Desarrollo en Google Glass para personas con discapacidad

Beca de Colaboración de Comillas
10 de Junio de 2015

José Javier González Ortiz

Índice

1 Introducción

El presente informe muestra la documentación del proyecto de Desarrollo en Google Glass para personas con discapacidad.

La investigación realizada fija el punto de partida y define los conocimientos básicos necesarios para continuar con el Proyecto.

2 Requisitos

El proyecto que se desea abordar en este trabajo de investigación consiste en dotar a personas con diferentes tipos de discapacidad motora de una interfaz especializada. Esa interfaz les permitirá hacer uso de contenidos interactivos tales como cuentos o juegos.

El problema de entrada es complejo y se puede afrontar de numerosas formas. Para poder empezar a analizar el problema se partió de una serie de suposiciones que se enumeran a continuación:

- La persona tiene una discapacidad **motriz casi completa**. Éste es un supuesto vinculado a la misma naturaleza de la plataforma empleada. Las Google Glass poseen una interfaz compleja y avanzada que permite hacer uso de sus cámaras y sensores. Para los casos en los que el paciente pudiera emplear diversos botones y actuadores, dispositivos como tablets y ordenadores especializados han cubierto ya estas necesidades. [4]
- Se va a emplear una arquitectura **cliente-servidor**. En ésta las Google Glass se emplearán únicamente como interfaz de entrada y se hará uso de un mecanismo de visualización para la salida de datos. Esto se ve motivado por la propia tecnología de las Google Glass que como ya se comentará más adelante, resultan incómodas para periodos de uso prolongado. Elegir esta arquitectura permite además modularizar el desarrollo y reducir el riesgo del mismo.
- La filosofía de la aplicación será un **juego** o un **cuento interactivo** en la que el usuario tomará una serie de decisiones y actuará en función de ellas.
- Las **interacciones** que deberá realizar el usuario no deben ser numerosas ni críticas en el tiempo, ya que la aplicación está orientada a individuos con discapacidades motoras que pueden dificultar el uso de la interfaz.
- El individuo permanecerá **quieto**. Esto viene provocado por la corta vida de la batería de las Google Glass, que imposibilitaría un uso prolongado sin cargarse continuamente.

3 Caso de uso

Con esta serie de limitaciones resulta más fácil concretar el problema, veamos un ejemplo.

Una posible aplicación que cubriría los requisitos mencionados anteriormente sería un **cuento interactivo**.

- El usuario descansaría sentado mientras ve y/o oye un cuento.
- Llegados ciertos puntos se vería obligado a tomar una decisión de entre una serie de opciones mostradas.
- Para tomar decisiones haría uso de códigos QR que serían reconocidos por las Google Glass.
- Las Google Glass traducirían el código bidimensional en una petición HTTP al servidor, el cual en función de la opción seleccionada mostraría un vídeo/audio diferente dependiendo de lo elegido por el usuario.
- A continuación el servidor respondería a las Google Glass con la cantidad de tiempo que deberán permanecer inactivas. De esta forma, llegado el momento en el que el usuario necesitase volver a introducir una acción las Google Glass estarían encendidas. El resto del tiempo permanecerán en reposo de cara a optimizar el uso de batería y ser lo menos intrusivas posible.

