UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE CURSOS EN COOPERACIÓN TÉCNICA Y DESARROLLO SOCIAL

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN PARA LA DETECCIÓN DE TEMBLORES DEL MAL DE PARKINSON

Por:
Alejandro Marzinotto Cos

Tutor Institucional:
Mónica Huerta

Representante de la Comunidad:
Mónica Huerta

INFORME DE SERVICIO COMUNITARIO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero Electrónico

Sartenejas, 14 de octubre de 2012

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

La ley de servicio comunitario establece que los estudiantes de educación superior deben realizar 120 horas de trabajo comunitario como requisito parcial para optar por su título. Idealmente dicho trabajo debe involucrar el uso de herramientas y destrezas adquiridas durante los años de estudio en la universidad. Dicho servicio comunitario tiene como objetivo beneficiar de algún modo a la comunidad.

La realización de este trabajo tiene como objetivo fundamental la implementación de una aplicacion para teléfonos móviles para la detección de los temblores producidos por la enfermedad de Parkinson. Para lograr este objetivo se utilizó la plataforma Android, la cual presenta un uso ampliamente difundido entre los usuarios de teléfonos móviles, permitiendo así que nuestro trabajo tenga un alcance mayor.

El trabajo involucra tanto la parte de investigación de los esfuerzos que se han hecho a lo largo del tiempo para detectar estos temblores, como la parte de implementación y desarrollo de la aplicación para lograr obtener un producto terminado que los pacientes puedan utilizar sin problemas.

En el presente informe explicaremos la importancia de dicho proyecto. Se presentarán las actividades realizadas durante el servicio comunitario, así como tambien la comunidad beneficiada y la relación entre dichas actividades y la formación académica del estudiante. Por último ofreceremos las conclusiones y recomendaciones para aquellos que deseen aprovechar este trabajo para otros proyectos.

CAPÍTULO 2

JUSTIFICACIÓN DEL SERVICIO COMUNITARIO

A pesar de que hoy en día muchas personas poseen teléfonos inteligentes, capaces de ejecutar una infinidad de aplicaciones tales como juegos, diccionarios, despertadores, etc., existe un déficit de aplicaciones en el área de la salud, tanto para la plataforma Android como para la plataforma iOS.

Para ayudar a solventar esta situación proponemos el presente proyecto en el cual se pretende acortar la brecha que existe entre las aplicaciones de índole comercial, y las aplicaciones en el campo de la salud, las cuales no tienen como propósito entretener, sino mejorar la calidad de vida de las personas que sufren de alguna enfermedad.

Para lograr este objetivo se utilizará un gran número de las capacidades de los telefonos móviles actuales, tales como: acelerómetro, conexión a internet, almacenaje de datos, reproducción de videos, entre otros.

Las personas beneficiadas con este proyecto serán aquellos que padezcan de la enfermedad de Parkinson, quienes podrán descargar la aplicación para su teléfono móvil y así poder medir la intensidad y duración de los temblores, mandar reportes por email, o guardarlos en la memoria interna del dispositivo para uso posterior.

CAPÍTULO 3

DESCRIPCIÓN DEL PROBLEMA

3.1 Descripción de la Comunidad

La comunidad que se beneficiará de este proyecto de servicio comunitario no se limita a un grupo de personas que padecen del mal de Parkinson en una determinada localidad. Dado que el resultado final de este proyecto es un software o aplicación móvil, cuaquier persona que tenga un teléfono inteligente puede descargar la aplicación y gozar de sus beneficios.

Más aún la comunidad beneficiada no se restringe a las personas que tengan el mal de Parkinson, puesto que incluso programadores o estudiantes pueden beneficiarse usando el código fuente presentado en el anexo para desarrollar otros programas, o aprender los principios de aplicaciones móviles de una forma rápida y sencilla.

En otras palabras, el presente proyecto tiene un alcance bastante amplio, principalmente debido a que no se trata de un objeto físico, sino de un objeto virtual, el cual no solo encuentra utilidad práctica en los pacientes para quienes fue diseñado, sino también encuentra utilidad en aquellos que deseen estudiarlo desde el punto de vista académico, o aquellos que deseen adaptarlo para crear otras aplicaciones similares.

