpciones.setText("OPCIONES DE AN LISIS");
55
56          botonAtras.setFont(new java.awt.Font("Candara", 1, 14)); // NOI18N
57          botonAtras.setIcon(new javax.swing.ImageIcon(getClass().getResource
                ("/PaqueteImagenes/atras32.png"))); // NOI18N
58          botonAtras.setText("ATRAS");
59          botonAtras.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
60          botonAtras.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
61          botonAtras.addActionListener(new java.awt.event.ActionListener() {
62              public void actionPerformed(java.awt.event.ActionEvent evt) {
63                  botonAtrasActionPerformed(evt);
64              }
65          });
```

```
66
67          botonAceptar.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
68          botonAceptar.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/play32.png"))); // NOI18N
69          botonAceptar.setText("ACEPTAR");
70          botonAceptar.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
71          botonAceptar.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
72          botonAceptar.addActionListener(new java.awt.event.ActionListener()
                {
73              public void actionPerformed(java.awt.event.ActionEvent evt) {
74                  botonAceptarActionPerformed(evt);
75              }
76          });
77
78          etiquetaConservacion.setText("   Conservacin    y analisis de
                fotogramas?");
79
80          etiquetaRuta.setText("Ruta para guardar archivos:");
81
82          botonSeleccionar.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
83          botonSeleccionar.setText("SELECCIONAR");
84          botonSeleccionar.addActionListener(new java.awt.event.
                ActionListener() {
85              public void actionPerformed(java.awt.event.ActionEvent evt) {
86                  botonSeleccionarActionPerformed(evt);
87              }
88          });
89
90          valorConservacion.setModel(new javax.swing.DefaultComboBoxModel<>(
                new String[] { "No", "Si" }));
91          valorConservacion.addActionListener(new java.awt.event.
                ActionListener() {
```

```
92              public void actionPerformed(java.awt.event.ActionEvent evt) {
93                  valorConservacionActionPerformed(evt);
94              }
95          });
96
97          valorUmbral.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
98          valorUmbral.setText("0.1");
99          valorUmbral.addActionListener(new java.awt.event.ActionListener() {
100             public void actionPerformed(java.awt.event.ActionEvent evt) {
101                 valorUmbralActionPerformed(evt);
102             }
103         });
104
105         valorSegundos.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
106         valorSegundos.setText("2");
107
108         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
109         getContentPane().setLayout(layout);
110         layout.setHorizontalGroup(
111             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
112             .addGroup(layout.createSequentialGroup()
113                 .addGap(29, 29, 29)
114                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
115                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.TRAILING, false)
116                         .addGroup(layout.createSequentialGroup()
117                             .addComponent(botonAceptar, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 201, javax.swing
                                    .GroupLayout.PREFERRED_SIZE)
118                             .addPreferredGap(javax.swing.LayoutStyle.
                                    ComponentPlacement.RELATED, javax.swing.
                                    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
119                             .addComponent(botonAtras, javax.swing.
```

```
                        GroupLayout.PREFERRED_SIZE, 125, javax.swing
                        .GroupLayout.PREFERRED_SIZE))
120                 .addGroup(layout.createSequentialGroup()
121                     .addGroup(layout.createParallelGroup(javax.
                            swing.GroupLayout.Alignment.TRAILING, false)
122                         .addComponent(etiquetaSegundos, javax.swing
                                .GroupLayout.Alignment.LEADING, javax.
                                swing.GroupLayout.DEFAULT_SIZE, javax.
                                swing.GroupLayout.DEFAULT_SIZE, Short.
                                MAX_VALUE)
123                         .addComponent(etiquetaUmbral, javax.swing.
                                GroupLayout.Alignment.LEADING, javax.
                                swing.GroupLayout.DEFAULT_SIZE, javax.
                                swing.GroupLayout.DEFAULT_SIZE, Short.
                                MAX_VALUE)
124                         .addComponent(etiquetaRuta, javax.swing.
                                GroupLayout.Alignment.LEADING, javax.
                                swing.GroupLayout.DEFAULT_SIZE, 152,
                                Short.MAX_VALUE))
125                     .addPreferredGap(javax.swing.LayoutStyle.
                            ComponentPlacement.RELATED)
126                     .addGroup(layout.createParallelGroup(javax.
                            swing.GroupLayout.Alignment.LEADING)
127                         .addComponent(valorUmbral, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 83, javax.
                                swing.GroupLayout.PREFERRED_SIZE)
128                         .addComponent(valorSegundos, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 83, javax.
                                swing.GroupLayout.PREFERRED_SIZE)
129                         .addComponent(botonSeleccionar, javax.swing
                                .GroupLayout.DEFAULT_SIZE, javax.swing.
                                GroupLayout.DEFAULT_SIZE, Short.
                                MAX_VALUE)))
130                 .addGroup(javax.swing.GroupLayout.Alignment.LEADING
                        , layout.createSequentialGroup()
131                     .addComponent(etiquetaConservacion, javax.swing
```

```
                                .GroupLayout.PREFERRED_SIZE, 324, javax.
                                swing.GroupLayout.PREFERRED_SIZE)
132                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.UNRELATED)
133                         .addComponent(valorConservacion, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 73, javax.swing.
                                GroupLayout.PREFERRED_SIZE)))
134                 .addComponent(etiquetaOpciones, javax.swing.GroupLayout
                        .PREFERRED_SIZE, 304, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
135             .addContainerGap(38, Short.MAX_VALUE))
136         );
137         layout.setVerticalGroup(
138             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
139             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
                createSequentialGroup()
140                 .addGap(18, 18, 18)
141                 .addComponent(etiquetaOpciones, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 38, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
142                 .addGap(18, 18, 18)
143                 .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.BASELINE)
144                     .addComponent(etiquetaSegundos, javax.swing.GroupLayout
                        .PREFERRED_SIZE, 33, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
145                     .addComponent(valorSegundos, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 33, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
146                 .addGap(11, 11, 11)
147                 .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.BASELINE)
148                     .addComponent(etiquetaUmbral, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 33, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
```

```
149                     .addComponent(valorUmbral, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 31, javax.swing.GroupLayout.
                            PREFERRED_SIZE))
150                 .addGap(18, 18, 18)
151                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.BASELINE)
152                     .addComponent(etiquetaRuta, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 33, javax.swing.GroupLayout.
                            PREFERRED_SIZE)
153                     .addComponent(botonSeleccionar, javax.swing.GroupLayout
                            .PREFERRED_SIZE, 33, javax.swing.GroupLayout.
                            PREFERRED_SIZE))
154                 .addGap(26, 26, 26)
155                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.BASELINE)
156                     .addComponent(etiquetaConservacion, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 33, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
157                     .addComponent(valorConservacion, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 33, javax.swing.
                            GroupLayout.PREFERRED_SIZE))
158                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .RELATED, 27, Short.MAX_VALUE)
159                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
160                     .addComponent(botonAceptar, javax.swing.GroupLayout.
                            Alignment.TRAILING, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 86, javax.swing.GroupLayout.
                            PREFERRED_SIZE)
161                     .addComponent(botonAtras, javax.swing.GroupLayout.
                            Alignment.TRAILING, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 86, javax.swing.GroupLayout.
                            PREFERRED_SIZE))
162                 .addGap(27, 27, 27))
163         );
164
```

```
165             pack();
166         }// </editor-fold>
167 
168         private void valorConservacionActionPerformed(java.awt.event.
                ActionEvent evt) {
169             // TODO add your handling code here:
170         }
171 
172         private void valorUmbralActionPerformed(java.awt.event.ActionEvent evt)
                {
173             // TODO add your handling code here:
174         }
175 
176         //FUNCION BOTON SELECCIONAR
177         private void botonSeleccionarActionPerformed(java.awt.event.ActionEvent
                evt) {
178 
179             ruta = cargarRuta();
180             Principal.valorUsuarioRuta = ruta;
181             System.out.println("La ruta seleccionada: ");
182             System.out.println("|"+ruta+"|");
183             File archivo = new File(ruta);
184 
185             //COMPROBACION DE SI EXISTE
186             if (archivo.exists() && archivo.isDirectory()) {
187 
188                 Principal.hayRutaUsuario=true;
189 
190             }else{
191 
192                 //ERROR DESEADO
193                 Principal.error = "NO SE HA SELECCIONADO UNA RUTA VALIDA";
194 
195                 //CREACI N VENTANA DE ERROR
196                 VentanaError ventana_error = new VentanaError();
197 
```

```
198             //PONER VISIBLE LA VENTANA
199             ventana_error.setVisible(true);
200 
201             //PONER LA VENTANA DE TAMA O FIJO
202             ventana_error.setResizable(false);
203          }
204      }
205 
206      //FUNCION BOTON ACEPTAR
207      private void botonAceptarActionPerformed(java.awt.event.ActionEvent evt
             ) {
208 
209          boolean sePuede1, sePuede2;
210          boolean ratio ;
211 
212          try{
213 
214             Double.parseDouble(valorUmbral.getText());
215             sePuede1 = true;
216 
217 
218          }catch (NumberFormatException excepcion){
219 
220              sePuede1=false;
221          }
222 
223          try{
224 
225             Double.parseDouble(valorSegundos.getText());
226             sePuede2 = true;
227 
228 
229          }catch (NumberFormatException excepcion){
230 
231              sePuede2=false;
232          }
```

```
233
234         if(sePuede1==true && sePuede2==true){
235
236             if(Double.parseDouble(valorUmbral.getText())>=0 && Double.
                    parseDouble(valorUmbral.getText())<=1){
237
238                 ratio=true;
239
240             }else{
241
242                 ratio=false;
243             }
244
245             if(ratio==true){
246
247                 Principal.valorUsuarioUmbral = Double.parseDouble(
                        valorUmbral.getText());
248
249                 Principal.valorUsuarioSegundosFrame = Double.parseDouble(
                        valorSegundos.getText());
250
251                 Principal.rutaGuardar = Principal.valorUsuarioRuta;
252
253                 Principal.hayRutaConfirmada = Principal.hayRutaUsuario;
254
255                 if(Principal.hayRutaConfirmada == true){
256
257                     Principal.ventana_opciones.setVisible(false);
258
259                     //System.out.println("|"+valorConservacion.
                            getSelectedIndex()+"|");
260
261                     if( valorConservacion.getSelectedIndex() == 0){
262
263                         Principal.valorUsuarioConservar = false;
264
```

```
265                     }else{
266
267                         Principal.valorUsuarioConservar = true;
268                     }
269
270                     Principal.ventana_principal.setVisible(true);
271                     Principal.ventana_principal.setResizable(false);
272
273                 }else{
274
275                     //ERROR DESEADO
276                     Principal.error = "ES NECESARIO SELECCIONAR UNA RUTA
                        PARA LOS ARCHIVOS TEMPORALES, HAGA CLIC EN EL BOTON
                        DE OPCIONES Y POSTERIORMENTE SELECCIONE UNA RUTA";
277
278                     //CREACI N VENTANA DE ERROR
279                     VentanaError ventana_error = new VentanaError();
280
281                     //PONER VISIBLE LA VENTANA
282                     ventana_error.setVisible(true);
283
284                     //PONER LA VENTANA DE TAMA O FIJO
285                     ventana_error.setResizable(false);
286
287                 }
288
289             }else{
290
291                 //ERROR DESEADO
292                 Principal.error = "EL UMBRAL DEBE DE ENCONTRARSE ENTRE 0 Y
                    1";
293
294                 //CREACI N VENTANA DE ERROR
295                 VentanaError ventana_error = new VentanaError();
296
297                 //PONER VISIBLE LA VENTANA
```

```
298                 ventana_error.setVisible(true);
299
300                 //PONER LA VENTANA DE TAMA O FIJO
301                 ventana_error.setResizable(false);
302
303             }
304
305         }else{
306
307             //ERROR DESEADO
308             Principal.error = "LOS 2 PRIMEROS CAMPOS TIENEN QUE CONTENER
                    VALORES NUMERICOS";
309
310             //CREACI N VENTANA DE ERROR
311             VentanaError ventana_error = new VentanaError();
312
313             //PONER VISIBLE LA VENTANA
314             ventana_error.setVisible(true);
315
316             //PONER LA VENTANA DE TAMA O FIJO
317             ventana_error.setResizable(false);
318
319         }
320
321     }
322
323     //FUNCION BOTON ATRAS
324     private void botonAtrasActionPerformed(java.awt.event.ActionEvent evt)
            {
325
326         Principal.ventana_opciones.setVisible(false);
327
328         if(Principal.hayRutaConfirmada == true){
329
330             Principal.ventana_principal.setVisible(true);
331             Principal.ventana_principal.setResizable(false);
```

```
332
333         }else{
334
335             Principal.ventana_usuarios.setVisible(true);
336             Principal.ventana_usuarios.setResizable(false);
337
338         }
339
340     }
341
342     //FUNCIONES PRINCIPAL
343     public static void main(String args[]) {
344
345         try {
346             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
347                 if ("Nimbus".equals(info.getName())) {
348                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
349                     break;
350                 }
351             }
352         } catch (ClassNotFoundException ex) {
353             java.util.logging.Logger.getLogger(VentanaOpciones.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
354         } catch (InstantiationException ex) {
355             java.util.logging.Logger.getLogger(VentanaOpciones.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
356         } catch (IllegalAccessException ex) {
357             java.util.logging.Logger.getLogger(VentanaOpciones.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
358         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
359             java.util.logging.Logger.getLogger(VentanaOpciones.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
360         }
361
```