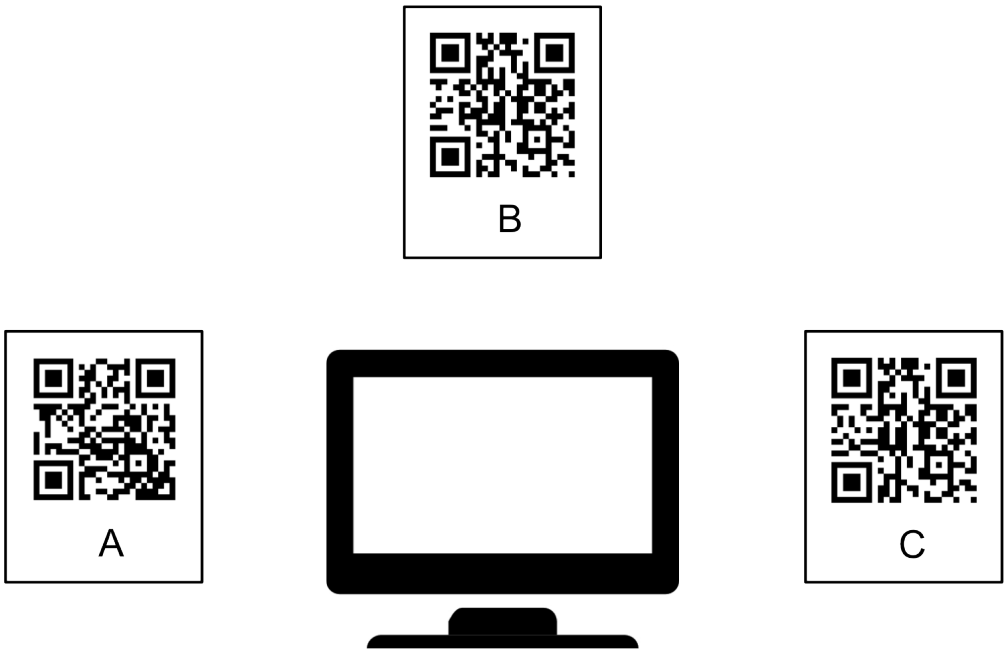

Figura 3.1: Esquema gráfico del cuento interactivo

Tal y como refleja la Figura 3.1, colocar unos simple carteles con opciones permitiría al usuario interactuar con el cuento haciendo uso de la mirada. Cabe mencionar que el desarrollo del cuento no sería necesariamente dedicado, porque gran multitud de este tipo de aventuras gráficas se podrían trasladar a este formato. Esto permite reutilizar contenidos ya creados y disponibles bajo licencias abiertas y libres.

Se recalca asimismo el hecho de que tras varios ensayos usando este esquema, los resultados llevaron a realizar varias observaciones.

- Conviene que las opciones no sean más de 3 o 4, ya que pasado ese umbral el usuario puede confundirse con las opciones y el hardware de reconocimento de QR dejar de funcionar correctamente.
- El tamaño y posición de los QR es un factor crítico ya que se debe garantizar que en la zona de visión sólo haya un QR al tiempo. Si esto no se cumple, no se podrá garantizar que se reconozca ninguno de los QR y aún en el caso de que se reconozca uno, no se podrá predecir cual será. Antes de establecer este esquema será necesario determinar el tamaño correcto de los QR así como la distancia a los mismos. Una vez realizado esto se podrá garantizar que el tamaño y posición de los QR son adecuados proporcionando una correcta experiencia de usuario.

3.1 Ejercicios didácticos

Se torna interesante analizar si podemos reutilizar este esquema para un contenido más educativo que lúdico. Empleando simples cuentas aritméticas podemos convertir este esquema en un simple mecanismo para aprender a sumar, restar o multiplicar.

Por ejemplo, dada la siguiente expresión:

$$7 \times 8 - 12 = \quad ?$$
$$A) \quad 34$$
$$B) \quad 44$$
$$C) \quad 48$$

El usuario podrá calcular la opción correcta y para responder mirará el QR correspondiente a la misma, en este caso la opción *B*.

Bajo este esquema podemos traducir numerosas cantidades de ejercicios didácticos que de otra forma serían inaccesibles a este tipo de usuarios. Variando los contenidos se pueden lograr todo tipo de ejercicios, desde ortografía y aritmética básica hasta cultura general. En caso de que el usuario se equivocase, resultaría conveniente mostrar una simple animación audiovisual que explicase la solución correcta.

4 Plataforma de desarrollo - Google Glass

Antes de continuar con el análisis del proyecto resulta imprescindible definir ante qué tipo de plataforma nos encontramos tanto desde el punto de vista del usuario como del desarrollador.

4.1 Visión de usuario

Desde el punto de vista del usuario, las Google Glass entran en el ámbito de los dispositivos vestibles o *wearables* que empiezan a ser cada vez más populares en la sociedad actual. Este tipo de dispositivos se caracterizan por ser sistemas electrónicos reducidos mediante los cuales el usuario puede obtener notificaciones asíncronas de forma menos intrusiva que un dispositivo móvil. Sin embargo, esta característica también les limita ya que la entrada de datos se ve fuertemente disminuida. Dispositivos similares como los *smartwatches* sufren de este mismo problema.

Las Google Glass tienen una serie de **mecanismos de salida**:

- **Prisma** - Una de sus tecnologías distintivas, las Google Glass poseen de un prisma que se sitúa sobre el ojo derecho y que permite al usuario ver un pequeño display interactivo a color. La principal limitación a tener en mente es que el prisma no está pensado para una visualización prolongada ya que esto puede inducir a fatiga visual.

  El display mostrado está compuesto por una *timeline* continua en la que se encuentran las *Live Cards*. Éstas muestran información sobre notificaciones y se actualizan constantemente para dotar al usuario de información en tiempo real. Cualquier otra aplicación dedicada se denomina *Immersion Activity* y nos impide interactuar con el resto de las Glass mientras no salgamos de la misma, de ahí el concepto de inmersión.

- **Altavoces** - Las Google Glass pueden reproducir contenido acústico sin problema, pudiendo ver vídeos breves cómodamente. Para mejorar la calidad del audio se puede emplear un auricular que se conecta directamente a las gafas.