3.2 Antecedentes del Proyecto

Actualmente la gran mayoría de las aplicaciones para teléfonos móviles son desarrolladas con el fin del entretener al usuario. Si bien es cierto que la diversión es una importante parte de la vida sana de un ser humano, existe una carencia considerable en las aplicaciones móviles en el área de la salud.

Dicho de otro modo, se han hecho pocos esfuerzos en usar las capacidades de procesamiento y de comunicación de los teléfonos inteligentes para ayudar a las personas que padezcan de

una determinada enfermedad, en el caso particular de este proyecto nos referimos al mal de Parkinson.

Con este trabajo se espera sentar un precedente que ayude a otros a desarrollar más aplicaciones en esta área de modo que de aquí a unos años se pueda aprovechar el gran potencial de los smartphones, los cuales día a día se encuentran más difundidos en nuestra sociedad.

CAPÍTULO 4

DESARROLLO DEL PROYECTO

4.1 Titulo Del Proyecto

El proyecto se titula *Diseño e Implementación de una Aplicación para la Detección de Temblores del Mal de Parkinson*. El cual se encuentra bajo el nombre: *Creación de una Red de Telemedicina*, con el código: SA-0809, en el sistema versión beta.

4.2 Objetivo General

Crear una aplicación para el sistema operativo Android que pueda funcionar en una amplia gama de teléfonos inteligentes con el fin de poder usar los sensores internos, y la capacidad de procesamiento del dispositivo para detectar y medir la intensidad de los temblores producidos por la enfermedad de Parkinson.

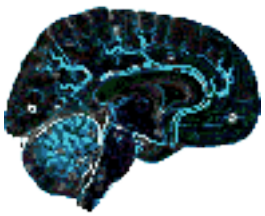

Figura 4.1: Logo de la aplicación

4.3 Objetivos Específicos

Entre los objetivos específicos se encuentran los siguentes:

1. Realizar una investigación preliminar sobre el estado de la telemedicina en Venezuela.

2. Realizar una investigación preliminar soble los esfuerzos que se han hecho a nivel mundial para medir los temblores producidos por la enfermedad de Parkinson.

3. Diseñar la estructura del software que permitirá la interacción de todos los componentes del programa.

4. Diseñar ta interfaz del programa que permitirá la fácil navegación por parte del usuario, accediendo a todas sus funciones.

5. Leer las señales del acelerómetro, procesarlas para detectar temblores, y hacer las mediciones pertinentes a los mismos tales como: intensidad, duración, etc.

6. Leer los datos insertados por el usuario tales como: nombre, edad, etc., para poder generar un reporte completo.

7. Guardar los datos pertinentes a la memoria SD del dispositivo.

8. Mandar los datos pertinentes por email a la dirección especificada por el usuario.

9. Probar la aplicación para detectar posibles fallas o errores de programación.

10. Realizar un reporte final donde se describa el trabajo realizado.

4.4 Ejecución de Actividades Realizadas

Las actividades realizadas se pueden dividir en dos grupos principales:

Investigación esto abarca todas las labores de obtención de información preliminares sobre temas como: telemedicina en Venezuela, Parkinson, los temblores y los diferentes avances que se han hecho en el área.

Programación esto abarca todas las labores relacionadas con el desarrollo del programa como tal, incluyendo tareas como: instalación del SDK, diseño, implementación, y pruebas del programa. Vale la pena destacar que los códigos fuentes del programa han sido adjuntados como anexos a este documento.

