Llamadas sin marcación: ayuda para discapacitados visuales y llamadas de emergencia

Non-dialing calls: help for the visually impaired and emergency calls

Joel Hernández-Infante, María del Carmen Gómez-Fuentes*

Departamento de Matemáticas Aplicadas y Sistemas
Universidad Autónoma Metropolitana Unidad Cuajimalpa
Avenida Vasco de Quiroga 4871, Col. Santa Fe Cuajimalpa.
Delegación Cuajimalpa de Morelos, Ciudad de México, México, C.P. 05348

ingjoelinfante27@gmail.com, *mgomez@cua.uam.mx

PALABRAS CLAVE:
ayuda para ciegos, aplicaciones para celular, acelerómetro para hacer llamadas

RESUMEN

Emergency calling shortcut es una aplicación para celular que tiene el propósito de ayudar a hacer llamadas telefónicas de emergencia a las personas con discapacidad visual o ciegas, sin embargo, ésta puede ser también útil a cualquier persona. Las llamadas se establecen más ágilmente que con el método convencional. Con esta aplicación no es necesario desbloquear el teléfono, ni solicitar su ejecución en el Smartphone, pues corre en segundo plano (así como WhatsApp). Con una combinación de movimientos simples del celular es posible llamar a tres números telefónicos diferentes, previamente guardados. La innovación principal es el uso del acelerómetro para interactuar con el usuario en combinación con el sensor de proximidad para evitar las llamadas no deseadas. Hasta donde sabemos, ésta aplicación es única en su forma de operar y más robusta que las que utilizan el botón de encendido/apagado ya que es prácticamente imposible que se establezcan llamadas de emergencia sin que el usuario lo solicite. Se sabe que las aplicaciones que usan el botón de encendido/apagado para establecer las llamadas algunas veces realizan llamadas no deseadas.

KEYWORDS:
help for the blind, mobile applications, accelerometer for making calls

ABSTRACT

Emergency calling shortcut is a mobile application with the aim of helping the visually impaired or blind people to make emergency phone calls, however, it can also be useful for anyone. The proposed calling procedure is more agile than the conventional one. With this application it is not necessary to unlock the phone neither request its execution on the Smartphone because it runs in the background. It consists of a combination of simple cell phone movements to call three previously saved different phone numbers. The innovation of this work is the use of the accelerometer to interact with the user, combined with the proximity sensor to avoid undesired calls. As far as we know, this application is unique in its way of operating and more robust than those that use the on/off button since it is practically impossible to make calls without the user requesting it. Applications that use the on/off button sometimes make undesired calls.

•**Recibido:** 21 de abril de 2021 • **Aceptado:** 26 de agosto de 2021 • **Publicado en línea:** 1 de octubre de 2021

Introducción

La inclusión social y digital de los discapacitados es una tendencia actual 0. El software especializado en la ayuda para discapacitados visuales y para invidentes es un área en la que las aplicaciones presentan el reto de tener que funcionar sin que el usuario vea lo que sucede en una pantalla, y por ello van muy ligadas al control de dispositivos hardware.

Existe una gran variedad de dispositivos hardware para auxiliar a los ciegos, por ejemplo: una tablet que usa aire o líquido para empujar las letras en braille 2, un casco que describe objetos y personas para el usuario, y que emite un sonido de advertencia cuando el usuario está demasiado cerca de un obstáculo 2, un dispositivo que guía a los usuarios a una ubicación específica mediante vibración 3, etc. En cuanto a las aplicaciones para celular que sirven para ayudar a los ciegos, la mayoría se basa en el reconocimiento de patrones, por ejemplo, un identificador de objetos mediante fotos 4, o los lectores que convierten texto en voz 5. Actualmente existen aplicaciones de celular sorprendentes para ayudar a los discapacitados visuales y a los ciegos 6, muchas de estas aplicaciones funcionan con comando de voz. Sin embargo, cuando estamos ante alguna situación estresante, a veces sucede que se nos corta la voz. También sabemos que el reconocimiento de voz falla de vez en cuando y hay que repetir lo que se dice varias veces, esto no es oportuno ni cómodo en una situación de emergencia. Así es que sería muy conveniente poder establecer una llamada de emergencia sin tener que usar la voz. Emergency Calling Shortcut (ECS) es una aplicación para celular que surgió con la idea de ayudar a las personas con discapacidad visual o ciegas a hacer llamadas telefónicas de emergencia, por lo que se requiere que la llamada se establezca rápidamente sin necesidad de ver la pantalla para hacer la marcación ni tener que desbloquear el teléfono. Al diseñar nuestra aplicación, nos dimos cuenta de que ésta puede ser útil no solo para los ciegos, sino también para cualquier persona.

Por otra parte, haciendo una búsqueda de aplicaciones para hacer llamadas de emergencia en Play Store, encontramos que prácticamente todas sirven solamente para llamar a teléfonos ya preestablecidos, como el 911, la policía, los bomberos o para compartir la ubicación. La única aplicación que encontramos que se puede configurar para llamar al teléfono que el usuario decida se llama Emergency Call, en la cual la llamada de emergencia se inicia con el botón de encendido del celular. Descargamos esta aplicación y la probamos durante una semana. Observamos que ésta no funciona correctamente pues, en algunas ocasiones se producen llamadas de emergencia de manera involuntaria. Esto también esta reportado por otros usuarios y consecuentemente su calificación es muy baja. Investigando las posibles causas de este desperfecto, encontramos que la razón por la que falla es que ésta usa el botón de “encendido/apagado” para establecer la llamada. Los botones de servicio, que son: “encendido/apagado” y “volumen”, no están diseñados para interactuar con el usuario y por eso fallan cuando se usan con este propósito.