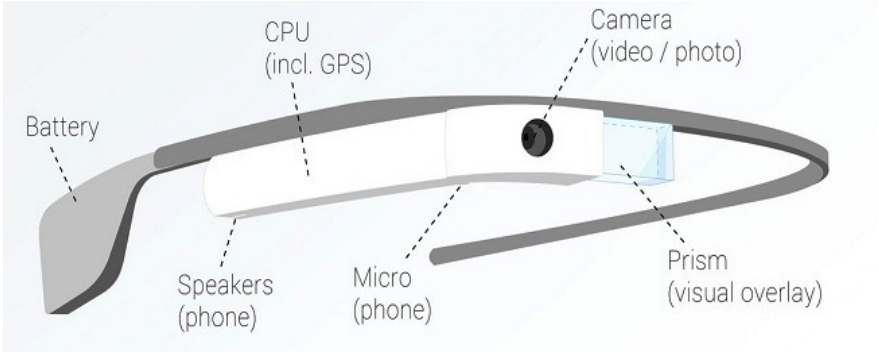

Figura 4.1: Esquema hardware de las Google Glass

Las Glass hacen uso de una serie de **mecanismos de entrada** para que el usuario pueda introducir órdenes o datos. Estos resultarán cruciales más tarde desde el punto de vista del desarrollador ya que partiremos de los mismos para poder interpretar las entradas del usuario.

- **Sensores táctiles** - La navegación de las Google Glass está diseñada para ser por defecto mediante una serie de controles táctiles predefinidos. Para un usuario medio de *smartphone* estos controles resultan intuitivos a la vez que técnicamente exigentes ya que la superficie para ejecutarlos es mínima. La zona táctil se sitúa en la parte lateral de las gafas.

- **Cámara** - Probablemente el punto de entrada más relevante de cara a datos. Las Google Glass no permiten entrada de caracteres alfanuméricos ni la conexión de teclados inalámbricos, con lo cual tan trivial como la introducción de la contraseña de una red WiFi requiere del escaneo de un código QR. Estos códigos bidimensionales se tornan muy relevantes a la hora de introducir direcciones web o contactos.

- **Micrófono** - Las Google Glass permiten el reconocimiento del habla para poder dirigirse hacia las mismas mediante comandos. Los comandos se reconocen únicamente en inglés y están limitados a una serie de palabras reservadas predefinidas por Google. En el futuro se planea ampliar el diccionario de *keywords* utilizables para comandos.

- **Sensores de movimiento** - Al igual que la mayoría de los dispositivos hoy en día, las Google Glass disponen de acelerómetros y giróscopos que le permiten detectar el movimiento del dispositivo. El uso de estos movimientos apenas es empleado por las aplicaciones, a diferencia de la mayoría de dispositivos modernos. Esto se debe a que no es fácil de monitorizar e interpretar, motivo que ha llevado a descartarlo como punto de entrada en el proyecto. Su mayor uso radica en suspender las gafas cuando notan que no las lleva nadie.

- **Smartphone** - Aunque parezca contradictorio, el punto de entrada insalvable para las Google Glass es el smartphone. Tal y como ya se ha mencionado, al igual que un *smartwatch*, las Google Glass no son un dispositivo independiente, sino que complementan la experiencia ofrecida por un terminal móvil. Por ello resulta comprensible que la configuración e instalación de aplicaciones se realice desde un *smartphone* asociado.

4.2 Visión de desarrollador

Lo primero que deberá saber todo desarrollador de Google Glass es el tipo de entorno en el que se encuentra.

Las Google Glass se programan en un dialecto de Android, específico para las Google Glass. En palabras de los propios desarrolladores de las Google Glass: "No vale con portar aplicaciones hechas en Android, hace falta una aproximación totalmente nueva".

El entorno de desarrollo parte de una base sólida de Android sobre la que se añaden nuevos tipos de interfaces, tales como las *Cards* (pantallas virtuales específicas para las Google Glass). Se añaden también nuevas formas de presentar la información así como nuevas entradas de datos. Éstas ya han sido analizadas previamente desde el punto de vista de usuario. Desde el punto de vista de desarrollador cabe mencionar varias cosas.

- Los sensores táctiles son esenciales a la par que complejos lo cual dificulta la navegación para el usuario medio, por lo que queda descartado su uso para personas con discapacidad.
- La cámara se muestra como segunda mejor opción ya que es una característica que no es tan accesible en otros dispositivos y que al hacer uso de códigos QR nos desvincula en cierto modo de la plataforma.
- Recordemos que toda aplicación desarrollada para las Google Glass pertenece a una de dos categorías: *Live Card* o *Immersion Activity*. Dado que queremos una interfaz dedicada y customizada emplearemos una arquitectura basada en *Immersion Activity*.
- El SDK empleado en este trabajo de investigación ha sido el GDK (*Glassware Development Kit*) suministrado por el IDE de Google Android Studio. A día de hoy es el que mejor soporta el desarrollo con Google Glass ya que el resto de IDE/SDK no cubren aún la casuística introducida por este dispositivo.

La plataforma de desarrollo para las Google Glass no está aún ni estable ni consolidada. Al ser un dispositivo único en su clase y al no estar todavía disponible al público, su ecosistema es considerado precario por la industria ya que presenta muchos inconvenientes y dificultades. Existe documentación, pero los tutoriales o ejemplos son muy escasos y desactualizados. Esto nos lleva a señalar varios inconvenientes sobre el entorno de desarrollo:

- El prototipo sobre el que se basó este estudio ha sido discontinuado, sin embargo mientras se lanzaban actualizaciones de software, estas no garantizaban la retrocompatibilidad y cambiaban elementos fundamentales sobre el comportamiento de determinadas clases.