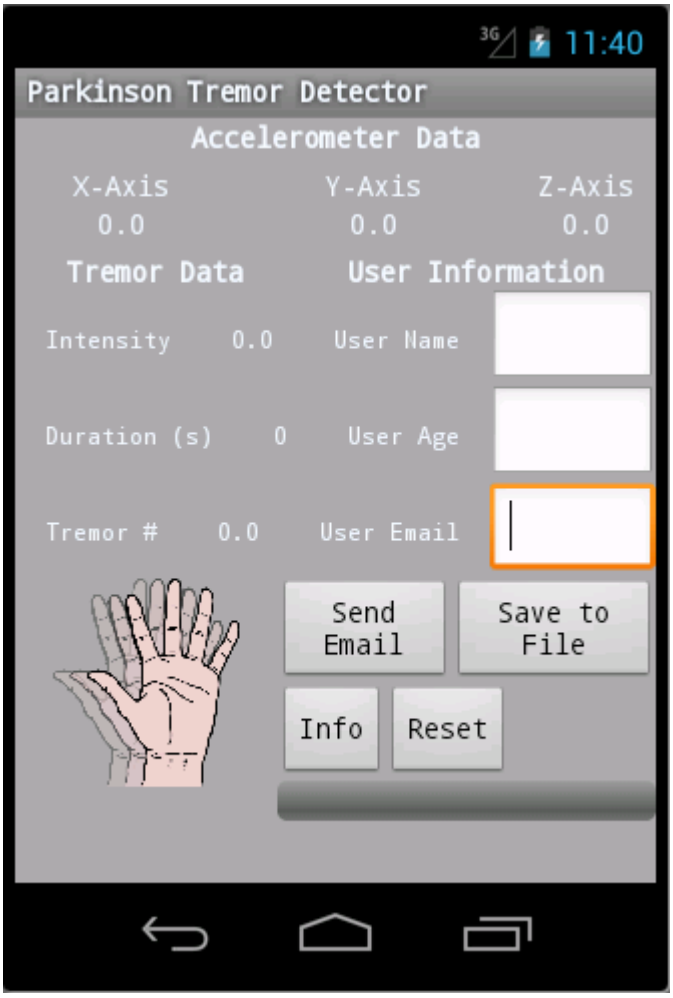

Figura 4.2: Interfaz principal de la aplicación

CAPÍTULO 5

RELACIÓN DEL PROYECTO TRABAJADO CON LA FORMACIÓN ACADÉMICA DEL ESTUDIANTE

Este proyecto esta relacionado con mi formación académica porque gracias a él he aprendido mucho sobre el área de programación de dispositivos móviles. A pesar de ya haber estado familiarizado con la programación en Java, nunca había trabajado antes con el SDK (Software Development Kit) para Android.

Por lo tanto una gran cantidad de horas fueron invertidas en aprender a usar este kit de programación para poder crear una aplicación completa desde el punto de vista de sus funcionalidades y desde el pundo de vista estético produciendo una interfaz gráfica amena e intuitiva para el usuario.

Entre los aspectos específicos que aprendí estan:

1. Installación y configuración del SDK junto con el plugin ADT para el entorno de desarrollo Eclipse.
2. Crear proyectos Android con Eclipse y a través de la linea de comandos.
3. Crear emuladores para dispositivos Android.
4. Correr los programas en modo de debug tanto en el dispositivo real como en el emulador.
5. Crear una interfaz de usuario utilizando: grupos, texto, recursos, botones, entre otros.
6. Entender la jerarquía del programa, con las diferentes actividades que lo componen y el ciclo de vida de las mismas.
7. Herramientas para ofrecer soporte para dispositivos con un tamaño de pantalla diferente.
8. Salvado de data, envío de emails, reproducción de video, lectura de sensores, etc.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1 Conclusiones

En este proyecto hemos visto como se pueden aprovechar las capacidades de procesamiento de datos, comunicación inalámbrica, y sensores internos de un teléfono móvil para producir una aplicación completa que se pueda utilizar para ayudar a las personas que sufren del mal de Parkinson.

Si bien es cierto que para poder utilizar la aplicación es necesario tener un teléfono inteligente Android, vale la pena destacar que estos dispositivos son cada vez más comúnes en el mundo. Se estima que el 27 % de los teléfonos del mundo son "smartphones", y sólo en Estados Unidos el número de usuarios sobrepasa los 100 millones.