La aportación de ECS es el uso del acelerómetro para interactuar con el usuario en combinación con el sensor de proximidad para evitar las llamadas no deseadas. Estos dos dispositivos ya están incluidos en un Smartphone. Con este novedoso método, se ha logrado una aplicación bastante confiable. Por medio de gestos o movimientos del celular, la

aplicación realiza llamadas de manera rápida a algún número predeterminado por el usuario.

A continuación, explicamos la estructura de este trabajo. En la sección II mencionamos los trabajos relacionados. Hemos documentado la construcción de ECS conforme a las siguientes fases del proceso de desarrollo de software: especificación de requerimientos, en la sección III; diseño, en la sección IV; implementación, sección V; y pruebas, en la sección VI. Finalmente, en la sección VII están las conclusiones.

Trabajos relacionados

Entre los trabajos recientes dedicados a las aplicaciones de ayuda para los discapacitados visuales y ciegos, encontramos muy diferentes frentes de acción. Por ejemplo, Zhang et al. 7 entrevistan ciegos para identificar las barreras comunes que podrían abordarse con proxies de interacción como una estrategia para la reparación en tiempo de ejecución y la mejora de la accesibilidad de las aplicaciones móviles.

Siebra et al. 0 hacen un estudio para identificar los requerimientos de las personas con alguna discapacidad y hacen una propuesta que sirva como guía para elaborar requerimientos funcionales cuando se trata de construir aplicaciones móviles que garanticen la accesibilidad y la usabilidad.

Awad et al. 3 presentan una aplicación para teléfono móvil dirigida a personas con discapacidad visual llamada “Ojo Inteligente”. Ésta proporciona un conjunto de características útiles como: detección de luz, detección de color, reconocimiento de objetos y reconocimiento de billetes. Los resultados de la detección se reproducen en voz alta para que el usuario pueda escucharlos.

Csapó et al. 8 hacen una revisión de los trabajos de investigación relacionados con las tecnologías de asistencia y aplicaciones para usuarios ciegos que funcionan en plataformas móviles (celulares y tablets) y las clasifican en cuatro grupos: i) Texto, habla y mecanografía, ii) Interfaces de comandos por voz, iii) ayuda al usuario para desplazarse de un lugar a otro, y iv) juegos. Ellos reportan que las plataformas móviles más utilizadas, es decir. Google, Android y iOS de Apple, ofrecen una gran variedad de aplicaciones de asistencia mediante el uso de los sensores integrados de dispositivos móviles. Todas estas aplicaciones se basan en el uso del sonido y de la vibración. Sin embargo, no se reporta ninguna que haga uso del acelerómetro.

Especificación de requerimientos

En esta sección describimos la especificación de requerimientos de acuerdo al estándar IEEE-830 9, el cual define la estructura recomendada para una especificación de requerimientos. Se utilizó la plantilla A7 (por jerarquía funcional).

Introducción

Propósito. - Desarrollar una aplicación para teléfono celular que sirva para hacer llamadas de emergencia sin necesidad de hacer una marcación.

Alcance. - La aplicación se desarrollará para teléfonos móviles con sistema operativo Android. La versión mínima de Android para ejecutar la aplicación es API 22: Android 5.1 (Lollipop) en adelante, versiones anteriores o inferiores a la establecida no podrán iniciarla.

Definiciones, siglas y abreviaciones:

ECS: Emergency Calling Shortcut

2 Descripción Global

2.1 Perspectiva del producto. - Se requiere que una persona invidente pueda operar la aplicación.

2.2 Funciones del producto. - El usuario podrá solicitar el establecimiento de una llamada de emergencia y también podrá cancelar su solicitud.

2.3 Características del usuario. - La aplicación está dirigida a usuarios invidentes o con discapacidad visual, sin embargo, cualquier otra persona también podrá aprovecharla.

2.4 Prorratear los requisitos. - En la primera versión se podrá llamar a tres números diferentes. En versiones posteriores la aplicación podrá configurarse para incrementar la cantidad de números telefónicos a los que se puede llamar.

3 requerimientos específicos

3.1 Interfaces externas. - No aplica.

3.2 Funcionalidades

3.2.1 El usuario debe proporcionar hasta tres números diferentes para que la aplicación haga llamadas.

3.2.2 El sistema debe establecer la llamada a cada uno de los tres números proporcionados.

3.3 Requerimientos de desarrollo. - No aplica.

3.4 Requerimientos del banco de datos lógicos. - No aplica.

3.5 Restricciones de diseño. - No aplica.

3.6 Atributos del software del sistema. - No aplica.

3.7 Requerimientos funcionales

3.7.1.- Las llamadas deben poder establecerse de una manera discreta, sin necesidad de usar la voz.