```
362         java.awt.EventQueue.invokeLater(new Runnable() {
363             public void run() {
364                 new VentanaOpciones().setVisible(true);
365             }
366         });
367     }
368 
369     //FUNCION CARGAR RUTA
370     public String cargarRuta(){
371 
372         //fichero seleccionado
373         JFileChooser fichero = new JFileChooser();
374 
375         //atajos
376         fichero.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
377 
378         //ventana madre para mostrar la ventana de abrir
379         int option = fichero.showDialog(this, "Aceptar");
380 
381         //devuelve directamente el fichero especificado
382         return fichero.getSelectedFile().toString();
383 
384     }
385 
386     // Variables declaration - do not modify
387     private javax.swing.JButton botonAceptar;
388     private javax.swing.JButton botonAtras;
389     private javax.swing.JButton botonSeleccionar;
390     private javax.swing.JLabel etiquetaConservacion;
391     private javax.swing.JLabel etiquetaOpciones;
392     private javax.swing.JLabel etiquetaRuta;
393     private javax.swing.JLabel etiquetaSegundos;
394     private javax.swing.JLabel etiquetaUmbral;
395     private javax.swing.JComboBox<String> valorConservacion;
396     private javax.swing.JTextField valorSegundos;
397     private javax.swing.JTextField valorUmbral;
```