Por esta razón podemos asegurar que la implementación de este programa en el teléfono es mucho más eficiente y tiene un alcance mucho mayor a que si se hubiera desarrollado una plataforma personalizada para cumplir con el mismo fin. De este modo usamos una plataforma de hardware estable y ampliamente difundida para correr nuestra aplicación.

6.2 Recomendaciones

Si bien es cierto que cada dispositivo tiene sensores con diferentes características, tamaños de pantalla, procesadores, etc., el programa fue diseñado para ser transparente a todas estas diferencias permitiendo que la aplicación funcione en cualquier dispositivo que cuente con las funcionalidades básicas del Android 2.2 (*Froyo*).

Dado que las construcciones de cada teléfono pueden variar, es posible que sea necesario sostener el teléfono de diferentes maneras hasta lograr un desempeño óptimo. Además se recomienda aumentar el tiempo antes de que el teléfono entre en modo de bajo consumo, para así evitar que la pantalla se apague mientras se está utilizando la aplicación.

BIBLIOGRAFÍA

[1] Boston. United states patent 7643882.

[2] Tomás Sanabria Drs. Gabriela Valero Briceño, Leopoldo Briceño-Iragorry. La telemedicina en las medicaturas rurales en venezuela.

[3] Canada el al. United states patent 4520674.

[4] Vonk el al. United states patent 5293879.

[5] Zilm et al. United states patent 4306291.

[6] Pablo Miranda-Rubén Medina Luis Núñez Ruth Marcano, Liris Gómez. La telemedicina en venezuela: una revisión.