  Por ejemplo, la clase Java que maneja las vistas, *Card*, fue declarada como *deprecated* y ahora se debe emplear un constructor mediante la clase *CardBuilder*. Esto como se puede apreciar es un cambio sustancial y será uno de los muchos aspectos que se deberá tener en cuenta a la hora de leer documentación de terceros.

- Al no garantizar la retrocompatibilidad, Google ha provocado que mucho código de ejemplo no funcione correctamente. Es necesario tener este factor en mente a la hora de emplear librerías de terceros o determinadas llamadas de sistema, ya que los problemas pueden residir en las versiones incompatibles de software.

- A pesar de estar basada en Android, en la mayoría de las ocasiones no podemos emplear la librerías de código desarrolladas para Android directamente. En nuestro caso en particular, esto se traduce en que si deseamos emplear una librería ya existente para leer QR como ZXing deberemos buscar una adaptación para las Google Glass como BarcodeEye.

Por todo lo expuesto anteriormente, se recomienda maximizar el control del servidor de cara a minimizar las incompatibilidades posibles debidas a las Google Glass. Estas se comportarán como un lector de QR inteligente, que será capaz de hacer peticiones y de activarse cuando sea oportuno, sin llegar a requerir de la interacción manual del usuario.

5 Diseño e Implementación

Conocidas las limitaciones del usuario y de la plataforma se puede pasar a definir, desde el punto de vista técnico, el tipo de programa que queremos desarrollar.

Partiendo del ejemplo expuesto anteriormente, se desarrollará para las Google Glass un programa mínimo que actúe como una entrada de datos de los movimientos de cabeza del usuario. Estos serán interpretados como distintas opciones tal y como ya se ha descrito.

5.1 Ejemplo

Se diseñó un experimento que requiriese la mayoría de la funcionalidad necesaria para comprobar la correcta viabilidad del proyecto.

- El experimento se centraba en una interfaz visual basada en un cubo tridimensional de LEDs. Esta interfaz era controlada por un pequeño microprocesador.
- El microprocesador era gobernado por un Servidor Web programado en Java. Dicho servidor web recibía peticiones HTTP y las interpretaba, enviando las órdenes correspondientes al microprocesador.
- Al emplear peticiones web, cualquier navegador podía interactuar con el Cubo LED, otorgando portabilidad al diseño.
- Para facilitar el acceso desde terminales móviles se codificaron QRs, que contenían dichas URLs y que podían ser leídos por cualquier dispositivo móvil.
- Para que las Google Glass pudiesen leer los distintos QRs ubicados en la sala, se empleaba un software de terceros (*QRLens* [2]). Dicha aplicación necesitaba de cierto control manual, así como de acceso a internet, características que impiden usarlo directamente en el proyecto a desarrollar.

* La Figura 5.1 muestra un esquema de este comportamiento.

Un aspecto relevante que se descubrió al realizar este experimento es que el navegador de las Google Glass se trata de una versión disminuida y restringida de Chrome. Se deberá tener en cuenta esta limitación si se desea emplear algún tipo de contenido dinámico pues cabe la posibilidad que el navegador de las Glass no sea capaz de renderizarlo.

La navegación de páginas se sustenta en el ángulo de visión. El usuario debe situar dos dedos de forma sostenida y mover la cabeza en la dirección que quiere desplazar la página web. El resultado es una navegación contraintuitiva, por lo que se recomienda encarecidamente no apoyarse en la navegación de páginas web como mecanismo de presentación de información, y mucho menos considerarlo para las personas a las que está destinada la aplicación.