```
398         // End of variables declaration
399 }
```

5.8 VentanaResultadosFinales.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  import PaquetePostAnalisis.FuncionesResultados;
5  import PaqueteAnalisisVideo.Funciones;
6  import PaquetePreAnalisis.Principal;
7  import java.io.IOException;
8  import java.util.ArrayList;
9  import java.util.logging.Level;
10 import java.util.logging.Logger;
11 import javax.swing.Icon;
12 import javax.swing.ImageIcon;
13
14 /**
15  *
16  * @author Angel Murcia Diaz
17  */
18
19 //VENTANA: VENTANARESULTADOSFINALES
20 public class VentanaResultadosFinales extends javax.swing.JFrame {
21
22     //CONSTRUCTOR
23     public VentanaResultadosFinales() {
24         initComponents();
25         actualizar();
26     }
27
28     //FUNCION MOSTRAR MENSAJE
29     public void mostrarMensaje(int tipo){
30
31         if(tipo==1){
32
33             //ERROR (AVISO) DESEADO
34             Principal.error = "HA TERMINADO EL PROCESO DE ESCRITURA EN
```

```
                DATASETUNICO";
35
36              //CREACI N VENTANA DE ERROR
37              VentanaAviso ventana_aviso = new VentanaAviso();
38
39              //PONER LA VENTANA DE TAMA O FIJO
40              ventana_aviso.setResizable(false);
41
42              //PONER VISIBLE LA VENTANA
43              ventana_aviso.setVisible(true);
44
45          }else if (tipo==0){
46
47              //ERROR (AVISO) DESEADO
48              Principal.error = "NO SE PUEDE SOBREESCRIBIR POR ALGUNO DE LOS
                    SIGUIENTES MOTIVOS:"
49                      + " \n 1.PORQUE EL ARCHIVO ESTA ABIERTO, PORFAVOR
                            CIERRELO E INTENTELO DE NUEVO"
50                      + "\n 2.PORQUE EL ARCHIVO SE HA DA ADO , PORFAVOR
                            COMPRUEBELO O CREE UN NUEVO DATASETUNICO EN "
51                      + "OTRA UBICACI N";
52
53              //CREACI N VENTANA DE ERROR
54              VentanaError ventana_error = new VentanaError();
55
56              //PONER LA VENTANA DE TAMA O FIJO
57              ventana_error.setResizable(false);
58
59              //PONER VISIBLE LA VENTANA
60              ventana_error.setVisible(true);
61
62          }
63      }
64
65      //FUNCION PROCESODATASETUNICO
66      public void procesoDatasetUnico(){
```

```
67
68          //clase FuncionesResultados auxiliar para poder realizar todas las
                operaciones
69          FuncionesResultados aux = new FuncionesResultados();
70
71          aux.crearResultadosPorcentajes();
72
73          //Comprobacion de si existe o no el dataset unico para almacenar
                los resultados de todos los videos juntos
74          boolean existe = aux.existeDatasetUnico();
75
76          //Tipo de mensaje a mostrar
77          int tipo = 0 ;
78
79          //Creacion y relleno de la nueva fila de datos que entrar  al
                dataset unico
80          ArrayList<Double> nuevosDatos = new ArrayList<>();
81          nuevosDatos = aux.rellenarVectorDatasetUnico();
82
83          //Creacion y relleno del nombre del video que se introducira en el
                dataset unico
84          String videoNuevo = Principal.nombreFichero;
85
86          //Si ya existe el dataset unico
87          if(existe == true){
88
89              //se cargan los datos del dataset unico existente
90              try {
91                  aux.cargarDatasetUnico();
92              } catch (IOException ex) {
93                  Logger.getLogger(VentanaResultadosFinales.class.getName()).
                        log(Level.SEVERE, null, ex);
94                  System.out.println("gokuu");
95
96              }
97
```

```
98              //Se sobreescribe el dataset unico con los datos anteriores mas
                    los nuevos
99              try {
100                 aux.agregarADatasetUnico(nuevosDatos, videoNuevo);
101                 tipo = 1;
102             } catch (IOException ex) {
103                 Logger.getLogger(VentanaResultadosFinales.class.getName()).
                        log(Level.SEVERE, null, ex);
104             }
105
106         //Si NO existe el dataset unico
107         }else{
108
109             System.out.println("NO EXISTE EL DATASET, POR LO TANTO SE EST
                    CREANDO");
110             //Creamos el dataset unico con la fila de datos de este video
111             try {
112                 aux.crearDatasetUnico(nuevosDatos, videoNuevo);
113                 tipo = 1 ;
114             } catch (IOException ex) {
115                 Logger.getLogger(VentanaResultadosFinales.class.getName()).
                        log(Level.SEVERE, null, ex);
116             }
117
118         }
119
120         mostrarMensaje(tipo);
121
122     }
123
124     //FUNCION ACTUALIZAR
125     public void actualizar(){
126
127         valorAnger.setText(Integer.toString(Principal.angerContador));
128         valorContempt.setText(Integer.toString(Principal.contemptContador))
                ;
```

```
129         valorDisgust.setText(Integer.toString(Principal.disgustContador));
130         valorFear.setText(Integer.toString(Principal.fearContador));
131         valorHappiness.setText(Integer.toString(Principal.happinessContador
                ));
132         valorNeutral.setText(Integer.toString(Principal.neutralContador));
133         valorSadness.setText(Integer.toString(Principal.sadnessContador));
134         valorSurprise.setText(Integer.toString(Principal.surpriseContador))
                ;
135         valorNo.setText(Integer.toString(Principal.noAnalizadoContador));
136         valorTotal.setText(Integer.toString(Principal.nFotosAnalizadas));
137         etiquetaNo.setText("not analysed");
138         etiquetaTotal.setText("total analysed");
139 
140     }
141 
142     //FUNCION ACTUALIZARPORCENTAJES
143     public void actualizarPorcentajes(){
144 
145         valorAnger.setText(String.format("%.2f", Principal.
                angerContadorPorcentaje));
146         valorContempt.setText(String.format("%.2f",Principal.
                contemptContadorPorcentaje));
147         valorDisgust.setText(String.format("%.2f",Principal.
                disgustContadorPorcentaje));
148         valorFear.setText(String.format("%.2f",Principal.
                fearContadorPorcentaje));
149         valorHappiness.setText(String.format("%.2f",Principal.
                happinessContadorPorcentaje));
150         valorNeutral.setText(String.format("%.2f",Principal.
                neutralContadorPorcentaje));
151         valorSadness.setText(String.format("%.2f",Principal.
                sadnessContadorPorcentaje));
152         valorSurprise.setText(String.format("%.2f",Principal.
                surpriseContadorPorcentaje));
153         etiquetaNo.setText("Useful");
154         valorNo.setText(Integer.toString(Principal.nFotosAnalizadas -
```

```
            Principal.noAnalizadoContador));
155         valorTotal.setText(" ");
156         etiquetaTotal.setText("");
157
158     }
159
160
161     @SuppressWarnings("unchecked")
162     // <editor-fold defaultstate="collapsed" desc="Generated Code">
163     private void initComponents() {
164
165         etiquetaDisgust = new javax.swing.JLabel();
166         etiquetaAnger = new javax.swing.JLabel();
167         etiquetaContempt = new javax.swing.JLabel();
168         etiquetaHappiness = new javax.swing.JLabel();
169         etiquetaFear = new javax.swing.JLabel();
170         etiquetaNeutral = new javax.swing.JLabel();
171         etiquetaNo = new javax.swing.JLabel();
172         etiquetaSadness = new javax.swing.JLabel();
173         etiquetaSurprise = new javax.swing.JLabel();
174         etiquetaTotal = new javax.swing.JLabel();
175         valorAnger = new javax.swing.JLabel();
176         valorContempt = new javax.swing.JLabel();
177         valorSadness = new javax.swing.JLabel();
178         valorDisgust = new javax.swing.JLabel();
179         valorFear = new javax.swing.JLabel();
180         valorHappiness = new javax.swing.JLabel();
181         valorNeutral = new javax.swing.JLabel();
182         valorTotal = new javax.swing.JLabel();
183         valorSurprise = new javax.swing.JLabel();
184         valorNo = new javax.swing.JLabel();
185         etiquetaTitulo = new javax.swing.JLabel();
186         botonXLSX = new javax.swing.JButton();
187         resultadosTXT = new javax.swing.JButton();
188         resultadosAtras = new javax.swing.JButton();
189         botonPorcentajes = new javax.swing.JButton();
```

```
190         botonTotales = new javax.swing.JButton();
191         datasetTotal = new javax.swing.JButton();
192 
193         setDefaultCloseOperation(javax.swing.WindowConstants.
                DISPOSE_ON_CLOSE);
194         setTitle("VENTANA DE RESULTADOS");
195 
196         etiquetaDisgust.setText("disgust");
197 
198         etiquetaAnger.setText("anger");
199 
200         etiquetaContempt.setText("contempt");
201 
202         etiquetaHappiness.setText("happiness");
203 
204         etiquetaFear.setText("fear");
205 
206         etiquetaNeutral.setText("neutral");
207 
208         etiquetaNo.setText("not analysed");
209 
210         etiquetaSadness.setText("sadness");
211 
212         etiquetaSurprise.setText("surprise");
213 
214         etiquetaTotal.setText("total analysed");
215 
216         etiquetaTitulo.setFont(new java.awt.Font("Tahoma", 1, 14)); //
                NOI18N
217         etiquetaTitulo.setText("RESULTADOS FINALES");
218 
219         botonXLSX.setFont(new java.awt.Font("Candara", 1, 14)); // NOI18N
220         botonXLSX.setIcon(new javax.swing.ImageIcon(getClass().getResource(
                "/PaqueteImagenes/disquete32.png"))); // NOI18N
221         botonXLSX.setText("A XLSX");
222         botonXLSX.setHorizontalTextPosition(javax.swing.SwingConstants.
```

```
            CENTER);
223         botonXLSX.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
            );
224         botonXLSX.addActionListener(new java.awt.event.ActionListener() {
225             public void actionPerformed(java.awt.event.ActionEvent evt) {
226                 botonXLSXActionPerformed(evt);
227             }
228         });
229
230         resultadosTXT.setFont(new java.awt.Font("Candara", 1, 14)); //
            NOI18N
231         resultadosTXT.setIcon(new javax.swing.ImageIcon(getClass().
            getResource("/PaqueteImagenes/disquete32.png"))); // NOI18N
232         resultadosTXT.setText("A TXT");
233         resultadosTXT.setHorizontalTextPosition(javax.swing.SwingConstants.
            CENTER);
234         resultadosTXT.setVerticalTextPosition(javax.swing.SwingConstants.
            BOTTOM);
235         resultadosTXT.addActionListener(new java.awt.event.ActionListener()
             {
236             public void actionPerformed(java.awt.event.ActionEvent evt) {
237                 resultadosTXTActionPerformed(evt);
238             }
239         });
240
241         resultadosAtras.setFont(new java.awt.Font("Candara", 1, 14)); //
            NOI18N
242         resultadosAtras.setIcon(new javax.swing.ImageIcon(getClass().
            getResource("/PaqueteImagenes/atras32.png"))); // NOI18N
243         resultadosAtras.setText("ATR S");
244         resultadosAtras.setHorizontalTextPosition(javax.swing.
            SwingConstants.CENTER);
245         resultadosAtras.setVerticalTextPosition(javax.swing.SwingConstants.
            BOTTOM);
246         resultadosAtras.addActionListener(new java.awt.event.ActionListener
            () {
```

```
247             public void actionPerformed(java.awt.event.ActionEvent evt) {
248                 resultadosAtrasActionPerformed(evt);
249             }
250         });
251 
252         botonPorcentajes.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
253         botonPorcentajes.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/formato32.png"))); // NOI18N
254         botonPorcentajes.setText("%");
255         botonPorcentajes.setHorizontalTextPosition(javax.swing.
                SwingConstants.CENTER);
256         botonPorcentajes.setVerticalTextPosition(javax.swing.SwingConstants
                .BOTTOM);
257         botonPorcentajes.addActionListener(new java.awt.event.
                ActionListener() {
258             public void actionPerformed(java.awt.event.ActionEvent evt) {
259                 botonPorcentajesActionPerformed(evt);
260             }
261         });
262 
263         botonTotales.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
264         botonTotales.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/formato32.png"))); // NOI18N
265         botonTotales.setText("totales");
266         botonTotales.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
267         botonTotales.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
268         botonTotales.addActionListener(new java.awt.event.ActionListener()
                {
269             public void actionPerformed(java.awt.event.ActionEvent evt) {
270                 botonTotalesActionPerformed(evt);
271             }
272         });
```

```
273
274         datasetTotal.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
275         datasetTotal.setText("A ADIR A DATASET UNICO");
276         datasetTotal.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
277         datasetTotal.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
278         datasetTotal.addActionListener(new java.awt.event.ActionListener()
                {
279             public void actionPerformed(java.awt.event.ActionEvent evt) {
280                 datasetTotalActionPerformed(evt);
281             }
282         });
283
284         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
285         getContentPane().setLayout(layout);
286         layout.setHorizontalGroup(
287             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
288             .addGroup(layout.createSequentialGroup()
289                 .addGap(21, 21, 21)
290                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
291                     .addGroup(layout.createSequentialGroup()
292                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING, false)
293                             .addGroup(layout.createSequentialGroup()
294                                 .addComponent(etiquetaFear, javax.swing.
                                        GroupLayout.PREFERRED_SIZE, 71, javax.
                                        swing.GroupLayout.PREFERRED_SIZE)
295                                 .addPreferredGap(javax.swing.LayoutStyle.
                                        ComponentPlacement.RELATED)
296                                 .addComponent(valorFear, javax.swing.
                                        GroupLayout.DEFAULT_SIZE, javax.swing.
```

```
                GroupLayout.DEFAULT_SIZE, Short.
                MAX_VALUE))
297         .addGroup(layout.createSequentialGroup()
298             .addComponent(etiquetaDisgust, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 71, javax.
                    swing.GroupLayout.PREFERRED_SIZE)
299             .addPreferredGap(javax.swing.LayoutStyle.
                    ComponentPlacement.RELATED)
300             .addComponent(valorDisgust, javax.swing.
                    GroupLayout.DEFAULT_SIZE, javax.swing.
                    GroupLayout.DEFAULT_SIZE, Short.
                    MAX_VALUE))
301         .addGroup(layout.createSequentialGroup()
302             .addComponent(etiquetaContempt, javax.swing
                    .GroupLayout.PREFERRED_SIZE, 71, javax.
                    swing.GroupLayout.PREFERRED_SIZE)
303             .addPreferredGap(javax.swing.LayoutStyle.
                    ComponentPlacement.RELATED)
304             .addComponent(valorContempt, javax.swing.
                    GroupLayout.DEFAULT_SIZE, javax.swing.
                    GroupLayout.DEFAULT_SIZE, Short.
                    MAX_VALUE))
305         .addGroup(layout.createSequentialGroup()
306             .addComponent(etiquetaAnger, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 71, javax.
                    swing.GroupLayout.PREFERRED_SIZE)
307             .addPreferredGap(javax.swing.LayoutStyle.
                    ComponentPlacement.RELATED)
308             .addComponent(valorAnger, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 55, javax.
                    swing.GroupLayout.PREFERRED_SIZE))
309         .addGroup(layout.createSequentialGroup()
310             .addComponent(etiquetaHappiness, javax.
                    swing.GroupLayout.PREFERRED_SIZE, 71,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
311             .addPreferredGap(javax.swing.LayoutStyle.
```

```
                        ComponentPlacement.RELATED)
312                     .addComponent(valorHappiness, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 63, javax.
                            swing.GroupLayout.PREFERRED_SIZE)))
313                 .addPreferredGap(javax.swing.LayoutStyle.
                        ComponentPlacement.RELATED)
314                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
315                     .addComponent(etiquetaNo, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.
                            GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
316                     .addComponent(etiquetaTotal, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.
                            GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
317                     .addComponent(etiquetaSurprise, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.
                            GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
318                     .addComponent(etiquetaSadness, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.
                            GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
319                     .addComponent(etiquetaNeutral, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.
                            GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
320                 .addPreferredGap(javax.swing.LayoutStyle.
                        ComponentPlacement.RELATED, javax.swing.
                        GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
321                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING, false)
322                     .addComponent(valorNo, javax.swing.GroupLayout.
                            DEFAULT_SIZE, 55, Short.MAX_VALUE)
323                     .addComponent(valorSurprise, javax.swing.
                            GroupLayout.Alignment.TRAILING, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.
                            GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
324                     .addComponent(valorSadness, javax.swing.
                            GroupLayout.Alignment.TRAILING, javax.swing.
```

```
                                GroupLayout.DEFAULT_SIZE, javax.swing.
                                GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
325                         .addComponent(valorNeutral, javax.swing.
                                GroupLayout.Alignment.TRAILING, javax.swing.
                                GroupLayout.DEFAULT_SIZE, javax.swing.
                                GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
326                         .addComponent(valorTotal, javax.swing.
                                GroupLayout.DEFAULT_SIZE, javax.swing.
                                GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
327                 .addComponent(etiquetaTitulo, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 231, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
328             .addGap(22, 22, 22))
329         .addGroup(layout.createSequentialGroup()
330             .addContainerGap()
331             .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.TRAILING, false)
332                 .addComponent(datasetTotal, javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE)
333                 .addGroup(layout.createSequentialGroup()
334                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.LEADING)
335                         .addGroup(layout.createSequentialGroup()
336                             .addComponent(botonTotales, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 99, javax.
                                    swing.GroupLayout.PREFERRED_SIZE)
337                             .addGap(18, 18, 18)
338                             .addComponent(botonPorcentajes, javax.swing
                                    .GroupLayout.PREFERRED_SIZE, 99, javax.
                                    swing.GroupLayout.PREFERRED_SIZE))
339                         .addGroup(layout.createSequentialGroup()
340                             .addComponent(botonXLSX, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 99, javax.
                                    swing.GroupLayout.PREFERRED_SIZE)
341                             .addGap(18, 18, 18)
```

```
342                                 .addComponent(resultadosTXT, javax.swing.
                                        GroupLayout.PREFERRED_SIZE, 99, javax.
                                        swing.GroupLayout.PREFERRED_SIZE)))
343                         .addGap(18, 18, 18)
344                         .addComponent(resultadosAtras, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 95, javax.swing.
                                GroupLayout.PREFERRED_SIZE)))
345                 .addContainerGap(14, Short.MAX_VALUE))
346         );
347         layout.setVerticalGroup(
348             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
349             .addGroup(layout.createSequentialGroup()
350                 .addGap(10, 10, 10)
351                 .addComponent(etiquetaTitulo, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 30, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
352                 .addGap(18, 18, 18)
353                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
354                     .addGroup(layout.createSequentialGroup()
355                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING)
356                             .addComponent(etiquetaAnger, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                    GroupLayout.PREFERRED_SIZE)
357                             .addComponent(valorAnger, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                    GroupLayout.PREFERRED_SIZE))
358                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
359                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.TRAILING)
360                             .addComponent(etiquetaContempt, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                    GroupLayout.PREFERRED_SIZE)
```

```
361                         .addComponent(valorContempt, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE))
362                     .addPreferredGap(javax.swing.LayoutStyle.
                            ComponentPlacement.RELATED)
363                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.LEADING)
364                         .addComponent(etiquetaDisgust, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
365                         .addComponent(valorDisgust, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE))
366                     .addPreferredGap(javax.swing.LayoutStyle.
                            ComponentPlacement.RELATED)
367                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.LEADING)
368                         .addComponent(valorFear, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
369                         .addComponent(etiquetaFear, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE))
370                     .addPreferredGap(javax.swing.LayoutStyle.
                            ComponentPlacement.RELATED)
371                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.LEADING)
372                         .addComponent(etiquetaHappiness, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
373                         .addComponent(valorHappiness, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE)))
374                 .addGroup(layout.createSequentialGroup()
375                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.LEADING)
```

```
376                 .addComponent(etiquetaNeutral, javax.swing.
                        GroupLayout.Alignment.TRAILING, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                        GroupLayout.PREFERRED_SIZE)
377                 .addComponent(valorNeutral, javax.swing.
                        GroupLayout.Alignment.TRAILING, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
378             .addPreferredGap(javax.swing.LayoutStyle.
                    ComponentPlacement.RELATED)
379             .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.TRAILING)
380                 .addComponent(etiquetaSadness, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                        GroupLayout.PREFERRED_SIZE)
381                 .addComponent(valorSadness, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
382             .addPreferredGap(javax.swing.LayoutStyle.
                    ComponentPlacement.RELATED)
383             .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.LEADING)
384                 .addComponent(etiquetaSurprise, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                        GroupLayout.PREFERRED_SIZE)
385                 .addComponent(valorSurprise, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
386             .addPreferredGap(javax.swing.LayoutStyle.
                    ComponentPlacement.RELATED)
387             .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.LEADING)
388                 .addComponent(etiquetaNo, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                        GroupLayout.PREFERRED_SIZE)
389                 .addComponent(valorNo, javax.swing.GroupLayout.
```

```
                                PREFERRED_SIZE, 30, javax.swing.GroupLayout.
                                PREFERRED_SIZE))
390                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
391                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING)
392                             .addComponent(etiquetaTotal, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                    GroupLayout.PREFERRED_SIZE)
393                             .addComponent(valorTotal, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                    GroupLayout.PREFERRED_SIZE))))
394                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
395                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
396                     .addGroup(layout.createSequentialGroup()
397                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING, false)
398                             .addComponent(botonPorcentajes, javax.swing.
                                    GroupLayout.DEFAULT_SIZE, javax.swing.
                                    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
399                             .addComponent(botonTotales, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 74, javax.swing.
                                    GroupLayout.PREFERRED_SIZE))
400                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.UNRELATED)
401                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING, false)
402                             .addComponent(resultadosTXT, javax.swing.
                                    GroupLayout.DEFAULT_SIZE, javax.swing.
                                    GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
403                             .addComponent(botonXLSX, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 74, javax.swing.
                                    GroupLayout.PREFERRED_SIZE)))
404                 .addComponent(resultadosAtras, javax.swing.GroupLayout.
```

```
                        PREFERRED_SIZE, 159, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
405                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
406                 .addComponent(datasetTotal, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 36, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
407                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
408         );
409
410         pack();
411     }// </editor-fold>
412
413     //FUNCION BOTON XLSX
414     private void botonXLSXActionPerformed(java.awt.event.ActionEvent evt) {
415
416         new FuncionesResultados().crearDataset(Principal.formatoResultado);
417
418     }
419
420     //FUNCION BOTON TXT
421     private void resultadosTXTActionPerformed(java.awt.event.ActionEvent
            evt) {
422         try {
423             // TODO add your handling code here:
424
425             new FuncionesResultados().resultadosTXT(Principal.
                    formatoResultado);
426         } catch (IOException ex) {
427             Logger.getLogger(VentanaResultadosFinales.class.getName()).log(
                    Level.SEVERE, null, ex);
428         }
429     }
430
431     //FUNCION BOTON ATRAS
```

```
432     private void resultadosAtrasActionPerformed(java.awt.event.ActionEvent
            evt) {
433         // TODO add your handling code here:
434         Principal.ventana_resultados.setVisible(false);
435 
436     }
437 
438     //FUNCION BOTON %
439     private void botonPorcentajesActionPerformed(java.awt.event.ActionEvent
             evt) {
440 
441         new FuncionesResultados().crearResultadosPorcentajes();
442         actualizarPorcentajes();
443         Principal.formatoResultado=2;
444         new Funciones().imprimir("Porcentajes", "Porcentajes");
445 
446     }
447 
448     //FUNCION BOTON TOTALES
449     private void botonTotalesActionPerformed(java.awt.event.ActionEvent evt
            ) {
450 
451         actualizar();
452         Principal.formatoResultado=1;
453     }
454 
455     //FUNCION DATASETTOTAL
456     private void datasetTotalActionPerformed(java.awt.event.ActionEvent evt
            ) {
457 
458         procesoDatasetUnico();
459 
460     }
461 
462     //FUNCION PRINCIPAL
463     public static void main(String args[]) {
```

```
464
465         try {
466             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
467                 if ("Nimbus".equals(info.getName())) {
468                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
469                     break;
470                 }
471             }
472         } catch (ClassNotFoundException ex) {
473             java.util.logging.Logger.getLogger(VentanaResultadosFinales.
                    class.getName()).log(java.util.logging.Level.SEVERE, null,
                    ex);
474         } catch (InstantiationException ex) {
475             java.util.logging.Logger.getLogger(VentanaResultadosFinales.
                    class.getName()).log(java.util.logging.Level.SEVERE, null,
                    ex);
476         } catch (IllegalAccessException ex) {
477             java.util.logging.Logger.getLogger(VentanaResultadosFinales.
                    class.getName()).log(java.util.logging.Level.SEVERE, null,
                    ex);
478         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
479             java.util.logging.Logger.getLogger(VentanaResultadosFinales.
                    class.getName()).log(java.util.logging.Level.SEVERE, null,
                    ex);
480         }
481
482         java.awt.EventQueue.invokeLater(new Runnable() {
483             public void run() {
484                 new VentanaResultadosFinales().setVisible(true);
485             }
486         });
487     }
488
489     // Variables declaration - do not modify
```

```
490     private javax.swing.JButton botonPorcentajes;
491     private javax.swing.JButton botonTotales;
492     private javax.swing.JButton botonXLSX;
493     private javax.swing.JButton datasetTotal;
494     private javax.swing.JLabel etiquetaAnger;
495     private javax.swing.JLabel etiquetaContempt;
496     private javax.swing.JLabel etiquetaDisgust;
497     private javax.swing.JLabel etiquetaFear;
498     private javax.swing.JLabel etiquetaHappiness;
499     private javax.swing.JLabel etiquetaNeutral;
500     private javax.swing.JLabel etiquetaNo;
501     private javax.swing.JLabel etiquetaSadness;
502     private javax.swing.JLabel etiquetaSurprise;
503     private javax.swing.JLabel etiquetaTitulo;
504     private javax.swing.JLabel etiquetaTotal;
505     private javax.swing.JButton resultadosAtras;
506     private javax.swing.JButton resultadosTXT;
507     private javax.swing.JLabel valorAnger;
508     private javax.swing.JLabel valorContempt;
509     private javax.swing.JLabel valorDisgust;
510     private javax.swing.JLabel valorFear;
511     private javax.swing.JLabel valorHappiness;
512     private javax.swing.JLabel valorNeutral;
513     private javax.swing.JLabel valorNo;
514     private javax.swing.JLabel valorSadness;
515     private javax.swing.JLabel valorSurprise;
516     private javax.swing.JLabel valorTotal;
517     // End of variables declaration
518 }
```