ANEXOS

Actividades del Programa

Código Fuente 1: Actividad principal (script en Java).

```
1  package com.example.parkinson;
2
3  import java.io.BufferedWriter;
4  import java.io.File;
5  import java.io.FileWriter;
6  import java.io.IOException;
7
8  import android.os.Bundle;
9  import android.os.Environment;
10 import android.app.Activity;
11 import android.content.Context;
12 import android.content.Intent;
13 import android.hardware.Sensor;
14 import android.hardware.SensorEvent;
15 import android.hardware.SensorEventListener;
16 import android.hardware.SensorManager;
17 import android.util.Log;
18 import android.view.View;
19 import android.widget.EditText;
20 import android.widget.ImageView;
21 import android.widget.ProgressBar;
22 import android.widget.TextView;
23 import android.widget.Toast;
24
25 public class MainActivity extends Activity implements ...
       SensorEventListener {
26     // use: adb kill-server & adb start-server to work out the glitch
27     private long startTime, sampleTime, resetTime;
```

```
28      private long tremor_duration, notremor_duration;
29      private int tremor, tremorLast;
30      private int animationFrame;
31      private boolean isHighX, isHighY, isHighZ;
32      private boolean handisShaking;
33      private float maxaccX, maxaccY, maxaccZ;
34      private float minaccX, minaccY, minaccZ;
35      private float mLastX, mLastY, mLastZ;
36      private float shakeIntensity;
37      private float Ls, Li;
38      private boolean mInitialized;
39      private SensorManager mSensorManager;
40      private Sensor mAccelerometer;
41      private ProgressBar mProgress;
42      private int mProgressStatus;
43
44      /** Called when the activity is first created. */
45      @Override
46      public void onCreate(Bundle savedInstanceState) {
47          super.onCreate(savedInstanceState);
48          setContentView(R.layout.activity_main);
49          mInitialized = false;
50          mSensorManager = (SensorManager) ...
                getSystemService(Context.SENSOR_SERVICE);
51          mAccelerometer = ...
                mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
52          mSensorManager.registerListener(this, mAccelerometer, ...
                SensorManager.SENSOR_DELAY_NORMAL);
53          startTime = System.currentTimeMillis();
54          isHighX = isHighY = isHighZ = false;
55          maxaccX = maxaccY = maxaccZ = 0;
56          minaccX = minaccY = minaccZ = 0;
57          Ls = 2;
58          Li = -2;
59          tremor = tremorLast = 0;
60          sampleTime = 2000;
61          resetTime = 30000;
62          tremor_duration = 0;
63          animationFrame = 0;
```

```
64          mProgressStatus = 0;
65          handisShaking = false;
66          mProgress = (ProgressBar) findViewById(R.id.progress_bar);
67      }
68
69
70      protected void onResume() {
71          super.onResume();
72          mSensorManager.registerListener(this, mAccelerometer, ...
                SensorManager.SENSOR_DELAY_NORMAL);
73      }
74
75      protected void onPause() {
76          super.onPause();
77          mSensorManager.unregisterListener(this);
78      }
79
80      public void onAccuracyChanged(Sensor sensor, int accuracy) {
81          // can be safely ignored for this demo
82      }
83
84      public void onSensorChanged(SensorEvent event) {
85          TextView tvX = (TextView)findViewById(R.id.x_axis);
86          TextView tvY = (TextView)findViewById(R.id.y_axis);
87          TextView tvZ = (TextView)findViewById(R.id.z_axis);
88          TextView tvTremorCounter = ...
                (TextView)findViewById(R.id.tremor_counter);
89          ImageView iv = (ImageView)findViewById(R.id.imageHand);
90          float x = event.values[0];
91          float y = event.values[1];
92          float z = event.values[2];
93          if (!mInitialized) {
94              mLastX = x;
95              mLastY = y;
96              mLastZ = z;
97              tvX.setText("0.0");
98              tvY.setText("0.0");
99              tvZ.setText("0.0");
100             mInitialized = true;
```

```
101         } else {
102             float ΔX = (mLastX − x);
103             float ΔY = (mLastY − y);
104             float ΔZ = (mLastZ − z);
105             isShaking(ΔX, ΔY, ΔZ);
106             mLastX = x;
107             mLastY = y;
108             mLastZ = z;
109             tvX.setText(Float.toString(minaccX));
110             tvY.setText(Float.toString(minaccY));
111             tvZ.setText(Float.toString(minaccZ));
112             tvTremorCounter.setText(Float.toString(tremor));
113             if (handisShaking)
114             {
115                 animationFrame++;
116                 switch (animationFrame)
117                 {
118                 case 1: iv.setImageResource(R.drawable.hand1); break;
119                 case 2: iv.setImageResource(R.drawable.hand2); break;
120                 case 3: iv.setImageResource(R.drawable.hand3); break;
121                 case 4: iv.setImageResource(R.drawable.hand4); ...
                        animationFrame = 0; break;
122                 default: animationFrame = 0;
123                 }
124             }
125         }
126     }
127
128     private void isShaking(float ΔX, float ΔY, float ΔZ) {
129
130         TextView tvIntensity = ...
                (TextView)findViewById(R.id.tremor_intensity);
131         TextView tvDuration = (TextView)findViewById(R.id.tremor_duration);
132
133
134         if (ΔX > Ls)
135         {
136             if (isHighX == false)
137             {
```

```
138                 isHighX = true;
139                 tremor++;
140             }
141             if (ΔX > maxaccX)
142             {
143                 maxaccX = ΔX;
144             }
145         }
146         else if (ΔX < Li)
147         {
148             if (isHighX == true)
149             {
150                 isHighX = false;
151                 tremor++;
152             }
153             if (ΔX < minaccX)
154             {
155                 minaccX = ΔX;
156             }
157         }
158 
159         if (ΔY > Ls)
160         {
161             if (isHighY == false)
162             {
163                 isHighY = true;
164                 tremor++;
165             }
166             if (ΔY > maxaccY)
167             {
168                 maxaccY = ΔY;
169             }
170         }
171         else if (ΔY < Li)
172         {
173             if (isHighY == true)
174             {
175                 isHighY = false;
176                 tremor++;
```

```
177             }