3.7.2.- La aplicación debe brindar la posibilidad de establecer llamadas a números diferentes.

3.7.3.- Se podrán modificar los números preestablecidos en cualquier momento.

3.7.4.- El usuario podrá cancelar la solicitud de una llamada.

3.7.5.- Se debe reducir al mínimo la probabilidad de hacer una llamada de forma accidental.

3.7.6.- La aplicación deberá correr en segundo plano. Es decir, una vez iniciada la primera vez, no será necesario que el usuario la active cuando la quiera usar.

En la Fig. 1 se ilustran los casos de uso del usuario. El usuario puede registrar los números a los que desea hacer una llamada y modificar estos números, solicitar el establecimiento de llamada a uno de los números y confirmar la petición, o bien, cancelar la petición.

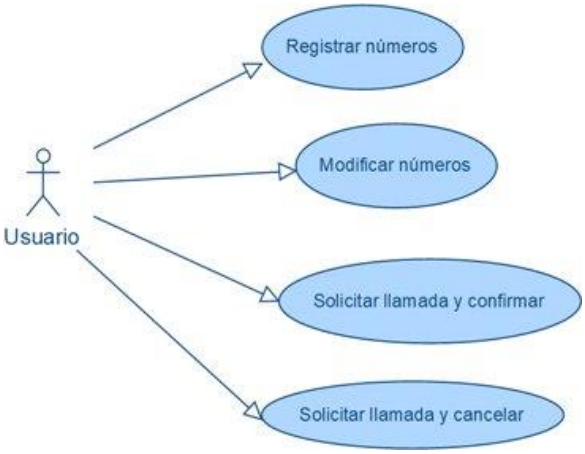

Fig. 1: Casos de uso de ECS

Diseño de la aplicación

Emergency calling shortcut tiene tres fases. La primera es la de inicio. En ella el usuario deberá indicar los números telefónicos a los cuales desea hacer llamadas. La segunda fase es la de operación, en la cual la aplicación está lista para hacer las llamadas cuando el usuario lo indique. Hay una tercera fase, que es la de inicialización, ésta se

ejecuta una única vez y consiste en otorgar los permisos necesarios para que ECS pueda funcionar. A continuación, describimos estas tres fases.

Fase 1: funcionamiento al inicio

Registro de los 3 números telefónicos

Al momento de abrir la aplicación se despliega una interfaz con el logo de presentación durante 2 segundos, posteriormente se muestra otra interfaz con 3 campos vacíos en los cuales el usuario deberá indicar los números telefónicos que elija (ver la Fig. 2).

Fig. 2: Interfaz de inicio

Una vez que se hayan registrado los números telefónicos deseados se debe seleccionar el botón de guardar para que queden almacenados en la aplicación. Cabe hacer notar que los números deben incluir la clave lada. En caso de que ya se hubieran guardado previamente algunos números telefónicos, éstos se mostrarán en los campos de texto correspondientes al momento de volver a abrir la aplicación. De esta manera el usuario tiene la posibilidad de hacer alguna modificación si así lo desea. En esta fase, los usuarios invidentes y discapacitados visualmente requerirán la ayuda de alguien que pueda configurar su aplicación con los números telefónicos deseados.

Fase 2: funcionamiento durante la operación

Cuando se abre la aplicación, ésta se inicia y se ejecuta un servicio que opera en segundo plano. Este servicio es el encargado de determinar cuándo un movimiento entra dentro del patrón preestablecido de movimiento y tiempo, y también se encarga de realizar llamadas al número que el usuario elija.

Patrón de movimiento

En la Fig. 3 se ilustra el patrón de movimiento con una secuencia de imágenes, este patrón consta de tres pasos:

Tomar el Smartphone con la pantalla de frente al usuario.

Girar el Smartphone hacia afuera con la muñeca de manera que la pantalla quede en posición opuesta al usuario.

Girar el Smartphone hacia adentro con la muñeca, para regresarlo a la posición inicial con la pantalla de frente al usuario.

Durante el diseño se hicieron experimentos con diferentes movimientos, y se tomó la decisión de utilizar este patrón porque es un movimiento antinatural, lo que de algún modo garantiza que no se efectúe accidentalmente y por tanto se realicen llamadas imprevistas. El movimiento ilustrado en la Fig. 3 es fácil de ejecutar una vez que se aprende.

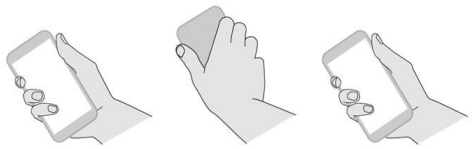

Fig. 3: Patrón de movimiento para hacer una llamada

En la Fig. 4, se ilustran los siguientes conceptos:

- Petición: Cuando se ejecuta el patrón de movimiento de la Fig. 3 para solicitar una llamada.