5.9 VentanaSeguimiento.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIOS
5  import PaquetePreAnalisis.Principal;
6  import javax.swing.Icon;
7  import javax.swing.ImageIcon;
8
9  /**
10  *
11  * @author Angel Murcia Diaz
12  */
13
14 //VENTANA: VENTANASEGUIMIENTO
15 public class VentanaSeguimiento extends javax.swing.JFrame {
16
17     //CONSTRUCTOR
18     public VentanaSeguimiento() {
19         initComponents();
20         actualizar();
21     }
22
23     //FUNCION ACTUALIZAR
24     public void actualizar(){
25
26         valorAnger.setText(Integer.toString(Principal.angerContador));
27         valorContempt.setText(Integer.toString(Principal.contemptContador))
               ;
28         valorDisgust.setText(Integer.toString(Principal.disgustContador));
29         valorFear.setText(Integer.toString(Principal.fearContador));
30         valorHappiness.setText(Integer.toString(Principal.happinessContador
               ));
31         valorNeutral.setText(Integer.toString(Principal.neutralContador));
32         valorSadness.setText(Integer.toString(Principal.sadnessContador));
```

```
33          valorSurprise.setText(Integer.toString(Principal.surpriseContador))
                ;
34          valorNo.setText(Integer.toString(Principal.noAnalizadoContador));
35          valorTotal.setText(Integer.toString(Principal.nFotosAnalizadas));
36
37          if(Principal.estaEnPausa == false){
38
39              valorEstado.setIcon(new javax.swing.ImageIcon(getClass().
                    getResource("/PaqueteImagenes/play64.png")));
40              etiquetaEstado.setText("Estado: Analizando");
41
42          }else{
43
44              valorEstado.setIcon(new javax.swing.ImageIcon(getClass().
                    getResource("/PaqueteImagenes/espera64.png")));
45              etiquetaEstado.setText("Estado: En espera");
46
47          }
48
49      }
50
51
52      @SuppressWarnings("unchecked")
53      // <editor-fold defaultstate="collapsed" desc="Generated Code">
54      private void initComponents() {
55
56          etiquetaDisgust = new javax.swing.JLabel();
57          etiquetaAnger = new javax.swing.JLabel();
58          etiquetaContempt = new javax.swing.JLabel();
59          etiquetaHappiness = new javax.swing.JLabel();
60          etiquetaFear = new javax.swing.JLabel();
61          etiquetaNeutral = new javax.swing.JLabel();
62          etiquetaNo = new javax.swing.JLabel();
63          etiquetaSadness = new javax.swing.JLabel();
64          etiquetaSurprise = new javax.swing.JLabel();
65          etiquetaTotal = new javax.swing.JLabel();
```

```
66          valorAnger = new javax.swing.JLabel();
67          valorContempt = new javax.swing.JLabel();
68          valorSadness = new javax.swing.JLabel();
69          valorDisgust = new javax.swing.JLabel();
70          valorFear = new javax.swing.JLabel();
71          valorHappiness = new javax.swing.JLabel();
72          valorNeutral = new javax.swing.JLabel();
73          valorTotal = new javax.swing.JLabel();
74          valorSurprise = new javax.swing.JLabel();
75          valorNo = new javax.swing.JLabel();
76          botonActualizar = new javax.swing.JButton();
77          etiquetaTitulo = new javax.swing.JLabel();
78          valorEstado = new javax.swing.JLabel();
79          etiquetaEstado = new javax.swing.JLabel();
80          botonTerminar = new javax.swing.JButton();
81
82          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
                ;
83          setTitle("VENTANA DE SEGUIMIENTO");
84
85          etiquetaDisgust.setText("disgust");
86
87          etiquetaAnger.setText("anger");
88
89          etiquetaContempt.setText("contempt");
90
91          etiquetaHappiness.setText("happiness");
92
93          etiquetaFear.setText("fear");
94
95          etiquetaNeutral.setText("neutral");
96
97          etiquetaNo.setText("not analysed");
98
99          etiquetaSadness.setText("sadness");
100
```

```
101         etiquetaSurprise.setText("surprise");
102
103         etiquetaTotal.setText("total analysed");
104
105         botonActualizar.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
106         botonActualizar.setText("ACTUALIZAR PROGRESO");
107         botonActualizar.addActionListener(new java.awt.event.ActionListener
                () {
108             public void actionPerformed(java.awt.event.ActionEvent evt) {
109                 botonActualizarActionPerformed(evt);
110             }
111         });
112
113         etiquetaTitulo.setFont(new java.awt.Font("Tahoma", 1, 14)); //
                NOI18N
114         etiquetaTitulo.setText("RESULTADOS PROVISIONALES");
115
116         valorEstado.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/play64.png"))); // NOI18N
117
118         etiquetaEstado.setText("Estado:");
119
120         botonTerminar.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
121         botonTerminar.setText("TERMINAR PROCESO");
122         botonTerminar.addActionListener(new java.awt.event.ActionListener()
                {
123             public void actionPerformed(java.awt.event.ActionEvent evt) {
124                 botonTerminarActionPerformed(evt);
125             }
126         });
127
128         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
129         getContentPane().setLayout(layout);
```

```
130         layout.setHorizontalGroup(
131             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
132             .addGroup(layout.createSequentialGroup()
133                 .addGap(21, 21, 21)
134                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
135                     .addGroup(layout.createSequentialGroup()
136                         .addComponent(etiquetaTitulo, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 231, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
137                         .addContainerGap(109, Short.MAX_VALUE))
138                     .addGroup(layout.createSequentialGroup()
139                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING, false)
140                             .addGroup(layout.createSequentialGroup()
141                                 .addComponent(etiquetaFear, javax.swing.
                                        GroupLayout.PREFERRED_SIZE, 71, javax.
                                        swing.GroupLayout.PREFERRED_SIZE)
142                                 .addPreferredGap(javax.swing.LayoutStyle.
                                        ComponentPlacement.RELATED)
143                                 .addComponent(valorFear, javax.swing.
                                        GroupLayout.DEFAULT_SIZE, javax.swing.
                                        GroupLayout.DEFAULT_SIZE, Short.
                                        MAX_VALUE))
144                             .addGroup(layout.createSequentialGroup()
145                                 .addComponent(etiquetaDisgust, javax.swing.
                                        GroupLayout.PREFERRED_SIZE, 71, javax.
                                        swing.GroupLayout.PREFERRED_SIZE)
146                                 .addPreferredGap(javax.swing.LayoutStyle.
                                        ComponentPlacement.RELATED)
147                                 .addComponent(valorDisgust, javax.swing.
                                        GroupLayout.DEFAULT_SIZE, javax.swing.
                                        GroupLayout.DEFAULT_SIZE, Short.
                                        MAX_VALUE))
148                             .addGroup(layout.createSequentialGroup()
```

```
149                         .addComponent(etiquetaContempt, javax.swing
                                .GroupLayout.PREFERRED_SIZE, 71, javax.
                                swing.GroupLayout.PREFERRED_SIZE)
150                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
151                         .addComponent(valorContempt, javax.swing.
                                GroupLayout.DEFAULT_SIZE, javax.swing.
                                GroupLayout.DEFAULT_SIZE, Short.
                                MAX_VALUE))
152                     .addGroup(layout.createSequentialGroup()
153                         .addComponent(etiquetaAnger, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 71, javax.
                                swing.GroupLayout.PREFERRED_SIZE)
154                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
155                         .addComponent(valorAnger, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 55, javax.
                                swing.GroupLayout.PREFERRED_SIZE))
156                     .addGroup(layout.createSequentialGroup()
157                         .addComponent(etiquetaHappiness, javax.
                                swing.GroupLayout.PREFERRED_SIZE, 71,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
158                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
159                         .addComponent(valorHappiness, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 63, javax.
                                swing.GroupLayout.PREFERRED_SIZE))
160                     .addComponent(etiquetaEstado, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.
                            GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
161                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
162                     .addGroup(layout.createSequentialGroup()
163                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
164                         .addGroup(layout.createParallelGroup(javax.
```

```
                swing.GroupLayout.Alignment.LEADING)
165              .addComponent(etiquetaNo, javax.swing.
                     GroupLayout.DEFAULT_SIZE, javax.
                     swing.GroupLayout.DEFAULT_SIZE,
                     Short.MAX_VALUE)
166              .addComponent(etiquetaTotal, javax.
                     swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE
                     , Short.MAX_VALUE)
167              .addComponent(etiquetaSurprise, javax.
                     swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE
                     , Short.MAX_VALUE)
168              .addComponent(etiquetaSadness, javax.
                     swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE
                     , Short.MAX_VALUE)
169              .addComponent(etiquetaNeutral, javax.
                     swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE
                     , Short.MAX_VALUE))
170          .addPreferredGap(javax.swing.LayoutStyle.
                 ComponentPlacement.RELATED)
171          .addGroup(layout.createParallelGroup(javax.
                 swing.GroupLayout.Alignment.LEADING)
172              .addComponent(valorSadness, javax.swing
                     .GroupLayout.DEFAULT_SIZE, javax.
                     swing.GroupLayout.DEFAULT_SIZE,
                     Short.MAX_VALUE)
173              .addComponent(valorNo, javax.swing.
                     GroupLayout.DEFAULT_SIZE, javax.
                     swing.GroupLayout.DEFAULT_SIZE,
                     Short.MAX_VALUE)
174              .addComponent(valorSurprise, javax.
                     swing.GroupLayout.DEFAULT_SIZE,
                     javax.swing.GroupLayout.DEFAULT_SIZE
```

```
                                        , Short.MAX_VALUE)
175                                     .addComponent(valorTotal, javax.swing.
                                            GroupLayout.DEFAULT_SIZE, javax.
                                            swing.GroupLayout.DEFAULT_SIZE,
                                            Short.MAX_VALUE)
176                                     .addComponent(valorNeutral, javax.swing
                                            .GroupLayout.DEFAULT_SIZE, javax.
                                            swing.GroupLayout.DEFAULT_SIZE,
                                            Short.MAX_VALUE))
177                                 .addGap(22, 22, 22))
178                             .addGroup(javax.swing.GroupLayout.Alignment.
                                    TRAILING, layout.createSequentialGroup()
179                                 .addPreferredGap(javax.swing.LayoutStyle.
                                        ComponentPlacement.RELATED, javax.swing.
                                        GroupLayout.DEFAULT_SIZE, Short.
                                        MAX_VALUE)
180                                 .addComponent(valorEstado)
181                                 .addGap(54, 54, 54))))
182                 .addGroup(layout.createSequentialGroup()
183                     .addGroup(layout.createParallelGroup(javax.swing.
                            GroupLayout.Alignment.LEADING)
184                         .addComponent(botonActualizar, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 316, javax.swing
                                .GroupLayout.PREFERRED_SIZE)
185                         .addComponent(botonTerminar, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 316, javax.swing
                                .GroupLayout.PREFERRED_SIZE))
186                     .addGap(0, 0, Short.MAX_VALUE))))
187         );
188         layout.setVerticalGroup(
189             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
190             .addGroup(layout.createSequentialGroup()
191                 .addGap(10, 10, 10)
192                 .addComponent(etiquetaTitulo, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 30, javax.swing.GroupLayout.
```

```
                PREFERRED_SIZE)
193         .addGap(18, 18, 18)
194         .addGroup(layout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.TRAILING)
195             .addGroup(layout.createSequentialGroup()
196                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
197                     .addComponent(etiquetaAnger, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
198                     .addComponent(valorAnger, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                            GroupLayout.PREFERRED_SIZE))
199                 .addPreferredGap(javax.swing.LayoutStyle.
                        ComponentPlacement.RELATED)
200                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.TRAILING)
201                     .addComponent(etiquetaContempt, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
202                     .addComponent(valorContempt, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                            GroupLayout.PREFERRED_SIZE))
203                 .addPreferredGap(javax.swing.LayoutStyle.
                        ComponentPlacement.RELATED)
204                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
205                     .addComponent(etiquetaDisgust, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
206                     .addComponent(valorDisgust, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                            GroupLayout.PREFERRED_SIZE))
207                 .addPreferredGap(javax.swing.LayoutStyle.
                        ComponentPlacement.RELATED)
208                 .addGroup(layout.createParallelGroup(javax.swing.
```

```
                GroupLayout.Alignment.LEADING)
209              .addComponent(valorFear, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                    GroupLayout.PREFERRED_SIZE)
210              .addComponent(etiquetaFear, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                    GroupLayout.PREFERRED_SIZE))
211          .addPreferredGap(javax.swing.LayoutStyle.
                ComponentPlacement.RELATED)
212          .addGroup(layout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.LEADING)
213              .addComponent(etiquetaHappiness, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                    GroupLayout.PREFERRED_SIZE)
214              .addComponent(valorHappiness, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                    GroupLayout.PREFERRED_SIZE)))
215      .addGroup(layout.createSequentialGroup()
216          .addComponent(etiquetaNeutral, javax.swing.
                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                GroupLayout.PREFERRED_SIZE)
217          .addPreferredGap(javax.swing.LayoutStyle.
                ComponentPlacement.RELATED)
218          .addComponent(etiquetaSadness, javax.swing.
                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                GroupLayout.PREFERRED_SIZE)
219          .addPreferredGap(javax.swing.LayoutStyle.
                ComponentPlacement.RELATED)
220          .addComponent(etiquetaSurprise, javax.swing.
                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                GroupLayout.PREFERRED_SIZE)
221          .addPreferredGap(javax.swing.LayoutStyle.
                ComponentPlacement.RELATED)
222          .addComponent(etiquetaNo, javax.swing.GroupLayout.
                PREFERRED_SIZE, 30, javax.swing.GroupLayout.
                PREFERRED_SIZE)
```