178             if (ΔY < minaccY)
179             {
180                 minaccY = ΔY;
181             }
182         }
183 
184         if (ΔZ > Ls)
185         {
186             if (isHighZ == false)
187             {
188                 isHighZ = true;
189                 tremor++;
190             }
191             if (ΔZ > maxaccZ)
192             {
193                 maxaccZ = ΔZ;
194             }
195         }
196         else if (ΔZ < Li)
197         {
198             if (isHighZ == true)
199             {
200                 isHighZ = false;
201                 tremor++;
202             }
203             if (ΔZ < minaccZ)
204             {
205                 minaccZ = ΔZ;
206             }
207         }
208 
209         long elapsedTime = System.currentTimeMillis() - startTime;
210 
211         if (elapsedTime > sampleTime)
212         {
213             startTime = System.currentTimeMillis();
214             if (!(tremor - tremorLast > 2))
215             {
```

```
216                 maxaccX = maxaccY = maxaccZ = 0;
217                 minaccX = minaccY = minaccZ = 0;
218                 notremor_duration += elapsedTime;
219                 handisShaking = false;
220                 if (notremor_duration > resetTime)
221                 {
222                     tremor = tremorLast = 0;
223                     tremor_duration = 0;
224                     shakeIntensity = 0;
225                 }
226             }
227             else
228             {
229                 shakeIntensity = maxaccX + maxaccY + maxaccZ - minaccX ...
                        - minaccY - minaccZ;
230                 handisShaking = true;
231                 tremor_duration += elapsedTime;
232                 notremor_duration = 0;
233             }
234             mProgressStatus = (int) (shakeIntensity + 0.5);
235             mProgress.setProgress(mProgressStatus);
236             tvIntensity.setText(Float.toString(shakeIntensity));
237             tvDuration.setText(Long.toString(tremor_duration/1000));
238             tremorLast = tremor;
239         }
240         return;
241     }
242
243     public void launchreset(View view)
244     {
245         tremor = tremorLast = 0;
246         tremor_duration = 0;
247         shakeIntensity = 0;
248     }
249
250     public void writeLog(View view)
251     {
252         Toast.makeText(MainActivity.this, "Saving data to SD memory ...
                main directory", Toast.LENGTH_LONG).show();
```

```
253         String assignArr = buildMessage();
254         File root = Environment.getExternalStorageDirectory();
255         File file = new File(root, "parkinson_report.txt");
256         if (assignArr.length() > 0)
257         {
258             try {
259                     if (root.canWrite()) {
260                     FileWriter filewriter = new FileWriter(file);
261                     BufferedWriter out = new BufferedWriter(filewriter);
262                     for (int i=0; i<assignArr.length(); i++)
263                     {
264                         out.write(assignArr.substring(i, i+1));
265                         // Toast.makeText(MainActivity.this, "out: " + ...
                                assignArr.substring(i, i+1), ...
                                Toast.LENGTH_LONG).show();
266                     }
267                     out.close();
268                 }
269             } catch (IOException e) {
270                 Log.e("TAG", "Could not write file " + e.getMessage());
271             }
272         }
273     }
274
275     public void sendEmail(View view)
276     {
277         EditText userEmail = (EditText) findViewById(R.id.editEmail);
278         String userEmail_s = userEmail.getText().toString();
279         String assignArr = buildMessage();
280         Intent i = new Intent(Intent.ACTION_SEND);
281         i.setType("text/plain");
282         i.putExtra(Intent.EXTRA_EMAIL  , new String[]{userEmail_s});
283         i.putExtra(Intent.EXTRA_SUBJECT, "Parkinson Report");
284         i.putExtra(Intent.EXTRA_TEXT   , assignArr);
285         try {
286             startActivity(Intent.createChooser(i, "Send mail..."));
287         } catch (android.content.ActivityNotFoundException ex) {
288             Toast.makeText(MainActivity.this, "There are no email ...
                  clients installed.", Toast.LENGTH_SHORT).show();
```

```
289         }
290     }
291
292     public void launchInfo(View view)
293     {
294         Intent intent = new Intent(this, InformationActivity.class);
295         startActivity(intent);
296     }
297
298
299     private String buildMessage()
300     {
301         String tremorIntensity_t = getString(R.string.tremorintensity);
302         String tremorDuration_t = getString(R.string.tremorduration);
303         String tremorCounter_t = getString(R.string.tremorcounter);
304         String userName_t = getString(R.string.name);
305         String userAge_t = getString(R.string.age);
306         String userEmail_t = getString(R.string.email);
307
308         EditText userName = (EditText) findViewById(R.id.editName);
309         String userName_s = userName.getText().toString();
310
311         EditText userAge = (EditText) findViewById(R.id.editAge);
312         String userAge_s = userAge.getText().toString();
313
314         EditText userEmail = (EditText) findViewById(R.id.editEmail);
315         String userEmail_s = userEmail.getText().toString();
316
317         TextView tremorIntensity = (TextView) ...
                findViewById(R.id.tremor_intensity);
318         String tremorIntensity_s = tremorIntensity.getText().toString();
319
320         TextView tremorDuration = (TextView) ...
                findViewById(R.id.tremor_duration);
321         String tremorDuration_s = tremorDuration.getText().toString();
322
323         TextView tremorCounter = (TextView) ...
                findViewById(R.id.tremor_counter);
324         String tremorCounter_s = tremorCounter.getText().toString();
```

```
325
326         String message = userName_t + " : " + userName_s + "\n" +
327                          userAge_t + " : " + userAge_s + "\n" +
328                          userEmail_t + " : " + userEmail_s + "\n" +
329                          tremorIntensity_t + " : " + tremorIntensity_s ...
                                 + "\n" +
330                          tremorDuration_t + " : " + tremorDuration_s + ...
                                 "\n" +
331                          tremorCounter_t + " : " + tremorCounter_s + "\n";
332         return message;
333     }
334
335 }
```