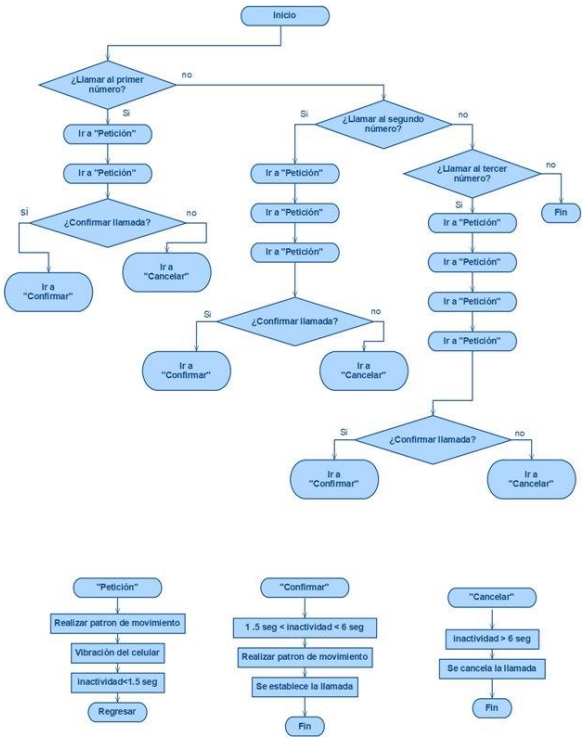

Fig. 4 Procedimiento para llamar a diferentes números.

- Confirmación: La aplicación reconoce como una confirmación para realizar la llamada cuando se lleva a cabo el patrón de movimiento de la Fig. 3 y, ha transcurrido más de 1.5 segundos de inactividad después de una petición.
- Cancelación: Si pasan más de 6 segundos de inactividad, se suspende el proceso de hacer una llamada.

- Después de un patrón de movimiento, el Smartphone vibra ligeramente con el propósito de notificar al usuario que ha realizado correctamente el movimiento y pueda llevar el conteo de movimientos que está realizando.
- Manera de hacer llamadas a diferentes números telefónicos
- ECS se puede configurar para hacer llamadas a tres números diferentes. El diagrama de flujo de Fig. 4 ilustra la forma de operar de la aplicación para hacer llamadas a diferentes números Si el usuario desea hacer una llamada al primer número telefónico de la lista se deberá hacer lo siguiente:
- Hacer 1 petición, esperar menos de 1.5 segundos para hacer otra petición. Luego, dejar pasar más de 1.5 segundos y hacer la confirmación.
- Si el usuario desea hacer una llamada al segundo número telefónico de la lista deberá hacer lo siguiente:
- Hacer 1 petición, esperar menos de 1.5 segundos para hacer otra petición, esperar menos de 1.5 segundos para hacer una tercera petición. Luego, dejar pasar por lo menos 1.5 segundos, y hacer la confirmación.
- Si el usuario desea hacer una llamada al

tercer número telefónico de la lista deberá hacer lo siguiente

- Hacer 1 petición, esperar menos de 1.5 segundos para hacer otra petición, esperar menos de 1.5 segundos para hacer una tercera petición y esperar menos de 1.5 segundos para hacer la cuarta petición. Luego, dejar pasar por lo menos 1.5 segundos, y hacer la confirmación.
- Cancelación
- En caso de perder la cuenta de las peticiones realizadas, o simplemente querer cancelar la solicitud de llamada, se debe dejar pasar 6 segundos de inactividad, es decir, no ejecutar la confirmación.
- Fase 3: Inicialización
- Cuando la aplicación se inicia por primera vez en el dispositivo móvil, se solicitará permiso para que la aplicación pueda realizar llamadas (ver Fig. 5). El usuario debe asegurarse de habilitar este permiso para que la aplicación pueda funcionar.

Fig. 5: Solicitud de permiso para llamar

Elementos clave de Emergency Calling Shortcut

ECS contiene tres elementos clave en su diseño, que son: 1) el funcionamiento de la aplicación en segundo plano y 2) la lectura del acelerómetro y 3) el sensor de proximidad, éstos se describen a continuación. La ventaja del acelerómetro y del sensor de proximidad, es que no interfieren con las funcionalidades preestablecidas del sistema operativo. Combinando adecuadamente estos dos dispositivos podemos evitar el establecimiento no deseado de llamadas.

Funcionamiento en segundo plano

ECS se debe ejecutar cuando el teléfono celular arranque y operar en segundo plano. Esto implica que la aplicación debe tener la capacidad de detectar el momento en el que el sistema operativo se inicia, lo cual sucede después de un reinicio o encendido del teléfono.

Acelerómetro

Es un componente que mide la aceleración del movimiento de los celulares sobre un espacio con ejes x-y-z (ver Fig. 6). Éste tiene la ventaja de que no interfiere con el sistema de activación de cámara o linterna de algunos dispositivos. ECS usa el acelerómetro para detectar las coordenadas en las que se mueve el dispositivo móvil. La detección del patrón para realizar una llamada se lleva a cabo mediante el registro de las coordenadas del celular. ECS está diseñado para que el patrón de movimiento se lleve a cabo sobre el plano x-y, es decir, las coordenadas en el eje de las z deben ser positivas para que se detecte un patrón de movimiento válido.

