

PMDM06.- Multimedia. Audio y Vídeo.

Caso práctico

- Equipo, tenemos una petición de un cliente -anuncia Ada-. Tenemos que realizar una audioguía.

Ada les informa de que uno de los clientes de **BK Programación** ha solicitado presupuesto para desarrollar una app cuya funcionalidad principal es ser una audioguía para uno de los principales monumentos de la ciudad, muy visitado tras haber sido restaurado. La ventaja de construir esta app es que no es necesario comprar tantos dispositivos como visitantes concurrentes existan, sino que es posible contar con un parque de dispositivos, y para los días de mayor afluencia de público, los usuarios que contraten este servicio simplemente pueden descargar la app en sus propios teléfonos móviles para acceder a la información sobre el monumento.

- De modo que el parque de dispositivos tendrá guardado en local los medios a reproducir -apunta Juan, muy atento a la explicación del proyecto que les ha realizado Ada-, sin embargo, será necesario disponer de los medios a través de servicios de streaming y reproducción online para los usuarios que contraten el servicio.

- ¡Bien visto! -exclama María-. De otro modo ocuparían mucho espacio en los terminales de los clientes.

- ¿Nuevos controles a la vista? -pregunta Pedro.

- Para este proyecto es necesario integrar este tipo de medios en las aplicaciones Android Studio -explica Ada-, incorporando **Audio**, **Vídeo** o usando controles de tipo **WebView** donde incluso podemos seleccionar si habilitamos o no la capacidad de ejecutar código **JavaScript**.



[Rahul Pandit \(Pexels\)](#)



[Ministerio de Educación y Formación Profesional.](#) (Dominio público)

Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.

[Aviso Legal](#) 

1.- Audio.

Caso práctico



[Mikhail \(Pexels\)](#)

- El primer elemento multimedia a incorporar serán los archivos de Audio -explica Ada-, que podrán ser reproducidos a través de la clase Media Player.
- ¿Qué formatos de audio es capaz de reproducir Android? -pregunta María.
- Android es capaz de reproducir distintos formatos -responde Ada-, pero uno de los más extendidos es el **mp3**.

El equipo de trabajo, gracias a algunos programas de **SW libre** y gratuitos como **Audacity**, puede generar los archivos que van a reproducirse en la app.

Para trabajar con la reproducción de sonido, existen en Android varias alternativas. Una de las más sencillas es trabajar con la clase **MediaPlayer**. Dicha clase permitirá crear un objeto **MediaPlayer** mediante el método **create()**, el cual recibirá como parámetro el propio sonido y que podrá encontrarse en la carpeta de recursos especial **raw**. Posteriormente reproduciremos dicho sonido con el método **start()**.

La clase **MediaPlayer** también se usará en la reproducción de vídeo.

Por otro lado, al igual que sucederá con los recursos de vídeo, es importante guardar los ficheros de sonido en una carpeta específica llamada **raw** dentro de la carpeta **res**.

1.1.- Ejercicio resuelto 1.

Ejercicio Resuelto

Vamos a crear una pequeña aplicación que permita reproducir el audio de la locución de unos números. La apariencia será la de la figura 1.

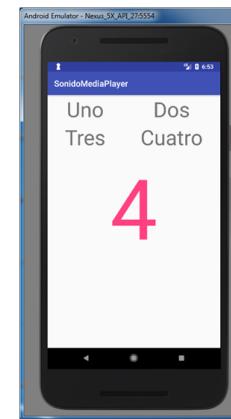
Al clicar en el texto de cualquiera de los números, aparecerá el texto en formato numérico y además se escuchará una locución relativa a ese número. Ver figura 2.

Figura 1



[Google Developers \(Uso educativo nc\)](#)

Figura 2



[Google Developers \(Uso educativo nc\)](#)

[Mostrar retroalimentación](#)

Lo primero será crear una carpeta de recursos (dentro de **res**) que llamaremos **raw** y que tendrá los ficheros de audio (en nuestro caso los meteremos en formato .mp3, pero reconoce otros formatos).

Una vez creada la carpeta, metemos los ficheros:

Para hacer el ejercicio el alumno debe tratar de crear dichos ficheros por cuenta propia. Se puede grabar un fichero con todos los números, y luego trocearlo con programas de tratamiento de sonido como Audacity.