Código Fuente 2: Actividad de información (script en Java).

```
1  package com.example.parkinson;
2
3  import android.app.Activity;
4  import android.net.Uri;
5  import android.os.Bundle;
6  import android.text.method.ScrollingMovementMethod;
7  import android.view.Menu;
8  import android.widget.MediaController;
9  import android.widget.TextView;
10 import android.widget.VideoView;
11
12 public class InformationActivity extends Activity {
13
14     String SrcPath0 = ...
           "rtsp://v5.cache7.c.youtube.com/CjYLENy73wIaLQm5lL4FInyOqxMYDSANFEI
15     JbXYtZ29vZ2xlSARSBXdhdGNoYI7j7fP20eu3UAw=/0/0/0/video.3gp";
16     String SrcPath1 = ...
           "rtsp://v3.cache8.c.youtube.com/CjYLENy73wIaLQnTejYGFSXWrhMYDSANFEI
17     JbXYtZ29vZ2xlSARSBXdhdGNoYI7j7fP20eu3UAw=/0/0/0/video.3gp";
18     String SrcPath2 = ...
           "rtsp://v3.cache4.c.youtube.com/CjYLENy73wIaLQncoGRdCgpIihMYDSANFEI
19     JbXYtZ29vZ2xlSARSBXdhdGNoYI7j7fP20eu3UAw=/0/0/0/video.3gp";
```

```
20
21          /** Called when the activity is first created. */
22          @Override
23          public void onCreate(Bundle savedInstanceState) {
24              super.onCreate(savedInstanceState);
25              setContentView(R.layout.activity_information);
26
27              TextView infoTextView = (TextView) findViewById(R.id.infonote);
28              infoTextView.setMovementMethod(new ScrollingMovementMethod());
29
30              VideoView myVideoView = (VideoView)findViewById(R.id.videoView);
31              myVideoView.setVideoURI(Uri.parse(SrcPath0));
32              myVideoView.setMediaController(new MediaController(this));
33              myVideoView.requestFocus();
34              myVideoView.start();
35          }
36
37      @Override
38      public boolean onCreateOptionsMenu(Menu menu) {
39          getMenuInflater().inflate(R.menu.activity_information, menu);
40          return true;
41      }
42
43  }
```