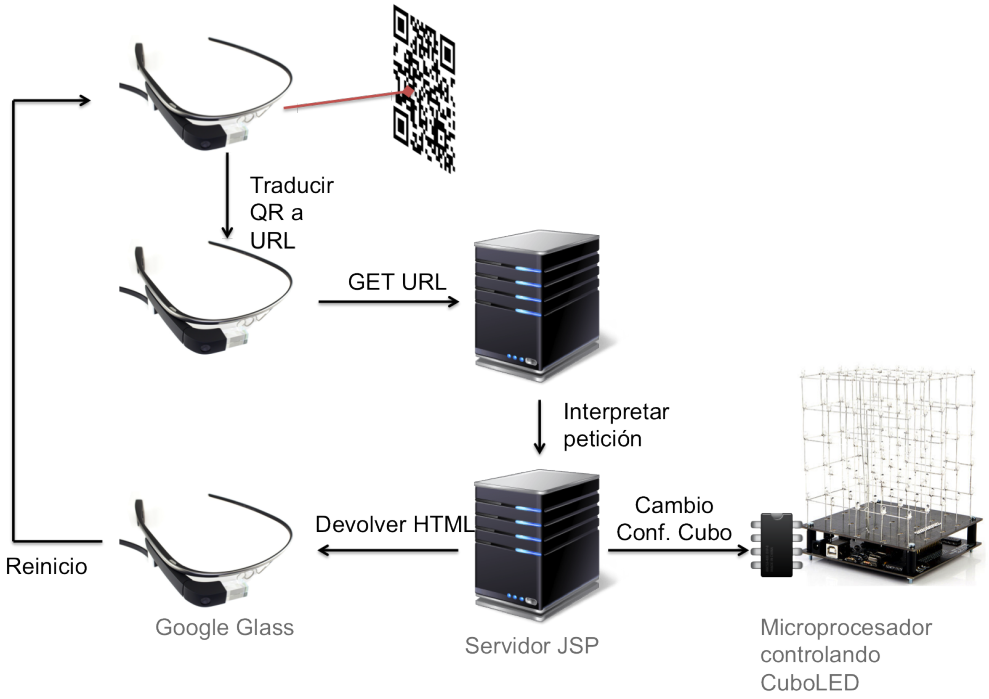

Figura 5.1: Esquema de secuencia del ejemplo del CuboLED

5.2 Diseño

Tras el análisis de programa de ejemplo podemos apreciar que sólamente necesitamos solventar dos problemáticas:

- Se requiere una aplicación de QR específica y dedicada, que no abra el navegador pero sí que haga la petición GET HTTP y con la respuesta suspenda a las Glass un tiempo determinado. Esta aplicación no deberá requerir de ningún control manual debido a lo sensibles que tienden a ser los controles de las Google Glass.

  Para dar solución a este problema resulta conveniente el uso de librerías ya existentes de reconocimiento de códigos de barras y QR tal como ZXing [3] o BarcodeEye [1]. Este último es un *port* de ZXing para Google Glass.

  El funcionamiento es tal que cada vez que la aplicación lea un código QR, realizará la petición HTTP al servidor. Si la respuesta es satisfactoria, se procederá a comunicar al usuario la opción que ha escogido, y a continuación las Glass entrarán en estado de reposo el tiempo que le indique el servidor.

  Una vez pasado este tiempo, el usuario deberá tomar una decisión por lo que será necesario que las Google Glass salgan del reposo y activen la cámara. Así el usuario sabrá que ya puede mirar hacia el QR que desee. Si por el contrario, el servidor responde con algún tipo de error, el programa no llegará a salir de la cámara y solicitará al usuario que vuelva a intentarlo.

- Hace falta disponer de una interfaz de administración, con el que la persona encargada del cuento/juego pueda configurar distintos aspectos del mismo, permitiendo la parametrización del código y de los contenidos. Esto resulta fácil de conseguir si lo planteamos desde el punto de vista del servidor web. Se empleará un navegador web que accederá a URLs reservadas. Se podrá así configurar fácilmente distintos aspectos del código.

5.3 Implementación

5.3.1 Peticiones HTTP

El primer problema a resolver son las peticiones HTTP. El módulo QR de la aplicación nos entregará la dirección URL con parámetros incluidos desde los que deberemos partir.

Para realizar esta petición nuestro programa deberá disponer de acceso a la red por lo que será necesario agregar la siguiente línea al archivo `AndroidManifest.xml`

```
<uses-permission android:name="android.permission.INTERNET" />
```

Una vez concedido acceso lo más sencillo es emplear el cliente de Apache que incluye Android por defecto. Un ejemplo genérico se puede consultar en el Código 5.1.

Código 5.1: Código Ejemplo Petición HTTP (http.java)

```
// url será el valor de la URL a la que deseamos acceder
//     esta cadena sera generada por el modulo de QR
String regex = "^(https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-
    zA-Z0-9+&@#/%=~_|]";
if(url.matches(regex))
{
    HttpClient httpclient = new DefaultHttpClient();
    HttpResponse response = httpclient.execute(new HttpGet(url));
    StatusLine statusLine = response.getStatusLine();
    if(statusLine.getStatusCode() == HttpStatus.SC_OK){
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        response.getEntity().writeTo(out);
        String responseString = out.toString();
        out.close();
        // Aqui iria la logica sobre la respuesta recibida
    } else{
        // Se cierra la conexion
        response.getEntity().getContent().close();
        throw new IOException(statusLine.getReasonPhrase());
    }
}
```

Este código permitirá realizar una petición HTTP sobre la cadena de caracteres obtenida. Al mismo tiempo, ejecuta una comprobación Regex sobre el valor de la cadena de caracteres de la URL, para no realizar la petición en caso de que no se trate de una dirección web. De querer limitar el ámbito sólo a direcciones privadas, con modificar la condición Regex sería suficiente.

5.3.2 Lectura de QR

Lo primero que deberemos hacer es conceder acceso a la cámara para lo que hará falta modificar una vez más nuestro `AndroidManifest.xml` para que incluya los permisos de cámara.