```
223                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
224                         .addComponent(etiquetaTotal, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE))
225                     .addGroup(layout.createSequentialGroup()
226                         .addComponent(valorNeutral, javax.swing.GroupLayout
                                .PREFERRED_SIZE, 30, javax.swing.GroupLayout.
                                PREFERRED_SIZE)
227                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
228                         .addComponent(valorSadness, javax.swing.GroupLayout
                                .PREFERRED_SIZE, 30, javax.swing.GroupLayout.
                                PREFERRED_SIZE)
229                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
230                         .addComponent(valorSurprise, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
231                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
232                         .addComponent(valorNo, javax.swing.GroupLayout.
                                PREFERRED_SIZE, 30, javax.swing.GroupLayout.
                                PREFERRED_SIZE)
233                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
234                         .addComponent(valorTotal, javax.swing.GroupLayout.
                                PREFERRED_SIZE, 30, javax.swing.GroupLayout.
                                PREFERRED_SIZE)))
235                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
236                     .addGroup(layout.createSequentialGroup()
237                         .addGap(34, 34, 34)
238                         .addComponent(etiquetaEstado, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 30, javax.swing.
                                GroupLayout.PREFERRED_SIZE))
```

```
239                     .addGroup(layout.createSequentialGroup()
240                         .addPreferredGap(javax.swing.LayoutStyle.
                                ComponentPlacement.RELATED)
241                         .addComponent(valorEstado, javax.swing.GroupLayout.
                                PREFERRED_SIZE, 71, javax.swing.GroupLayout.
                                PREFERRED_SIZE)))
242                 .addGap(18, 18, 18)
243                 .addComponent(botonActualizar, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 67, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
244                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
245                 .addComponent(botonTerminar, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 67, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
246                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
247         );
248
249         pack();
250     }// </editor-fold>
251
252     //FUNCION BOTON ACTUALIZAR
253     private void botonActualizarActionPerformed(java.awt.event.ActionEvent
            evt) {
254         // TODO add your handling code here:
255         actualizar();
256     }
257
258     //FUNCION BOTON TERMINAR
259     private void botonTerminarActionPerformed(java.awt.event.ActionEvent
            evt) {
260
261         Principal.matarHilo=true;
262         Principal.ventana_seguimiento.setVisible(false);
263         Principal.ventana_principal.setVisible(true);
```

```
264         Principal.ventana_principal.setResizable(false);
265           System.out.printf("Proceso terminado.");
266 
267     }
268 
269     //FUNCION PRINCIPAL
270     public static void main(String args[]) {
271 
272         try {
273             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
274                 if ("Nimbus".equals(info.getName())) {
275                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
276                     break;
277                 }
278             }
279         } catch (ClassNotFoundException ex) {
280             java.util.logging.Logger.getLogger(VentanaSeguimiento.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
281         } catch (InstantiationException ex) {
282             java.util.logging.Logger.getLogger(VentanaSeguimiento.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
283         } catch (IllegalAccessException ex) {
284             java.util.logging.Logger.getLogger(VentanaSeguimiento.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
285         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
286             java.util.logging.Logger.getLogger(VentanaSeguimiento.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
287         }
288 
289         java.awt.EventQueue.invokeLater(new Runnable() {
290             public void run() {
291                 new VentanaSeguimiento().setVisible(true);
292             }
293         });
```

```
294         }
295
296         // Variables declaration - do not modify
297         private javax.swing.JButton botonActualizar;
298         private javax.swing.JButton botonTerminar;
299         private javax.swing.JLabel etiquetaAnger;
300         private javax.swing.JLabel etiquetaContempt;
301         private javax.swing.JLabel etiquetaDisgust;
302         private javax.swing.JLabel etiquetaEstado;
303         private javax.swing.JLabel etiquetaFear;
304         private javax.swing.JLabel etiquetaHappiness;
305         private javax.swing.JLabel etiquetaNeutral;
306         private javax.swing.JLabel etiquetaNo;
307         private javax.swing.JLabel etiquetaSadness;
308         private javax.swing.JLabel etiquetaSurprise;
309         private javax.swing.JLabel etiquetaTitulo;
310         private javax.swing.JLabel etiquetaTotal;
311         private javax.swing.JLabel valorAnger;
312         private javax.swing.JLabel valorContempt;
313         private javax.swing.JLabel valorDisgust;
314         private javax.swing.JLabel valorEstado;
315         private javax.swing.JLabel valorFear;
316         private javax.swing.JLabel valorHappiness;
317         private javax.swing.JLabel valorNeutral;
318         private javax.swing.JLabel valorNo;
319         private javax.swing.JLabel valorSadness;
320         private javax.swing.JLabel valorSurprise;
321         private javax.swing.JLabel valorTotal;
322         // End of variables declaration
323 }
```

5.10 VentanaUsuario.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIOS
5  import PaquetePreAnalisis.Principal;
6  import java.io.BufferedReader;
7  import java.io.File;
8  import java.io.FileReader;
9  import java.io.IOException;
10 import java.nio.file.Files;
11 import java.util.logging.Level;
12 import java.util.logging.Logger;
13 import javax.swing.JFileChooser;
14 import javax.swing.filechooser.FileNameExtensionFilter;
15
16 /**
17  *
18  * @author Angel Murcia Diaz
19  */
20
21 //VENTANA: VENTANAUSUARIO
22 public class VentanaUsuario extends javax.swing.JFrame {
23
24     //CONSTRUCTOR
25     public VentanaUsuario() {
26         initComponents();
27     }
28
29     //VARIABLE ARCHIVO USUARIO
30     public static String userFile = null;
31
32     @SuppressWarnings("unchecked")
33     // <editor-fold defaultstate="collapsed" desc="Generated Code">
34     private void initComponents() {
```

```
35
36          botonCargarUsuario = new javax.swing.JButton();
37          botonNuevoUsuario = new javax.swing.JButton();
38          botonAtras = new javax.swing.JButton();
39          botonActualizarUsuario = new javax.swing.JButton();
40
41          setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE)
                ;
42          setTitle("MENU DE USUARIO");
43
44          botonCargarUsuario.setFont(new java.awt.Font("Candara", 1, 14)); //
                 NOI18N
45          botonCargarUsuario.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/cargar-usuario32x32.png"))); //
                NOI18N
46          botonCargarUsuario.setText("CARGAR USUARIO");
47          botonCargarUsuario.setToolTipText("Carga un usuario guardado
                anteriormente.");
48          botonCargarUsuario.setDefaultCapable(false);
49          botonCargarUsuario.setHorizontalTextPosition(javax.swing.
                SwingConstants.CENTER);
50          botonCargarUsuario.setVerticalTextPosition(javax.swing.
                SwingConstants.BOTTOM);
51          botonCargarUsuario.addActionListener(new java.awt.event.
                ActionListener() {
52              public void actionPerformed(java.awt.event.ActionEvent evt) {
53                  botonCargarUsuarioActionPerformed(evt);
54              }
55          });
56
57          botonNuevoUsuario.setFont(new java.awt.Font("Candara", 1, 14)); //
                NOI18N
58          botonNuevoUsuario.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/agregar-usuario32x32.png"))); //
                NOI18N
59          botonNuevoUsuario.setText("NUEVO USUARIO");
```

```
60          botonNuevoUsuario.setToolTipText("Crea un nuevo usuario, con una
                clave de Microsoft v lida.");
61          botonNuevoUsuario.setDefaultCapable(false);
62          botonNuevoUsuario.setHorizontalTextPosition(javax.swing.
                SwingConstants.CENTER);
63          botonNuevoUsuario.setVerticalTextPosition(javax.swing.
                SwingConstants.BOTTOM);
64          botonNuevoUsuario.addActionListener(new java.awt.event.
                ActionListener() {
65              public void actionPerformed(java.awt.event.ActionEvent evt) {
66                  botonNuevoUsuarioActionPerformed(evt);
67              }
68          });
69
70          botonAtras.setFont(new java.awt.Font("Candara", 1, 14)); // NOI18N
71          botonAtras.setIcon(new javax.swing.ImageIcon(getClass().getResource
                ("/PaqueteImagenes/play32.png"))); // NOI18N
72          botonAtras.setText("CONTINUAR");
73          botonAtras.setToolTipText("Una vez activado el usuario deseado haga
                 clic en este boton para continuar con el proceso.");
74          botonAtras.setDefaultCapable(false);
75          botonAtras.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
76          botonAtras.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
77          botonAtras.addActionListener(new java.awt.event.ActionListener() {
78              public void actionPerformed(java.awt.event.ActionEvent evt) {
79                  botonAtrasActionPerformed(evt);
80              }
81          });
82
83          botonActualizarUsuario.setFont(new java.awt.Font("Candara", 1, 14))
                ; // NOI18N
84          botonActualizarUsuario.setIcon(new javax.swing.ImageIcon(getClass()
                .getResource("/PaqueteImagenes/actualizar-usuario32x32.png")));
                // NOI18N
```

```
85          botonActualizarUsuario.setText("ACTUALIZAR USUARIO");
86          botonActualizarUsuario.setToolTipText("Actualiza la informaci n de
                un usuario creado con anterioridad.");
87          botonActualizarUsuario.setDefaultCapable(false);
88          botonActualizarUsuario.setHorizontalTextPosition(javax.swing.
                SwingConstants.CENTER);
89          botonActualizarUsuario.setVerticalTextPosition(javax.swing.
                SwingConstants.BOTTOM);
90          botonActualizarUsuario.addActionListener(new java.awt.event.
                ActionListener() {
91              public void actionPerformed(java.awt.event.ActionEvent evt) {
92                  botonActualizarUsuarioActionPerformed(evt);
93              }
94          });
95
96          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
97          getContentPane().setLayout(layout);
98          layout.setHorizontalGroup(
99              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
100             .addGroup(layout.createSequentialGroup()
101                 .addGap(19, 19, 19)
102                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING, false)
103                     .addComponent(botonCargarUsuario, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                            DEFAULT_SIZE, Short.MAX_VALUE)
104                     .addComponent(botonActualizarUsuario, javax.swing.
                            GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                            DEFAULT_SIZE, Short.MAX_VALUE))
105                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
106                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING, false)
107                     .addComponent(botonAtras, javax.swing.GroupLayout.
```

```
                        DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE)
108                     .addComponent(botonNuevoUsuario, javax.swing.
                        GroupLayout.DEFAULT_SIZE, 161, Short.MAX_VALUE))
109                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE))
110         );
111         layout.setVerticalGroup(
112             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
113             .addGroup(layout.createSequentialGroup()
114                 .addContainerGap()
115                 .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.LEADING, false)
116                     .addComponent(botonCargarUsuario, javax.swing.
                        GroupLayout.DEFAULT_SIZE, 80, Short.MAX_VALUE)
117                     .addComponent(botonNuevoUsuario, javax.swing.
                        GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                        DEFAULT_SIZE, Short.MAX_VALUE))
118                 .addGap(18, 18, 18)
119                 .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.LEADING, false)
120                     .addComponent(botonActualizarUsuario, javax.swing.
                        GroupLayout.DEFAULT_SIZE, 80, Short.MAX_VALUE)
121                     .addComponent(botonAtras, javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE))
122                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE))
123         );
124
125         pack();
126     }// </editor-fold>
127
128     //FUNCION CARGAR USUARIO
129     private void botonCargarUsuarioActionPerformed(java.awt.event.
```

```
        ActionEvent evt) {
130
131         //LLAMA A LA FUNCION CARGAR, para la seleccion manual del archivo
                de video
132         userFile = cargar();
133
134         //COMPROBACION, de la ruta de video elegido
135         System.out.println("La ruta del archivo .TXT contenedor del usuario
                deseado es: ");
136         System.out.println(userFile);
137
138         //PARA PROBAR CONTENIDO
139
140         //declaracion del File para la comprobaci n
141         File archivo = new File(userFile);
142
143         //String para la comprobacion
144         String tipodeArchivo = null;
145         //probar el archivo
146         try {
147             tipodeArchivo = Files.probeContentType(archivo.toPath());
148         } catch (IOException ex) {
149             Logger.getLogger(VentanaPrincipal.class.getName()).log(Level.
                    SEVERE, null, ex);
150         }
151
152
153         //System.out.println(tipodeArchivo);
154
155         //manipulacion de la cadena para comprobar facilmente si es de tipo
                video o no
156         String tipodeArchivoComparar = tipodeArchivo.substring(0, 5 );
157
158        // System.out.println("|"+tipodeArchivoComparar+"|");
159
160         //comprobacion de si es o no un archivo de video
```

```
161         if("text/".equals(tipodeArchivoComparar)){
162 
163             System.out.println("El archivo:");
164             System.out.println(userFile);
165             System.out.println("es valido.");
166 
167             File archivoLeer = null;
168 
169             FileReader fr = null;
170 
171             BufferedReader br = null;
172 
173             try{
174 
175                 archivoLeer = new File (userFile);
176 
177                 if(archivoLeer.exists()==false){
178 
179                     return;
180                 }
181 
182                 fr = new FileReader (archivoLeer);
183 
184                 br =  new BufferedReader (fr);
185 
186                 String linea;
187 
188                 linea = br.readLine();
189 
190                 Principal.user.setNick(linea);
191 
192                 linea = br.readLine();
193 
194                 Principal.user.setPass(linea);
195 
196                 linea = br.readLine();
```

```
197
198                 Principal.user.setPconex(linea);
199
200                 linea = br.readLine();
201
202                // System.out.println("|" + linea + "|");
203
204                 if("true".equals(linea)){
205
206                     Principal.user.setVersionPrueba(true);
207
208                 }else{
209
210                     Principal.user.setVersionPrueba(false);
211                 }
212
213
214                 //variabble para que funcipone el analizar
215                     Principal.sesionIniciada = true;
216
217
218             }catch (Exception e) {
219
220                         e.printStackTrace();
221
222             }finally{
223
224                 try{
225
226                     if(null != fr){
227                         fr.close();
228                     }
229
230                 } catch (Exception e2){
231
232                     e2.printStackTrace();
```

```
233
234                 }
235
236             }
237
238         }else{
239
240             System.out.println("Seleccione un archivo TXT que contenga los
                    datos de su usuario");
241
242             //ERROR DESEADO
243             Principal.error = "SELECCIONE UN ARCHIVO .TXT QUE CONTENGA LOS
                    DATOS DE SU USUARIO";
244
245             //CREACI N VENTANA DE ERROR
246             VentanaError ventana_error = new VentanaError();
247
248             //PONER VISIBLE LA VENTANA
249             ventana_error.setVisible(true);
250
251             //PONER LA VENTANA DE TAMA O FIJO
252             ventana_error.setResizable(false);
253
254         }
255
256     }
257
258     //FUNCION NUEVO USUARIO
259     private void botonNuevoUsuarioActionPerformed(java.awt.event.
            ActionEvent evt) {
260
261         Principal.ventana_usuario_nuevo.setVisible(true);
262         Principal.ventana_usuario_nuevo.setResizable(false);
263     }
264
265     //FUNCION BOTON ATRAS
```

```
266     private void botonAtrasActionPerformed(java.awt.event.ActionEvent evt)
            {
267
268         if(Principal.sesionIniciada == true){
269
270             Principal.ventana_usuarios.setVisible(false);
271
272             if(Principal.hayRutaConfirmada == true){
273
274                 Principal.ventana_principal.setVisible(true);
275                 Principal.ventana_principal.setResizable(false);
276
277             }else{
278
279                 Principal.ventana_opciones.setVisible(true);
280                 Principal.ventana_opciones.setResizable(false);
281
282             }
283
284         }else{
285
286              //ERROR DESEADO
287             Principal.error = "SE DEBE SELECCIONAR UN USUARIO ANTES DE
                    INGRESAR EN LA APLICACION";
288
289             //CREACI N VENTANA DE ERROR
290             VentanaError ventana_error = new VentanaError();
291
292             //PONER VISIBLE LA VENTANA
293             ventana_error.setVisible(true);
294
295             //PONER LA VENTANA DE TAMA O FIJO
296             ventana_error.setResizable(false);
297
298         }
299
```

```
300     }
301
302     //FUNCION ACTUALIZAR USUARIO
303     private void botonActualizarUsuarioActionPerformed(java.awt.event.
            ActionEvent evt) {
304
305         Principal.ventana_usuario_actualizar.setVisible(true);
306         Principal.ventana_usuario_actualizar.setResizable(false);
307     }
308
309
310     //FUNCION PRINCIPAL
311     public static void main(String args[]) {
312
313         try {
314             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
315                 if ("Nimbus".equals(info.getName())) {
316                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
317                     break;
318                 }
319             }
320         } catch (ClassNotFoundException ex) {
321             java.util.logging.Logger.getLogger(VentanaUsuario.class.getName
                    ()).log(java.util.logging.Level.SEVERE, null, ex);
322         } catch (InstantiationException ex) {
323             java.util.logging.Logger.getLogger(VentanaUsuario.class.getName
                    ()).log(java.util.logging.Level.SEVERE, null, ex);
324         } catch (IllegalAccessException ex) {
325             java.util.logging.Logger.getLogger(VentanaUsuario.class.getName
                    ()).log(java.util.logging.Level.SEVERE, null, ex);
326         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
327             java.util.logging.Logger.getLogger(VentanaUsuario.class.getName
                    ()).log(java.util.logging.Level.SEVERE, null, ex);
328         }
```

```
329
330             java.awt.EventQueue.invokeLater(new Runnable() {
331                 public void run() {
332                     new VentanaUsuario().setVisible(true);
333                 }
334             });
335         }
336
337         //FUNCION CARGAR
338         public String cargar(){
339
340             //fichero seleccionado
341             JFileChooser fichero = new JFileChooser();
342
343             //atajos
344             fichero.setFileFilter(new FileNameExtensionFilter("Archivos TXT", "
                    txt"));
345
346             //atajos
347             fichero.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
348
349             //ventana madre para mostrar la ventana de abrir
350             int option = fichero.showDialog(this, "Seleccionar");
351
352             //devuelve directamente el fichero especificado
353             return fichero.getSelectedFile().toString();
354
355         }
356
357         // Variables declaration - do not modify
358         private javax.swing.JButton botonActualizarUsuario;
359         private javax.swing.JButton botonAtras;
360         private javax.swing.JButton botonCargarUsuario;
361         private javax.swing.JButton botonNuevoUsuario;
362         // End of variables declaration
363 }
```