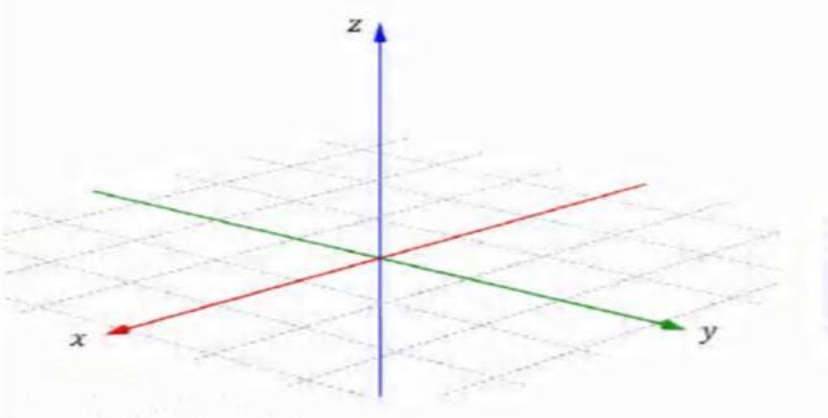

Fig. 6: Espacio tridimensional de detección de las coordenadas del celular

Sensor de proximidad

Es un componente que sirve para detectar la distancia a la que está el oído de las personas al momento de realizar una llamada telefónica. Sirve para que la pantalla se bloquee momentáneamente y no se realice una marcación accidental. ECS usa el sensor de proximidad para detectar si el celular está guardado, por ejemplo, dentro de una bolsa del pantalón o en una mochila, o si está cerca del oído del usuario. Cuando el sensor está obstruido se inhibe la detección de las coordenadas del celular. De esta manera es imposible que se establezca una llamada no deseada, incluso en el rarísimo caso de que accidentalmente se ejecute el patrón de

patrón de movimiento válido pasa al estado “n = 1” y, si no hay actividad por más de 1.5 segundos la aplicación regresa al estado inicial. Cuando se detecta un segundo patrón de movimiento se llega al estado “n = 2”, es decir, se han detectado dos peticiones. Si el usuario ejecuta otro patrón de movimiento antes de que transcurra un segundo y movimiento.

ECS realiza lecturas del sensor de proximidad para que, en caso de que se obstruya este sensor, se detenga la ejecución de lectura del acelerómetro, y por lo tanto se suspenda el registro de las peticiones de llamada. Si este sensor no está obstruido, se registran constantemente las lecturas del acelerómetro y se evalúa si las coordenadas del celular concuerdan con el patrón de movimiento.

Diagrama de Estados de ECS

En la Fig. 7 se ilustra el diagrama de estados de la operación de ECS. En el estado inicial: “Esperando Patrón” la aplicación registra constantemente las coordenadas del celular. Cuando se detecta un

medio, el estado cambiará a “n = 3” y si ejecuta otro patrón más sin esperar más de un segundo y medio, el estado será “n = 4”. Cuando el usuario tiene más de 1 segundo y medio de inactividad y el estado de ECS es “n = 2”, “n = 3” o “n = 4” entonces el nuevo estado será “Esperando confirmación”.

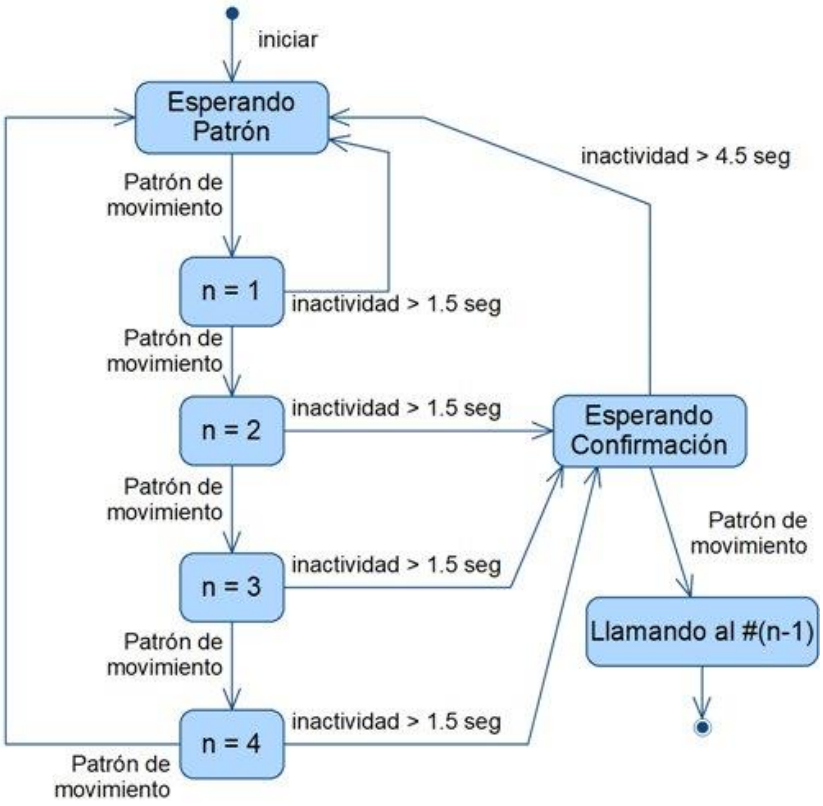

Fig. 7 Diagrama de estados de la operación de ECS

La llamada se establece al número (n-1) cuando, en este estado, se ejecuta la confirmación, es decir, un patrón de movimiento antes de 4.5 segundos de inactividad. Si trascurren más de 6 segundos de inactividad, la aplicación regresa al estado inicial.