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Para la lectura de QR necesitamos hacer uso de una librería externa que implemente el conjunto de primitivas de reconocimiento de códigos bidimensionales. A la hora decidir qué librería de QR emplear se ha optado por la versión específica para Google Glass de ZXing. Se ha descartado el uso de BarcodeEye debido a que no permite el uso mediante *Intent* y por lo tanto no la podríamos ejecutar dentro de nuestra propia aplicación.

Deberá comenzarse por configurar correctamente el fichero de `build.graddle Module app` para que importe la libreria ZXing. Para ello será necesario añadir las líneas mostradas en el Código 5.2.

Una vez importada la librería, en la actividad principal se deberá tener un código muy similar al mostrado en el Código 5.3 incluido a continuación. Se comprueba que ahora se obtiene la cadena de caracteres a partir de la imagen, y es aquí donde se insertará el código para realizar la petición HTTP.

Código 5.2: Líneas a añadir a `build.graddle Module app`

```
repositories {
    mavenCentral()

    maven {
        url "https://raw.github.com/embarkmobile/zxing-android-
   minimal/mvn-repo/maven-repository/"
    }
}
dependencies {
     // Zxing libraries
    compile 'com.embarkmobile:zxing-android-minimal:2.0.0@aar'
    compile 'com.embarkmobile:zxing-android-integration:2.0.0@aar
   '
    compile 'com.google.zxing:core:3.0.1'
}
```

Código 5.3: Codigo Ejemplo Lectura de QR (qr.java)

```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    IntentIntegrator integrator = new IntentIntegrator(this);
    integrator.initiateScan();

    // Aqui se procesaria algun tipo de notificacion si se desea
}

//Cuando se lea un codigo QR se realizará la llamada a la aplicacion
protected void onActivityResult(int requestCode, int resultCode, Intent
    data)
{
    IntentResult scanResult = IntentIntegrator.parseActivityResult(
   requestCode, resultCode, intent);
    if (scanResult != null) {
        String url = scanResult.getContents();
        // Aqui se encontraria el codigo de la peticion http
        // ...
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

5.3.3 Control de Flujo

El último problema está vinculado a que la operación de leer QR es asíncrona. Esta asincronía impide determinar el tiempo que la aplicación va a permanecer en la llamada al sistema. Esto fuerza a definir la actividad principal como bucle sin fin que realiza *Intents* a una actividad secundaria que incluye la funcionalidad de HTTP y QR ya descrita.

En el punto actual de desarollo, suspender las Google Glass cierra el contexto de la aplicación. Como no hay ninguna manera de lanzar una aplicación al salir del reposo, la funcionalidad de reposo debe ser retrasada hasta que las Glass permitan esta característica. Sin embargo, este inconveniente no afecta al comportamiento general de la aplicación, ya que era una característica enfocada a ahorrar batería durante su uso.

5.3.4 Integración

Una vez definidos los módulos y descrita la funcionalidad básica se ha procedido a la integración de los módulos programados y su correspondiente depuración. El proyecto asociado se encuentra adjunto a este informe. En el Código 5.4 se muestra la actividad principal que se ha desarrollado.

Esta aplicación tiene un problema al mostrar el éscaner de QR ya que la pantalla aparece partida en cuatro fragmentos horizontales. El problema viene provocado por un desajuste de los valores del ancho de la cámara, lo que genera una pérdida de sincronización vertical.

Tras varios intentos de depurar la aplicación no se ha logrado determinar el origen del fallo. La documentación en la Wiki de ZXing clasifica este *bug* como pendiente de resolución sin explicar un motivo concreto acerca de por qué sucede.

Código 5.4: Actividad Principal de la aplicación

```
1  package com.qrimpaired;
2
3  import android.app.Activity;
4  import android.content.Context;
5  import android.content.Intent;
6  import android.media.AudioManager;
7  import android.os.Bundle;
8  import android.view.View;
9  import android.view.ViewGroup;
10 import android.widget.AdapterView;
11
12 import com.google.android.glass.media.Sounds;
13 import com.google.android.glass.widget.CardBuilder;
14 import com.google.android.glass.widget.CardScrollAdapter;
15 import com.google.android.glass.widget.CardScrollView;
16 import com.google.zxing.integration.android.IntentIntegrator;
17 import com.google.zxing.integration.android.IntentResult;
18
19 import org.apache.http.HttpResponse;
20 import org.apache.http.HttpStatus;
21 import org.apache.http.StatusLine;
22 import org.apache.http.client.HttpClient;
23 import org.apache.http.client.methods.HttpGet;
24 import org.apache.http.impl.client.DefaultHttpClient;
25
26 import java.io.ByteArrayOutputStream;
27 import java.io.IOException;
28
29 public class MainActivity extends Activity {
30
31     private CardScrollView mCardScroller;
```