5.11 VentanaUsuarioActualizar.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIOS
5  import PaquetePreAnalisis.Principal;
6  import java.io.File;
7  import java.io.FileWriter;
8  import java.io.IOException;
9  import java.io.PrintWriter;
10 import java.io.BufferedReader;
11 import java.io.FileReader;
12 import java.nio.file.Files;
13 import java.util.logging.Level;
14 import java.util.logging.Logger;
15 import javax.swing.JFileChooser;
16 import javax.swing.filechooser.FileNameExtensionFilter;
17
18 /**
19  *
20  * @author Angel Murcia Diaz
21  */
22
23 //VENTANA: VENTANAUSUARIOACTUALIZAR
24 public class VentanaUsuarioActualizar extends javax.swing.JFrame {
25
26     //CONSTRUCTOR
27     public VentanaUsuarioActualizar() {
28         initComponents();
29     }
30
31     //VARIABLES PARA ALMACENAR LOS DATOS DEL PROCESO
32     public static String userFile = null;
33     public static String rutaFileGuardar = null;
34     public static boolean hayRutaGuardar = false;
```

```
35      public static String nick = null;
36      public static String pass = null;
37      public static String pConex = null;
38      public static boolean esPrueba = false;
39
40
41      @SuppressWarnings("unchecked")
42      // <editor-fold defaultstate="collapsed" desc="Generated Code">
43      private void initComponents() {
44
45          botonAceptar = new javax.swing.JButton();
46          botonAtras = new javax.swing.JButton();
47          etiqueta = new javax.swing.JLabel();
48          etiquetaPass = new javax.swing.JLabel();
49          etiquetaPConex = new javax.swing.JLabel();
50          valorPass = new javax.swing.JPasswordField();
51          valorPConex = new javax.swing.JTextField();
52          etiquetaRuta = new javax.swing.JLabel();
53          butonSeleccionarRuta = new javax.swing.JButton();
54          etiquetaVersionPrueba = new javax.swing.JLabel();
55          valorVersionPrueba = new javax.swing.JComboBox<>();
56
57          setDefaultCloseOperation(javax.swing.WindowConstants.
                DISPOSE_ON_CLOSE);
58          setTitle("ACTUALIZAR USUARIO");
59
60          botonAceptar.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/play32.png"))); // NOI18N
61          botonAceptar.setText("ACEPTAR");
62          botonAceptar.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
63          botonAceptar.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
64          botonAceptar.addActionListener(new java.awt.event.ActionListener()
                {
65              public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
66                botonAceptarActionPerformed(evt);
67            }
68        });
69
70        botonAtras.setIcon(new javax.swing.ImageIcon(getClass().getResource
              ("/PaqueteImagenes/atras32.png"))); // NOI18N
71        botonAtras.setText("ATRAS");
72        botonAtras.setHorizontalTextPosition(javax.swing.SwingConstants.
              CENTER);
73        botonAtras.setVerticalTextPosition(javax.swing.SwingConstants.
              BOTTOM);
74        botonAtras.addActionListener(new java.awt.event.ActionListener() {
75            public void actionPerformed(java.awt.event.ActionEvent evt) {
76                botonAtrasActionPerformed(evt);
77            }
78        });
79
80        etiqueta.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
81        etiqueta.setText("ACTUALIZAR USUARIO:");
82
83        etiquetaPass.setText("Contrase a de ApiFace:");
84
85        etiquetaPConex.setText("Punto de conexion:");
86
87        valorPConex.setText("https://westcentralus.api.cognitive.microsoft.
              com/face/v1.0/detect");
88
89        etiquetaRuta.setText("Usuario a actualizar:");
90
91        butonSeleccionarRuta.setText("Seleccionar");
92        butonSeleccionarRuta.addActionListener(new java.awt.event.
              ActionListener() {
93            public void actionPerformed(java.awt.event.ActionEvent evt) {
94                butonSeleccionarRutaActionPerformed(evt);
95            }
96        });
```

```
97
98          etiquetaVersionPrueba.setText(" Version  de Prueba?:");
99
100         valorVersionPrueba.setModel(new javax.swing.DefaultComboBoxModel<>(
                new String[] { "Si", "No" }));
101         valorVersionPrueba.setCursor(new java.awt.Cursor(java.awt.Cursor.
                DEFAULT_CURSOR));
102
103         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
104         getContentPane().setLayout(layout);
105         layout.setHorizontalGroup(
106             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
107             .addGroup(layout.createSequentialGroup()
108                 .addGap(27, 27, 27)
109                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
110                     .addGroup(layout.createSequentialGroup()
111                         .addComponent(etiquetaPConex, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 159, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
112                         .addContainerGap(javax.swing.GroupLayout.
                                DEFAULT_SIZE, Short.MAX_VALUE))
113                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                            layout.createSequentialGroup()
114                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.TRAILING)
115                             .addGroup(layout.createSequentialGroup()
116                                 .addComponent(botonAceptar, javax.swing.
                                        GroupLayout.PREFERRED_SIZE, 168, javax.
                                        swing.GroupLayout.PREFERRED_SIZE)
117                                 .addPreferredGap(javax.swing.LayoutStyle.
                                        ComponentPlacement.RELATED, javax.swing.
                                        GroupLayout.DEFAULT_SIZE, Short.
                                        MAX_VALUE)
```

```
118                         .addComponent(botonAtras, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 125, javax.
                                swing.GroupLayout.PREFERRED_SIZE))
119                     .addGroup(javax.swing.GroupLayout.Alignment.
                            LEADING, layout.createSequentialGroup()
120                         .addGroup(layout.createParallelGroup(javax.
                                swing.GroupLayout.Alignment.LEADING)
121                             .addComponent(etiquetaPass, javax.swing
                                    .GroupLayout.PREFERRED_SIZE, 159,
                                    javax.swing.GroupLayout.
                                    PREFERRED_SIZE)
122                             .addComponent(etiquetaVersionPrueba,
                                    javax.swing.GroupLayout.
                                    PREFERRED_SIZE, 159, javax.swing.
                                    GroupLayout.PREFERRED_SIZE))
123                         .addGap(45, 45, 45)
124                         .addGroup(layout.createParallelGroup(javax.
                                swing.GroupLayout.Alignment.LEADING)
125                             .addComponent(valorPConex, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 0, Short
                                    .MAX_VALUE)
126                             .addComponent(valorPass)
127                             .addComponent(butonSeleccionarRuta,
                                    javax.swing.GroupLayout.DEFAULT_SIZE
                                    , javax.swing.GroupLayout.
                                    DEFAULT_SIZE, Short.MAX_VALUE)
128                             .addComponent(valorVersionPrueba, 0,
                                    javax.swing.GroupLayout.DEFAULT_SIZE
                                    , Short.MAX_VALUE))))
129                 .addGap(26, 26, 26))
130             .addGroup(layout.createSequentialGroup()
131                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
132                     .addComponent(etiquetaRuta, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 168, javax.swing
                            .GroupLayout.PREFERRED_SIZE)
```

```
133                             .addComponent(etiqueta, javax.swing.GroupLayout
                                    .PREFERRED_SIZE, 234, javax.swing.
                                    GroupLayout.PREFERRED_SIZE))
134                         .addGap(0, 0, Short.MAX_VALUE))))
135         );
136         layout.setVerticalGroup(
137             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                LEADING)
138             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
                createSequentialGroup()
139                 .addContainerGap()
140                 .addComponent(etiqueta, javax.swing.GroupLayout.
                    PREFERRED_SIZE, 37, javax.swing.GroupLayout.
                    PREFERRED_SIZE)
141                 .addGap(5, 5, 5)
142                 .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.BASELINE)
143                     .addComponent(etiquetaRuta, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 48, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
144                     .addComponent(butonSeleccionarRuta, javax.swing.
                        GroupLayout.PREFERRED_SIZE, 48, javax.swing.
                        GroupLayout.PREFERRED_SIZE))
145                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                    .UNRELATED)
146                 .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.BASELINE)
147                     .addComponent(etiquetaPass, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 47, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
148                     .addComponent(valorPass, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 47, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
149                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                    .UNRELATED)
150                 .addGroup(layout.createParallelGroup(javax.swing.
```

```
                    GroupLayout.Alignment.LEADING, false)
151                  .addComponent(etiquetaPConex, javax.swing.GroupLayout.
                        DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                        Short.MAX_VALUE)
152                  .addComponent(valorPConex, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 47, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
153              .addGap(18, 18, 18)
154              .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.LEADING)
155                  .addComponent(etiquetaVersionPrueba, javax.swing.
                        GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                        DEFAULT_SIZE, Short.MAX_VALUE)
156                  .addGroup(layout.createSequentialGroup()
157                      .addComponent(valorVersionPrueba, javax.swing.
                            GroupLayout.PREFERRED_SIZE, 49, javax.swing.
                            GroupLayout.PREFERRED_SIZE)
158                      .addGap(0, 0, Short.MAX_VALUE)))
159              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                    .RELATED, 26, Short.MAX_VALUE)
160              .addGroup(layout.createParallelGroup(javax.swing.
                    GroupLayout.Alignment.LEADING)
161                  .addComponent(botonAceptar, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 86, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
162                  .addComponent(botonAtras, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 86, javax.swing.GroupLayout.
                        PREFERRED_SIZE))
163              .addContainerGap())
164      );
165
166      pack();
167  }// </editor-fold>
168
169
170  //FUNCION BOTON ACEPTAR
```