Implementación

En esta sección incluimos el código de los tres elementos clave mencionados en el diseño. El código está escrito en Java y se establecieron reglas de codificación para facilitar su lectura. 1) Funcionamiento en segundo plano: A continuación, se muestra el código encargado de detectar cuando el sistema operativo se ha encendido después de un reinicio o apagado.

Este código se encarga de lanzar la aplicación desde el arranque del dispositivo para que ésta opere en segundo plano.

```
public class bootStart extends
BroadcastReceiver {@Override public void
onReceive(Context context,

Intent intent) {
     if
(Intent.ACTION_BOOT_COMPLETED.equals(intent.g
etAction())) {
         Intent i = new Intent(context,
Splash.class);

i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
         context.startActivity(i);
     }
   }
 }
```

2) Lectura del acelerómetro y sensor de proximidad

La lista de librerías a importar es la siguiente:

```
android.net.Uri;
android.Manifest;
android.util.Log;
android.os.IBinder;
android.os.Vibrator;
android.app.Service;
android.widget.Toast;
android.content.Intent;
android.content.Context;
android.hardware.Sensor;
android.hardware.SensorEvent;
android.hardware.SensorManager;
android.content.SharedPreferences
;
android.content.pm.PackageManager
;
android.hardware.SensorEventListe
ner;
android.support.v4.app.ActivityCo
mpat;
```

A continuación, se muestra el código del servicio encargado de realizar lecturas del sensor de proximidad. Conforme al diseño, la ejecución de lectura del acelerómetro se detiene cuando el sensor está obstruido, de lo contrario se registran las coordenadas del acelerómetro para evaluar si las coordenadas del celular concuerdan con el patrón de movimiento. Se incluye la evaluación del tiempo en que fue ejecutado el patrón y el número de veces que se repitió para determinar el número telefónico al cual se marcará en caso de que se detecte el patrón de establecimiento de llamada.

```
public class CallingShortcut extends Service
{
   public int call=0;
   public int twist=0;
   public int state=0;
   public Sensor sensor,proximity;
   public SensorEventListener SEL;
   public SensorManager SensorManager;
   public static long
tInicio=0,tFinal,tDiferencia,
tDiferenciaT,tDiferenciaC,tLapso=0,tLapsoT;
   @Override
   public int onStartCommand(Intent intent,
int flags, int startId) {
      final Vibrator vibrator =
(Vibrator)getSystemService(VIBRATOR_SERVICE);
      SensorManager =
(SensorManager)getSystemService(Context.SENSO
R_SERVICE);
      proximity =
SensorManager.getDefaultSensor(Sensor.TYPE_PR
OXIMITY);
      sensor =
SensorManager.getDefaultSensor(Sensor.TYPE_AC
CELEROMETER);
      SEL = new SensorEventListener() {
                @Override
                public void
onSensorChanged(SensorEvent sensorEvent) {

if(sensorEvent.sensor.getType()==
Sensor.TYPE_PROXIMITY){
                     if(sensorEvent.values[0]
< proximity.getMaximumRange()){
                         state=1;}
                    else{
                         state=0;}
               }
      if(sensorEvent.sensor.getType()==
Sensor.TYPE_ACCELEROMETER && state == 0) {
        if(!(tInicio==0) && !(call==0)){
           tFinal =
System.currentTimeMillis();
           tDiferencia = tFinal - tInicio;
           tDiferenciaC = tFinal - tLapso;
           tDiferenciaT = tFinal - tLapsoT;
           if(tDiferenciaC>1500 && call<2){
              call = 0;
              twist = 0;
              tInicio = 0;
        }
        if(twist == 1 && tDiferenciaT>3000){
           twist = 0;}
        if(tDiferencia>6600){
           call = 0;
           twist = 0;
           tInicio = 0;
        }
     }
     float X = sensorEvent.values[0];
     float Y = sensorEvent.values[1];
     float Z = sensorEvent.values[2];
     if (X > 0.5 && X < 6 && Y < 9.5 && Y > 1
&& Z < -2 && Z > -10 && twist == 0) {
         twist++;
         tLapsoT =
System.currentTimeMillis();
     }
     else {
        if (X > -0.5 && Y < 10 && Y > 1 && Z
> 3 && Z < 10 && twist == 1) {
           twist++;}
     }
     if (twist == 2) {
       if (call == 0) {
           call++;
           twist = 0;
           vibrator.vibrate(70);
           tInicio =
System.currentTimeMillis();
           tLapso =
System.currentTimeMillis();
       }
    else{
       if(call>0){
          tFinal =
System.currentTimeMillis();
          tDiferencia = tFinal - tLapso;
          if (tDiferencia > 1000 && call>=2)
```

```
{
            if (call < 5) {
                SharedPreferences
preferencias = getSharedPreferences("datos",
Context.MODE_PRIVATE);
                switch (call) {
                  case
2:vibrator.vibrate(500);
                   if
(!preferencias.getString("numero1",
"").equals("")) {
                              Intent i = new
Intent(Intent.ACTION_CALL,

Uri.parse("tel:"

+ preferencias.getString("numero1", "")));
                               if
(ActivityCompat.checkSelfPermission(CallingSh
ortcut.this,

Manifest.permission.CALL_PHONE)!=
PackageManager.PERMISSION_GRANTED){

Toast.makeText(CallingShortcut.this.getApplic
ationContext(),

"Datos guardadosexitosamente",
Toast.LENGTH_LONG).show();
                          }
                          else
                              startActivity(i);
                        }
                        call = 0;
                        twist = 0;
                        tInicio = 0;
                        break;
              case 3:
                        vibrator.vibrate(500);
                        if
(!preferencias.getString("numero2",
"").equals("")) {
                         Intent i = new
Intent( Intent.ACTION_CALL, Uri.parse("tel:"
+

preferencias.getString("numero2", "")));
                        if
(ActivityCompat.checkSelfPermission(CallingSh
ortcut.this,

Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
                              Toast.makeText(
CallingShortcut.this.getApplicationContext(),

"Datos guardados exitosamente",
Toast.LENGTH_LONG).show();}
                          else
                              startActivity(i);
                        }
                        call = 0;
                        twist = 0;
                        break;
              case 4:
                        vibrator.vibrate(500);
                        if
(!preferencias.getString("numero3",
"").equals("")) {
                         Intent i = new
Intent(Intent.ACTION_CALL, Uri.parse("tel:" +
preferencias.getString("numero3", "")));
                        if
(ActivityCompat.checkSelfPermission(CallingSh
ortcut.this,

Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {

Toast.makeText(CallingShortcut.this.getApplic
ationContext(),

"Datos guardados exitosamente",
Toast.LENGTH_LONG).show();}
                          else
                              startActivity(i);
                        }
                        call = 0;
                        twist = 0;
                        tInicio = 0;
                        break;
                      }
                      else {
                        call++;
                        twist = 0;
                        vibrator.vibrate(70);
                        tLapso =
System.currentTimeMillis(); }
                      }
                  }
              }
            }
          }
          @Override
           public void onAccuracyChanged(Sensor
sensor,int i) {}};

SensorManager.registerListener(SEL,proximity,
SensorManager.SENSOR_DELAY_FASTEST);

SensorManager.registerListener(SEL,sensor,Sen
sorManager.SENSOR_DELAY_FASTEST);
          return Service.START_STICKY;
      }
    @Override
    public IBinder onBind(Intent intent) {
      throw new
UnsupportedOperationException("Not yet
implemented");
    }
}
```