```
32      private View mView;
33
34      @Override
35      protected void onCreate(Bundle savedInstanceState)
36      {
37          super.onCreate(savedInstanceState);
38
39          mView = buildView();
40
41          mCardScroller = new CardScrollView(this);
42          mCardScroller.setAdapter(new CardScrollAdapter() {
43              @Override
44              public int getCount() {
45                  return 1;
46              }
47
48              @Override
49              public Object getItem(int position) {
50                  return mView;
51              }
52
53              @Override
54              public View getView(int position, View convertView,
    ViewGroup parent) {
55                  return mView;
56              }
57
58              @Override
59              public int getPosition(Object item) {
60                  if (mView.equals(item)) {
61                      return 0;
62                  }
63                  return AdapterView.INVALID_POSITION;
64              }
65          });
66          // Handle the TAP event.
67          mCardScroller.setOnItemClickListener(new AdapterView.
    OnItemClickListener() {
68              @Override
69              public void onItemClick(AdapterView<?> parent, View view,
    int position, long id) {
70                  // Plays disallowed sound to indicate that TAP actions
    are not supported.
71                  AudioManager am = (AudioManager) getSystemService(
    Context.AUDIO_SERVICE);
72                  am.playSoundEffect(Sounds.DISALLOWED);
73              }
```

```
74          });
75          setContentView(mCardScroller);
76
77          IntentIntegrator integrator = new IntentIntegrator(this);
78          integrator.initiateScan();
79
80
81          // Aqui se procesaria algun tipo de notificacion si se desea
82      }
83
84      //Cuando se lea un codigo QR se realizará la llamada a la
    aplicacion
85      protected void onActivityResult(int requestCode, int resultCode,
    Intent intent)
86      {
87          IntentResult scanResult = IntentIntegrator.parseActivityResult(
    requestCode, resultCode, intent);
88          if (scanResult != null) {
89              String url = scanResult.getContents();
90              String regex = "^(https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~_
    |!:,.;]*[-a-zA-Z0-9+&@#/%=~_|]";
91              if(url.matches(regex))
92              {
93                  try {
94                      HttpClient httpclient = new DefaultHttpClient();
95                      HttpResponse response = httpclient.execute(new
    HttpGet(url));
96                      StatusLine statusLine = response.getStatusLine();
97                      if (statusLine.getStatusCode() == HttpStatus.SC_OK)
     {
98                          ByteArrayOutputStream out = new
    ByteArrayOutputStream();
99                          response.getEntity().writeTo(out);
100                         String responseString = out.toString();
101                         out.close();
102                         // Aqui iria la logica sobre la respuesta
    recibida
103                     } else {
104                         // Se cierra la conexion
105                         response.getEntity().getContent().close();
106                         throw new IOException(statusLine.
    getReasonPhrase());
107                     }
108                 }catch(Exception e)
109                 {
110                     // TO DO: Tratamiento de excepciones
111                 }
```