```
171     private void botonAceptarActionPerformed(java.awt.event.ActionEvent evt
            ) {
172
173         if(hayRutaGuardar == true){
174
175             String nickUsuario = nick;
176             String passUsuario = new String(valorPass.getPassword());
177             String pConexUsuario = valorPConex.getText();
178             System.out.println("--------");
179             System.out.println("|"+ passUsuario + "|");
180             System.out.println("|"+ pConexUsuario + "|");
181             System.out.println("--------");
182
183             boolean esPruebaUsuario = false;
184
185             if( valorVersionPrueba.getSelectedIndex() == 0){
186
187                 esPruebaUsuario = true;
188
189             }else{
190
191                 esPruebaUsuario = false;
192             }
193
194             if(!"".equals(passUsuario) && !"".equals(pConexUsuario)){
195
196                 Principal.user.setNick(nickUsuario);
197
198                 Principal.user.setPass(passUsuario);
199
200                 Principal.user.setPconex(pConexUsuario);
201
202                 Principal.user.setVersionPrueba(esPruebaUsuario);
203
204                 //volcar usuario a fichero en esa ruta
205                 FileWriter fichero = null;
```

```
206
207                 PrintWriter pw = null;
208
209                 String nombreFichero = userFile ;
210
211                 System.out.println(nombreFichero);
212
213                 try{
214
215                    fichero = new FileWriter(nombreFichero);
216                    pw = new PrintWriter(fichero);
217                    System.out.println("Escribiendo fichero de usuario en
                          formato .TXT");
218
219                    pw.println(Principal.user.getNick());
220                    pw.println(Principal.user.getPass());
221                    pw.println(Principal.user.getPconex());
222                    pw.println(Principal.user.getVersionPrueba());
223
224                 }catch (Exception e) {
225
226                     e.printStackTrace();
227
228                 } finally{
229
230                     try{
231
232                         if(null != fichero){
233                             fichero.close();
234                         }
235
236                     } catch (Exception e2){
237
238                         e2.printStackTrace();
239                     }
240
```

```
241                 }
242
243                 //ocultar la ventana
244                 Principal.ventana_usuario_actualizar.setVisible(false);
245
246                 //variabble para que funcipone el analizar
247                 Principal.sesionIniciada = true;
248
249             }else{
250
251                 //ERROR NO PUEDEN QUEDARSE NINGUNO DE LOS 3 PRIMEROS CAMPOS
                        VACIOS
252                 //ERROR DESEADO
253                 Principal.error = "NO PUEDEN QUEDARSE NINGUNO DE LOS CAMPOS
                        VACIOS Y SE HA DE SELECIONAR UN USUARIO EN EL FORMATO.
                       TXT";
254
255                 //CREACI N VENTANA DE ERROR
256                 VentanaError ventana_error = new VentanaError();
257
258                 //PONER VISIBLE LA VENTANA
259                 ventana_error.setVisible(true);
260
261                 //PONER LA VENTANA DE TAMA O FIJO
262                 ventana_error.setResizable(false);
263
264             }
265
266         }else{
267
268             //ERROR NO HAY RUTA PARA ALMACENAR EL USUARIO EN FORMATO TXT
269             //ERROR DESEADO
270             Principal.error = "NO SE HA SELECCIONADO NINGUN USUARIO EN
                    FORMATO TXT";
271
272             //CREACI N VENTANA DE ERROR
```

```
273             VentanaError ventana_error = new VentanaError();
274
275             //PONER VISIBLE LA VENTANA
276             ventana_error.setVisible(true);
277
278             //PONER LA VENTANA DE TAMA O FIJO
279             ventana_error.setResizable(false);
280
281         }
282
283     }
284
285     //FUNCION BOTON ATRAS
286     private void botonAtrasActionPerformed(java.awt.event.ActionEvent evt)
            {
287
288         Principal.ventana_usuario_actualizar.setVisible(false);
289         Principal.ventana_usuarios.setVisible(true);
290
291     }
292
293     //FUNCION BOTON SELECCIONAR TRUTA
294     private void butonSeleccionarRutaActionPerformed(java.awt.event.
            ActionEvent evt) {
295
296         //LLAMA A LA FUNCION CARGAR, para la seleccion manual del archivo
                de video
297         userFile = cargar();
298
299         //COMPROBACION, de la ruta de video elegido
300         System.out.println("La ruta del archivo .TXT contenedor del usuario
                deseado es: ");
301         System.out.println(userFile);
302
303         //PARA PROBAR CONTENIDO
304
```

```
305         //declaracion del File para la comprobaci n
306         File archivo = new File(userFile);
307 
308         //String para la comprobacion
309         String tipodeArchivo = null;
310         //probar el archivo
311         try {
312             tipodeArchivo = Files.probeContentType(archivo.toPath());
313         } catch (IOException ex) {
314             Logger.getLogger(VentanaPrincipal.class.getName()).log(Level.
                SEVERE, null, ex);
315         }
316 
317         //manipulacion de la cadena para comprobar facilmente si es de tipo
              video o no
318         String tipodeArchivoComparar = tipodeArchivo.substring(0, 5 );
319 
320         System.out.println("|"+tipodeArchivoComparar+"|");
321 
322         //comprobacion de si es o no un archivo de video
323         if("text/".equals(tipodeArchivoComparar)){
324 
325             System.out.println("texto");
326             System.out.println(userFile);
327 
328             File archivoLeer = null;
329 
330             FileReader fr = null;
331 
332             BufferedReader br = null;
333 
334             try{
335 
336                 archivoLeer = new File (userFile);
337 
338                 if(archivoLeer.exists()==false){
```

```
339
340                     return;
341                 }
342
343                 fr = new FileReader (archivoLeer);
344                 br =  new BufferedReader (fr);
345                 String linea;
346                 linea = br.readLine();
347                 nick = linea;
348                 hayRutaGuardar = true;
349
350             }catch (Exception e) {
351
352                     e.printStackTrace();
353
354             } finally{
355
356                 try{
357
358                     if(null != fr){
359                         fr.close();
360                     }
361
362                 } catch (Exception e2){
363
364                     e2.printStackTrace();
365
366                 }
367
368             }
369
370         }else{
371
372             System.out.println("Seleccione un archivo TXT que contenga los
                    datos de su usuario");
373
```

```
374             //ERROR DESEADO
375             Principal.error = "SELECCIONE UN ARCHIVO .TXT QUE CONTENGA LOS
                    DATOS DE SU USUARIO";
376
377             //CREACI N VENTANA DE ERROR
378             VentanaError ventana_error = new VentanaError();
379
380             //PONER VISIBLE LA VENTANA
381             ventana_error.setVisible(true);
382
383             //PONER LA VENTANA DE TAMA O FIJO
384             ventana_error.setResizable(false);
385
386         }
387
388     }
389
390     //FUNCION PRINCIPAL
391     public static void main(String args[]) {
392
393         try {
394             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
395                 if ("Nimbus".equals(info.getName())) {
396                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
397                     break;
398                 }
399             }
400         } catch (ClassNotFoundException ex) {
401             java.util.logging.Logger.getLogger(VentanaUsuarioNuevo.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
402         } catch (InstantiationException ex) {
403             java.util.logging.Logger.getLogger(VentanaUsuarioNuevo.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
404         } catch (IllegalAccessException ex) {
```

```
405             java.util.logging.Logger.getLogger(VentanaUsuarioNuevo.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
406         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
407             java.util.logging.Logger.getLogger(VentanaUsuarioNuevo.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
408         }
409 
410         java.awt.EventQueue.invokeLater(new Runnable() {
411             public void run() {
412                 new VentanaUsuarioNuevo().setVisible(true);
413             }
414         });
415     }
416 
417     //FUNCION CARGARRUTA
418     public String cargarRuta(){
419 
420         //fichero seleccionado
421         JFileChooser fichero = new JFileChooser();
422 
423         //atajos
424         fichero.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
425 
426         //ventana madre para mostrar la ventana de abrir
427         int option = fichero.showDialog(this, "Aceptar");
428 
429         //devuelve directamente el fichero especificado
430         return fichero.getSelectedFile().toString();
431 
432     }
433 
434     //FUNCION CARGAR
435     public String cargar(){
436 
437         //fichero seleccionado
438         JFileChooser fichero = new JFileChooser();
```

```
439
440             //atajos
441             //fcPicture.setFileFilter(new FileNameExtensionFilter("Archivo de
                    imagen", "jpg", "JPG", "jpeg", "JPEG", "png", "PNG", "gif", "GIF
                    ", "tif", "TIF", "tiff", "TIFF"));
442             fichero.setFileFilter(new FileNameExtensionFilter("Archivos TXT", "
                    txt"));
443
444             //atajos
445             fichero.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
446
447             //ventana madre para mostrar la ventana de abrir
448             int option = fichero.showDialog(this, "Seleccionar");
449
450             //devuelve directamente el fichero especificado
451             return fichero.getSelectedFile().toString();
452
453         }
454
455
456         // Variables declaration - do not modify
457         private javax.swing.JButton botonAceptar;
458         private javax.swing.JButton botonAtras;
459         private javax.swing.JButton butonSeleccionarRuta;
460         private javax.swing.JLabel etiqueta;
461         private javax.swing.JLabel etiquetaPConex;
462         private javax.swing.JLabel etiquetaPass;
463         private javax.swing.JLabel etiquetaRuta;
464         private javax.swing.JLabel etiquetaVersionPrueba;
465         private javax.swing.JTextField valorPConex;
466         private javax.swing.JPasswordField valorPass;
467         private javax.swing.JComboBox<String> valorVersionPrueba;
468         // End of variables declaration
469     }
```

5.12 VentanaUsuarioNuevo.java

```
1  //PAQUETE NECESARIO
2  package PaqueteVentanas;
3
4  //IMPORT NECESARIOS
5  import PaquetePreAnalisis.Principal;
6  import java.io.File;
7  import java.io.FileWriter;
8  import java.io.IOException;
9  import java.io.PrintWriter;
10 import java.io.BufferedReader;
11 import java.nio.file.Files;
12 import java.util.logging.Level;
13 import java.util.logging.Logger;
14 import javax.swing.JFileChooser;
15 import javax.swing.filechooser.FileNameExtensionFilter;
16
17 /**
18  *
19  * @author Angel Murcia Diaz
20  */
21
22 //VENTANA: VENTANAUSUARIONUEVO
23 public class VentanaUsuarioNuevo extends javax.swing.JFrame {
24
25     //CONSTRUCTOR
26     public VentanaUsuarioNuevo() {
27         initComponents();
28     }
29
30     //VARIABLES PARA ALMACENAR LOS DATOS INTRODUCIDOS
31     public static String rutaFileGuardar = null;
32     public static boolean hayRutaGuardar = false;
33     public static String nick = null;
34     public static String pass = null;
```

```
35      public static String pConex = null;
36      public static boolean esPrueba = false;
37
38      @SuppressWarnings("unchecked")
39      // <editor-fold defaultstate="collapsed" desc="Generated Code">
40      private void initComponents() {
41
42          botonAceptar = new javax.swing.JButton();
43          botonAtras = new javax.swing.JButton();
44          etiqueta = new javax.swing.JLabel();
45          etiquetaPass = new javax.swing.JLabel();
46          etiquetaPConex = new javax.swing.JLabel();
47          etiquetaNick = new javax.swing.JLabel();
48          valorPass = new javax.swing.JPasswordField();
49          valorNick = new javax.swing.JTextField();
50          valorPConex = new javax.swing.JTextField();
51          etiquetaRuta = new javax.swing.JLabel();
52          butonSeleccionarRuta = new javax.swing.JButton();
53          etiquetaVersionPrueba = new javax.swing.JLabel();
54          valorVersionPrueba = new javax.swing.JComboBox<>();
55
56          setDefaultCloseOperation(javax.swing.WindowConstants.
                DISPOSE_ON_CLOSE);
57
58          botonAceptar.setIcon(new javax.swing.ImageIcon(getClass().
                getResource("/PaqueteImagenes/play32.png"))); // NOI18N
59          botonAceptar.setText("ACEPTAR");
60          botonAceptar.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
61          botonAceptar.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
62          botonAceptar.addActionListener(new java.awt.event.ActionListener()
                {
63              public void actionPerformed(java.awt.event.ActionEvent evt) {
64                  botonAceptarActionPerformed(evt);
65              }
```

```
66          });
67
68          botonAtras.setIcon(new javax.swing.ImageIcon(getClass().getResource
                ("/PaqueteImagenes/atras32.png"))); // NOI18N
69          botonAtras.setText("ATRAS");
70          botonAtras.setHorizontalTextPosition(javax.swing.SwingConstants.
                CENTER);
71          botonAtras.setVerticalTextPosition(javax.swing.SwingConstants.
                BOTTOM);
72          botonAtras.addActionListener(new java.awt.event.ActionListener() {
73              public void actionPerformed(java.awt.event.ActionEvent evt) {
74                  botonAtrasActionPerformed(evt);
75              }
76          });
77
78          etiqueta.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
79          etiqueta.setText("NUEVO USUARIO:");
80
81          etiquetaPass.setText("Contrase a de ApiFace:");
82
83          etiquetaPConex.setText("Punto de conexion:");
84
85          etiquetaNick.setText("Nombre o nick:");
86
87          valorPass.setText("ae0b18f78ec44aedbf41fe475acf1949");
88
89          valorPConex.setText("https://westcentralus.api.cognitive.microsoft.
                com/face/v1.0/detect");
90
91          etiquetaRuta.setText("Ruta para guardarlo:");
92
93          butonSeleccionarRuta.setText("Seleccionar");
94          butonSeleccionarRuta.addActionListener(new java.awt.event.
                ActionListener() {
95              public void actionPerformed(java.awt.event.ActionEvent evt) {
96                  butonSeleccionarRutaActionPerformed(evt);
```