Pruebas de validación y de usabilidad

A continuación, se describen las pruebas de validación que sirven para demostrar el funcionamiento de los requerimientos específicos de la sección II.

PV1.- Para demostrar que se cumple con el requerimiento 3.7.1, se establece una llamada al primer número registrado ejecutando dos solicitudes y una confirmación.

PV2.- Para demostrar que se cumple con el requerimiento 3.7.2, se ejecutan tres solicitudes y una confirmación para llamar al segundo número registrado. También se ejecutan cuatro solicitudes y una confirmación para llamar al tercer número registrado.

PV3.- Para demostrar que se cumple con el requerimiento 3.7.3, se abre la aplicación y se modifica el primer número registrado, se guarda y después se establece la llamada a este primer número modificado.

PV4.- Para demostrar que se cumple con el requerimiento 3.7.4, se ejecutarán dos solicitudes y no se ejecutará la confirmación, la solicitud de la llamada deberá ser cancelada.

PV5.- Para demostrar que se cumple con el requerimiento 3.7.5, se cubre la pantalla del celular con algún objeto y se realizan dos solicitudes y una confirmación. No se deberá establecer la llamada. El teléfono no deberá vibrar, es decir, no reconocerá el patrón de movimiento. Esto implica que el celular no reconocerá el patrón de movimiento cuando el detector de proximidad indique que un objeto está cerca (cuando está dentro de una bolsa, mochila, etc.).

PV6.- Para demostrar que se cumple con el requerimiento 3.7.6, se apagará el celular. Posteriormente se enciende nuevamente y se realiza la prueba PV1.

Para que la aplicación cumpla con la finalidad para la que ECS fue pensada, se requiere que el movimiento para realizar la llamada sea claro y a la vez cómodo. Durante el diseño de la operación de ECS se realizaron pruebas para encontrar un movimiento sencillo y discreto, pero al mismo tiempo inconfundible. Dos voluntarios prestaron su ayuda para encontrar las coordenadas adecuadas para el registro del patrón de movimiento. Las coordenadas se determinaron mediante la observación del usuario cuando mueve el celular, estas coordenadas se pueden apreciar en el código. Las posiciones en las que se hicieron pruebas fueron dos: sentado y de pie. El manejo de los tiempos de espera es intuitivo. En la práctica, no es necesario que el usuario cuente el tiempo que transcurre entre una petición y otra, simplemente debe pensar en que debe hacer las peticiones “seguidas”. La vibración del celular es la indicación de que ya puede hacer otra petición. Para hacer una confirmación solo debe pensar en esperar un poco antes de ejecutarla, es útil decir, por ejemplo, la palabra “uno” antes de confirmar. Se realizaron varios experimentos en los que los usuarios ejecutaron peticiones y confirmaciones y, los tiempos de espera con los que opera ECS fueron los que se ajustaron mejor a los tiempos en los que los usuarios ejecutaron los movimientos con más naturalidad y claridad.