```
112                 }
113             }
114             super.onActivityResult(requestCode, resultCode, intent);
115         }
116 
117         @Override
118         protected void onResume() {
119             super.onResume();
120             mCardScroller.activate();
121         }
122 
123         @Override
124         protected void onPause() {
125             mCardScroller.deactivate();
126             super.onPause();
127         }
128 
129         private View buildView() {
130             CardBuilder card = new CardBuilder(this, CardBuilder.Layout.
    TEXT);
131 
132             card.setText(R.string.hello_world);
133             return card.getView();
134         }
135 
136 }
```

6 Perfil

A continuación se recogen los conocimientos requeridos para continuar el desarrollo del proyecto.

Las Google Glass se han definido como una ramificación de la familia Android de dispositivos con unas peculiaridades propias. Algunas de estas peculiaridades son las interfaces mediante *Cards* y el hecho de estar menos orientadas a eventos de entrada. Esto hace necesario el uso de clases y estructuras específicas para complementar las nociones de Android.

Idealmente sería conveniente tener experiencia en el desarrollo de aplicaciones para Google Glass. Sin embargo, la novedad de la plataforma lo dificulta, y no resulta indispensable.

Los conocimientos básicos recomendados para continuar con el desarrollo de la aplicación se listan a continuación.

- **Amplia experiencia en el ecosistema Android**
  Se recomienda estar familiarizado con el entorno de desarrollo y el kit de desarrollo de software de Android Studio, necesario para el desarrollo de software para Google Glass.

- **Conocimiento sobre la gestión de proyectos en el entorno Java/Android** por medio del gestor *Graddle*
  Maven no está disponible aún para las Google Glass. El uso de código de terceros es imprescindible, y para la inclusión de las librerías utilizadas se necesita un conocimiento base.

* Los anteriores requisitos tienen su origen en que el proceso de aprendizaje para el desarrollo en el ecosistema Android no puede completarse fácilmente en un intervalo de tiempo relativamente corto. En consecuencia, se reitera la importancia de tener experiencia en el desarrollo de aplicaciones Android nativas.

- **Nociones de arquitecturas web**
  La segunda parte de la implementación requiere del uso de un servidor web. El aplicativo web empleará funciones básicas y no requerirá de grandes módulos web. Igualmente debe tenerse en cuenta que familiarizarse con una arquitectura web requiere cierto tiempo.

7 Consideraciones Finales

- En este punto de desarrollo de la aplicación el programa posee la funcionalidad requerida. El código de la aplicación compila, y se ejecuta y realiza satisfactoriamente la petición HTTP. Existe sin embargo un error a la hora de mostrar el escáner en el prisma, que impide que se visualice correctamente. A pesar de ello, se reconoce correctamente el QR si la posición y orientación del código son las adecuadas.

  Dicho fallo es reconocido por los desarrolladores de la librería aunque a día de hoy se desconoce como solucionarlo. Una posible solución alternativa sería emplear otra librería de gestión de QRs.

- Se ha indicado anteriormente que las Google Glass son un producto en fase de desarrollo y no disponible al gran público. Google ha modificado la API vinculada al GDK de forma notable tratando de mejorar la experiencia de usuarios y desarrolladores. Como consecuencia de estos cambios, gran parte del código de ejemplo existente en Internet no es directamente utilizable.

- El lado del servidor queda pendiente. La aplicación más sencilla de implementar sería la mencionada en el apartado 3.1. Posteriormente se podrían añadir a esta funcionalidad los contenidos gráficos o audiovisuales. Para la implementación se recomienda una solución basada en PHP o *Java Servlet Pages* que simplifique al máximo las necesidades de desarrollo.

- La arquitectura cliente-servidor utilizada permite adaptar la aplicación para utilizar otra interfaz de entrada si así se desea. El dispositivo utilizado debe tener cámara y disponer de un cliente HTTP con el que realizar las peticiones al servidor. La arquitectura propuesta permite implementar la lectura de QR en el lado del servidor. Debido a lo anterior, no es necesario que la interfaz de entrada disponga de las librerías o la capacidad de procesamiento necesarias para leer QR.

Las consideraciones anteriormente expuestas se han elaborado a 11 de junio de 2015, por lo que se recomienda realizar una investigación del estado de las Google Glass y su programa para desarrolladores antes de continuar con el proyecto. De este modo se podrán emplear los recursos más actualizados disponibles.

A Referencias y Fuentes de Información

Debido a lo escasa que es la documentación relacionada con desarrollo de las Google Glass se ha considerado conveniente reflejar fuentes de información para futura referencia.

- **A Tutorial for Aspiring Google Glass Developers: Building Your First Glass App** - Tutorial magnífico para iniciarse en el desarrollo de Google Glass. No es 100% funcional pero con un poco de ajuste permite hacer un simple Hello World.
  `http://www.toptal.com/google-glass`
- **Glassware Development Kit** - API pública sobre el SDK específico que poseen las Google Glass basado en Android. Contiene mucha información tanto de la versión actual como de las previas.
  `https://developers.google.com/glass/`
- **StackOverflow GDK** - Categoría específica de la famosa plataforma Stack Overflow. Especialmente útil a la hora de resolver problemas de configuración y conflictos entre versiones.
  `http://stackoverflow.com/questions/tagged/google-gdk`
- **Mirror API** - La segunda API que poseen las Glass. Mientras que el GDK permite el desarrollo de Software para las Glass, la Mirror API permite el desarrollo de aplicativos web en Java, Python y PHP para que se comuniquen en remoto con las Glass y le envíen notificaciones.
  `https://developers.google.com/glass/v1/reference/`
- **GDK Quick Start** - Tutorial de configuración inicial para las Google Glass. No tan detallado como el Toptal pero el código sí que se basa en la última versión.
  `https://developers.google.com/glass/develop/gdk/quick-start`
- **ZXing Wiki** - Wiki interna de la librería ZXing para el reconocimiento de QRs y códigos de barras. Incluye ejemplos genéricos para Android.
  `https://github.com/zxing/zxing/wiki`
- **Glassware Collection** - Marketplace oficial de Google en el que se encuentran numerosas aplicaciones en su versión para Google Glass
  `https://glass.google.com/glassware`
- **Glass Support** - Soporte de las Google Glass con FAQs, tutoriales y videos.
  `http://support.google.com/glass`
- **Glass Etiquette** - Patrones de diseño recomendados por Google para el desarrollo de aplicaciones para Google Glass, qué se debe hacer y qué no.
  `sites.google.com/site/glasscomms/glass-explorers`

Referencias

[1] GitHub. *BarcodeEye Source Code*. 2014. URL: http://github.com/barcodeeye/barcodeeye (visitado 09-06-2015).

[2] GitHub. *QRLens Source Code*. 2014. URL: http://github.com/jaxbot/glass-qrlens (visitado 05-06-2015).

[3] GitHub. *ZXing Source Code*. 2014. URL: http://github.com/zxing/zxing (visitado 08-06-2015).

[4] wwwhatsnew. *15 aplicaciones de iPad para enseñar a niños con discapacidad*. 2012. URL: http://wwwhatsnew.com/2012/10/09/aplicaciones-ipad-ensenar-a-ninos-con-discapacidad (visitado 03-06-2015).