```
97              }
98          });
99
100         etiquetaVersionPrueba.setText(" Version  de Prueba?:");
101
102         valorVersionPrueba.setModel(new javax.swing.DefaultComboBoxModel<>(
                new String[] { "Si", "No" }));
103         valorVersionPrueba.setCursor(new java.awt.Cursor(java.awt.Cursor.
                DEFAULT_CURSOR));
104
105         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
                getContentPane());
106         getContentPane().setLayout(layout);
107         layout.setHorizontalGroup(
108             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
109             .addGroup(layout.createSequentialGroup()
110                 .addGap(27, 27, 27)
111                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING)
112                     .addGroup(layout.createSequentialGroup()
113                         .addComponent(etiquetaPConex, javax.swing.
                                GroupLayout.PREFERRED_SIZE, 159, javax.swing.
                                GroupLayout.PREFERRED_SIZE)
114                         .addContainerGap(javax.swing.GroupLayout.
                                DEFAULT_SIZE, Short.MAX_VALUE))
115                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                            layout.createSequentialGroup()
116                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.TRAILING)
117                             .addGroup(layout.createSequentialGroup()
118                                 .addComponent(botonAceptar, javax.swing.
                                        GroupLayout.PREFERRED_SIZE, 168, javax.
                                        swing.GroupLayout.PREFERRED_SIZE)
119                                 .addPreferredGap(javax.swing.LayoutStyle.
                                        ComponentPlacement.RELATED, javax.swing.
```

```
                GroupLayout.DEFAULT_SIZE, Short.
                MAX_VALUE)
120             .addComponent(botonAtras, javax.swing.
                GroupLayout.PREFERRED_SIZE, 125, javax.
                swing.GroupLayout.PREFERRED_SIZE))
121         .addGroup(javax.swing.GroupLayout.Alignment.
            LEADING, layout.createSequentialGroup()
122             .addGap(204, 204, 204)
123             .addComponent(valorNick))
124         .addGroup(javax.swing.GroupLayout.Alignment.
            LEADING, layout.createSequentialGroup()
125             .addGroup(layout.createParallelGroup(javax.
                swing.GroupLayout.Alignment.LEADING)
126                 .addComponent(etiquetaPass, javax.swing
                    .GroupLayout.PREFERRED_SIZE, 159,
                    javax.swing.GroupLayout.
                    PREFERRED_SIZE)
127                 .addComponent(etiquetaRuta, javax.swing
                    .GroupLayout.PREFERRED_SIZE, 168,
                    javax.swing.GroupLayout.
                    PREFERRED_SIZE)
128                 .addComponent(etiquetaVersionPrueba,
                    javax.swing.GroupLayout.
                    PREFERRED_SIZE, 159, javax.swing.
                    GroupLayout.PREFERRED_SIZE))
129             .addGap(36, 36, 36)
130             .addGroup(layout.createParallelGroup(javax.
                swing.GroupLayout.Alignment.LEADING)
131                 .addComponent(valorPConex, javax.swing.
                    GroupLayout.PREFERRED_SIZE, 0, Short
                    .MAX_VALUE)
132                 .addComponent(valorPass)
133                 .addComponent(butonSeleccionarRuta,
                    javax.swing.GroupLayout.DEFAULT_SIZE
                    , javax.swing.GroupLayout.
                    DEFAULT_SIZE, Short.MAX_VALUE)
```

```
134                                     .addComponent(valorVersionPrueba, 0,
                                            javax.swing.GroupLayout.DEFAULT_SIZE
                                            , Short.MAX_VALUE))))
135                         .addGap(26, 26, 26))
136                     .addGroup(layout.createSequentialGroup()
137                         .addGroup(layout.createParallelGroup(javax.swing.
                                GroupLayout.Alignment.LEADING)
138                             .addComponent(etiquetaNick, javax.swing.
                                    GroupLayout.PREFERRED_SIZE, 159, javax.swing
                                    .GroupLayout.PREFERRED_SIZE)
139                             .addComponent(etiqueta, javax.swing.GroupLayout
                                    .PREFERRED_SIZE, 234, javax.swing.
                                    GroupLayout.PREFERRED_SIZE))
140                         .addGap(0, 0, Short.MAX_VALUE))))
141         );
142         layout.setVerticalGroup(
143             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
                    LEADING)
144             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
                    createSequentialGroup()
145                 .addContainerGap()
146                 .addComponent(etiqueta, javax.swing.GroupLayout.
                        PREFERRED_SIZE, 37, javax.swing.GroupLayout.
                        PREFERRED_SIZE)
147                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .RELATED)
148                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.LEADING, false)
149                     .addComponent(etiquetaNick, javax.swing.GroupLayout.
                            DEFAULT_SIZE, 47, Short.MAX_VALUE)
150                     .addComponent(valorNick))
151                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                        .UNRELATED)
152                 .addGroup(layout.createParallelGroup(javax.swing.
                        GroupLayout.Alignment.BASELINE)
153                     .addComponent(etiquetaPass, javax.swing.GroupLayout.
```

```
                PREFERRED_SIZE, 47, javax.swing.GroupLayout.
                PREFERRED_SIZE)
154             .addComponent(valorPass, javax.swing.GroupLayout.
                PREFERRED_SIZE, 47, javax.swing.GroupLayout.
                PREFERRED_SIZE))
155         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
            .UNRELATED)
156         .addGroup(layout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.LEADING, false)
157             .addComponent(etiquetaPConex, javax.swing.GroupLayout.
                DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                Short.MAX_VALUE)
158             .addComponent(valorPConex, javax.swing.GroupLayout.
                PREFERRED_SIZE, 47, javax.swing.GroupLayout.
                PREFERRED_SIZE))
159         .addGap(18, 18, 18)
160         .addGroup(layout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.LEADING)
161             .addComponent(etiquetaVersionPrueba, javax.swing.
                GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.
                DEFAULT_SIZE, Short.MAX_VALUE)
162             .addComponent(valorVersionPrueba, javax.swing.
                GroupLayout.DEFAULT_SIZE, 46, Short.MAX_VALUE))
163         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
            .UNRELATED)
164         .addGroup(layout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.BASELINE)
165             .addComponent(etiquetaRuta, javax.swing.GroupLayout.
                PREFERRED_SIZE, 48, javax.swing.GroupLayout.
                PREFERRED_SIZE)
166             .addComponent(butonSeleccionarRuta, javax.swing.
                GroupLayout.PREFERRED_SIZE, 48, javax.swing.
                GroupLayout.PREFERRED_SIZE))
167         .addGap(42, 42, 42)
168         .addGroup(layout.createParallelGroup(javax.swing.
            GroupLayout.Alignment.LEADING)
```

```
169                     .addComponent(botonAtras, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 86, javax.swing.GroupLayout.
                            PREFERRED_SIZE)
170                     .addComponent(botonAceptar, javax.swing.GroupLayout.
                            PREFERRED_SIZE, 86, javax.swing.GroupLayout.
                            PREFERRED_SIZE))
171                 .addContainerGap())
172         );
173 
174         pack();
175     }// </editor-fold>
176 
177     //FUNCION BOTON ACEPTAR
178     private void botonAceptarActionPerformed(java.awt.event.ActionEvent evt
            ) {
179 
180         if(hayRutaGuardar == true){
181 
182             String nickUsuario = valorNick.getText();
183             String passUsuario = new String(valorPass.getPassword());
184             String pConexUsuario = valorPConex.getText();
185 
186             boolean esPruebaUsuario = false;
187 
188             if( valorVersionPrueba.getSelectedIndex() == 0){
189 
190                 esPruebaUsuario = true;
191 
192             }else{
193 
194                 esPruebaUsuario = false;
195             }
196 
197             if(!"".equals(nickUsuario) && !"".equals(passUsuario) && !"".
                    equals(pConexUsuario)){
198 
```

```
199                 Principal.user.setNick(nickUsuario);
200
201                 Principal.user.setPass(passUsuario);
202
203                 Principal.user.setPconex(pConexUsuario);
204
205                 Principal.user.setVersionPrueba(esPruebaUsuario);
206
207                 //volcar usuario a fichero en esa ruta
208                 FileWriter fichero = null;
209                 PrintWriter pw = null;
210                 String nombreFichero = rutaFileGuardar + "\\" + Principal.
                        user.getNick() + ".txt" ;
211                 System.out.println(nombreFichero);
212
213                 try{
214
215                    fichero = new FileWriter(nombreFichero);
216                    pw = new PrintWriter(fichero);
217
218                    System.out.println("Escribiendo fichero de usuario en
                           formato .TXT");
219                    pw.println(Principal.user.getNick());
220                    pw.println(Principal.user.getPass());
221                    pw.println(Principal.user.getPconex());
222                    pw.println(Principal.user.getVersionPrueba());
223
224                 }catch (Exception e) {
225
226                     e.printStackTrace();
227
228                 } finally{
229
230                     try{
231
232                         if(null != fichero){
```

```
233                         fichero.close();
234                     }
235
236                 } catch (Exception e2){
237
238                     e2.printStackTrace();
239
240                 }
241
242             }
243
244
245             //ocultar la ventana
246             Principal.ventana_usuario_nuevo.setVisible(false);
247
248             //variabble para que funcipone el analizar
249             Principal.sesionIniciada = true;
250
251         }else{
252
253             //ERROR NO PUEDEN QUEDARSE NINGUNO DE LOS 3 PRIMEROS CAMPOS
                    VACIOS
254             //ERROR DESEADO
255             Principal.error = "NO PUEDEN QUEDARSE NINGUNO DE LOS 3
                   PRIMEROS CAMPOS VACIOS";
256
257             //CREACI N VENTANA DE ERROR
258             VentanaError ventana_error = new VentanaError();
259
260             //PONER VISIBLE LA VENTANA
261             ventana_error.setVisible(true);
262
263             //PONER LA VENTANA DE TAMA O FIJO
264             ventana_error.setResizable(false);
265
266         }
```

```
267
268         }else{
269
270             //ERROR NO HAY RUTA PARA ALMACENAR EL USUARIO EN FORMATO TXT
271             //ERROR DESEADO
272             Principal.error = "NO HAY RUTA PARA ALMACENAR EL USUARIO EN
                    FORMATO TXT";
273
274             //CREACI N VENTANA DE ERROR
275             VentanaError ventana_error = new VentanaError();
276
277             //PONER VISIBLE LA VENTANA
278             ventana_error.setVisible(true);
279
280             //PONER LA VENTANA DE TAMA O FIJO
281             ventana_error.setResizable(false);
282
283         }
284
285     }
286
287     //FUNCION BOTON ATRAS
288     private void botonAtrasActionPerformed(java.awt.event.ActionEvent evt)
            {
289
290         Principal.ventana_usuario_nuevo.setVisible(false);
291         Principal.ventana_usuarios.setVisible(true);
292
293     }
294
295     //FUNCION BOTON SELECCIONAR RUTA
296     private void butonSeleccionarRutaActionPerformed(java.awt.event.
            ActionEvent evt) {
297
298         //resetear valores iniciales de los contadores
299         rutaFileGuardar = cargarRuta();
```

```
300         System.out.println("|"+rutaFileGuardar+"|");
301 
302         //declaracion del File para la comprobaci n
303         File archivo = new File(rutaFileGuardar);
304 
305         if (archivo.exists() && archivo.isDirectory()) {
306 
307             hayRutaGuardar=true;
308 
309         }else{
310 
311             //ERROR DESEADO
312             Principal.error = "NO SE HA SELECCIONADO UNA RUTA VALIDA";
313 
314             //CREACI N VENTANA DE ERROR
315             VentanaError ventana_error = new VentanaError();
316 
317             //PONER VISIBLE LA VENTANA
318             ventana_error.setVisible(true);
319 
320             //PONER LA VENTANA DE TAMA O FIJO
321             ventana_error.setResizable(false);
322          }
323     }
324 
325     //FUNCION PRINCIPAL
326     public static void main(String args[]) {
327 
328         try {
329             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.
                    UIManager.getInstalledLookAndFeels()) {
330                 if ("Nimbus".equals(info.getName())) {
331                     javax.swing.UIManager.setLookAndFeel(info.getClassName
                            ());
332                     break;
333                 }
```

```
334             }
335         } catch (ClassNotFoundException ex) {
336             java.util.logging.Logger.getLogger(VentanaUsuarioNuevo.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
337         } catch (InstantiationException ex) {
338             java.util.logging.Logger.getLogger(VentanaUsuarioNuevo.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
339         } catch (IllegalAccessException ex) {
340             java.util.logging.Logger.getLogger(VentanaUsuarioNuevo.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
341         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
342             java.util.logging.Logger.getLogger(VentanaUsuarioNuevo.class.
                    getName()).log(java.util.logging.Level.SEVERE, null, ex);
343         }
344 
345         java.awt.EventQueue.invokeLater(new Runnable() {
346             public void run() {
347                 new VentanaUsuarioNuevo().setVisible(true);
348             }
349         });
350      }
351 
352     //FUNCION CARGARRUTA
353     public String cargarRuta(){
354 
355         //fichero seleccionado
356         JFileChooser fichero = new JFileChooser();
357 
358         //atajos
359         fichero.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
360 
361         //ventana madre para mostrar la ventana de abrir
362         int option = fichero.showDialog(this, "Aceptar");
363 
364         //devuelve directamente el fichero especificado
365         return fichero.getSelectedFile().toString();
```

```
366
367         }
368
369
370         // Variables declaration - do not modify
371         private javax.swing.JButton botonAceptar;
372         private javax.swing.JButton botonAtras;
373         private javax.swing.JButton butonSeleccionarRuta;
374         private javax.swing.JLabel etiqueta;
375         private javax.swing.JLabel etiquetaNick;
376         private javax.swing.JLabel etiquetaPConex;
377         private javax.swing.JLabel etiquetaPass;
378         private javax.swing.JLabel etiquetaRuta;
379         private javax.swing.JLabel etiquetaVersionPrueba;
380         private javax.swing.JTextField valorNick;
381         private javax.swing.JTextField valorPConex;
382         private javax.swing.JPasswordField valorPass;
383         private javax.swing.JComboBox<String> valorVersionPrueba;
384         // End of variables declaration
385 }
```