Conclusiones

Emergency calling shortcut (ECS) es una aplicación para celular que tiene el propósito de ayudar hacer llamadas telefónicas de emergencia a las personas con discapacidad visual o ciegas, sin embargo, ésta puede ser también útil a cualquier persona. No tener que usar la voz durante una emergencia resulta cómodo e incluso puede llegar a ser muy conveniente. ECS se puede configurar para hacer llamadas a tres números distintos. Las llamadas se establecen rápidamente mediante patrones de movimiento, sin necesidad de ver la

pantalla para hacer la marcación. Estos patrones de movimiento son sencillos, discretos y fáciles de ejecutar. Además, no es necesario desbloquear el teléfono para que ECS funcione.

Otras aplicaciones para hacer llamadas de emergencia sin usar la voz tienen el inconveniente de que en ocasiones establecen llamadas sin que el usuario lo solicite. Esto sucede porque utilizan el botón de "encendido/apagado" para establecer una llamada, y este es un botón de servicio, el cual no está diseñado para interactuar con el usuario en una aplicación.

La aplicación que aquí presentamos hace uso del acelerómetro y del sensor de proximidad para interactuar con el usuario, esto hace que funcione mejor que las ya existentes pues no establece llamadas sin que el usuario lo solicite. Estos dos componentes hardware, vienen por defecto en prácticamente todos los Smartphone.

En este trabajo se documentan las siguientes etapas del proceso de desarrollo de ECS: la especificación de requerimientos, el diseño, la implementación y las pruebas.

Si bien consideramos que ECS es muy fácil de operar, en el futuro sería interesante estudiar la percepción de más usuarios.

.

REFERENCIAS

1. Siebra, C., Gouveia, T. B., Macedo, J., da Silva, F. Q., Santos, A. L., Correia, W., Florentin, F.. Toward accessibility with usability: understanding the requirements of impaired uses in the mobile context. In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, 2017, 1-8.https://doi.org/10.1145/3022227.3022233

2. BlindNewWorld's Top 10 Assistive Tech Innovations in 2017. Recuperado el 28 de agosto de 2020 de. http://blindnewworld.org/blog/blindnewworld-top-10-assistive-tech-innovations-2017/

3. M. Awad, J. E. Haddad, E. Khneisser, T. Mahmoud, E. Yaacoub, M. Malli, "Intelligent eye: A mobile application for assisting blind people," 2018 IEEE Middle East and North Africa Communications Conference *(MENACOMM)*, 2018, 1-6, doi: 10.1109/MENACOMM.2018.8371005.

4. Kacorri, H., Kitani, K. M., Bigham, J. P., Asakawa, C. (2017, May). People with visual impairment training personal object recognizers: Feasibility and challenges. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 2017, 5839-5849.https://doi.org/10.1145/3025453.3025899

5. Neto, R., Fonseca, N. Camera reading for blind people. Procedia Technology, 2014, 16, 1200-1209. https://doi.org/10.1016/j.protcy.2014.10.135

6. Mekhalfi, M. L., Melgani, F., Zeggada, A., De Natale, F. G., Salem, M. A. M., Khamis, A. Recovering the sight to blind people in indoor environments with smart technologies. Expert systems with applications, 2016, 46, 129-138. https://doi.org/10.1016/j.eswa.2015.09.054

7. Zhang, X., Ross, A. S., Caspi, A., Fogarty, J., Wobbrock, J. O. Interaction proxies for runtime repair and enhancement of mobile application accessibility. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 2017, 6024-6037. https://doi.org/10.1145/3025453.3025846

8. Csapó, Á., Wersényi, G., Nagy, H. et al. A survey of assistive technologies and applications for blind users on mobile platforms: a review and foundation for research. J Multimodal User Interfaces, 2015, 9, 275–286. https://doi.org/10.1007/s12193-015-0182-7.

9. IEEE Software Engineering Standards Committee, "IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications", October 20, 1998. https://segoldmine.ppi-int.com/content/standard-ieee-std-830-ieee-recommended-practice-software-requirements-specifications

Acerca de los autores

Es Ingeniero en Computación por la Universidad Autónoma Metropolitana Unidad Cuajimalpa. Actualmente trabaja como ingeniero iOS. Sus principales áreas de interés son la programación reactiva, el diseño, las metodologías y las arquitecturas aplicadas al desarrollo escalable, mantenible y robusto de aplicaciones móviles nativas.

Es Doctora en Ciencias (Computación) por la Universidad Nacional Autónoma de México. Estudió Ingeniería Electrónica en la Universidad Autónoma Metropolitana (UAM). Actualmente es Profesor Asociado de tiempo completo en la UAM Unidad Cuajimalpa, en el Departamento de Matemáticas Aplicadas y Sistemas. Trabajó 8 años como ingeniero de software en la empresa Alcatel participando durante tres años y medio en el departamento de Diseño y Desarrollo de Software en Alcatel Bell, Bélgica. Su principal área de interés es la Ingeniería de Software, particularmente la Ingeniería de Requerimientos, el modelado y el desarrollo de sistemas.