Y codificamos lo siguiente:

activity_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="vertical"
8     tools:context="com.example.dell.sonidomediaplayer.MainActivity">
9
10    <TableLayout
11        android:layout_width="match_parent"
12        android:layout_height="wrap_content">
13        <TableRow>
14            <TextView
15                android:id="@+id/tv1"
16                android:layout_height="wrap_content"
17                android:layout_weight="1"
18                android:textSize="50dp"
19                android:text="Uno"
20                android:gravity="center"/>
21            <TextView />
```

```
22
23     <TextView
24         android:id="@+id/tv2"
25         android:layout_height="wrap_content"
26         android:layout_weight="1"
27         android:textSize="50dp"
28         android:text="Dos"
29         android:gravity="center"/>
30     <TextView />
31 </TableRow>
32 <TableRow>
33     <TextView
34         android:id="@+id/tv3"
35         android:layout_height="wrap_content"
36         android:layout_weight="1"
37         android:textSize="50dp"
38         android:text="Tres"
39         android:gravity="center"/>
40     <TextView />
41     <TextView
42         android:id="@+id/tv4"
43         android:layout_height="wrap_content"
44         android:layout_weight="1"
45         android:textSize="50dp"
46         android:text="Cuatro"
47         android:gravity="center"/>
48     <TextView />
49 </TableRow>
50 </TableLayout>
51 <TextView
52     android:id="@+id/tvNumero"
53     android:layout_width="wrap_content"
54     android:layout_height="wrap_content"
55     android:textSize="200dp"
56     android:layout_gravity="center"
57     android:textColor="@color/colorAccent"/>
58 </LinearLayout>
```

MainActivity.java

```
1 package com.example.dell.sonidomediaplayer;
2
3 import android.media.MediaPlayer;
4 import androidx.appcompat.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.TextView;
8
9 public class MainActivity extends AppCompatActivity implements View.OnClickListener {
10
11     TextView tv1, tv2, tv3, tv4, tvNumero;
12     MediaPlayer mp;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         tv1=(TextView)findViewById(R.id.tv1);
19         tv2=(TextView)findViewById(R.id.tv2);
20         tv3=(TextView)findViewById(R.id.tv3);
21         tv4=(TextView)findViewById(R.id.tv4);
22         tvNumero=(TextView)findViewById(R.id.tvNumero);
23         tv1.setOnClickListener(this);
24         tv2.setOnClickListener(this);
25         tv3.setOnClickListener(this);
26         tv4.setOnClickListener(this);
27     }
28
29     @Override
30
31     public void onClick(View v) {
32         switch(v.getId()){
33             case(R.id.tv1):
```

```
34         tvNumero.setText("1");
35         mp=MediaPlayer.create(this, R.raw.uno);
36         break;
37     case(R.id.tv2):
38         tvNumero.setText("2");
39         mp=MediaPlayer.create(this, R.raw.dos);
40         break;
41     case(R.id.tv3):
42         tvNumero.setText("3");
43         mp=MediaPlayer.create(this, R.raw.tres);
44         break;
45     case(R.id.tv4):
46         tvNumero.setText("4");
47         mp=MediaPlayer.create(this, R.raw.cuatro);
48         break;
49     }
50     mp.start();
51 }
52 }
```

2.- Vídeo.

Caso práctico



Liza Summer ([Pexels](#))

- El siguiente elemento a incorporar en nuestra app es el vídeo -informa Ada a su equipo, que va cogiendo confianza con el paso de los días y la ejecución de sus proyectos-. Este medio puede generarse directamente grabando desde los dispositivos móviles actuales, aunque es imprescindible llevar a cabo la renderización y el procesamiento de los mismos para bajar el peso de los ficheros generados.
- ¿Tenemos también programas que nos ayuden a generar los vídeos? -pregunta Pedro.
- Por supuesto -responde Ada-. Existen distintos programas de SW libre que permiten llevar a cabo estas tareas como **Handbrake**.
- Otras alternativas son conseguir vídeos de bancos de imágenes y vídeos públicos -prosigue Ada en su respuesta-. En este caso es fundamental entender los términos en los que el autor comparte su obra y cede los derechos de su uso. Es importante cumplir con ellos para evitar problemas legales y para garantizar que los derechos de los autores se respetan, lo que garantizará el que estos se sigan generando en unos parámetros de calidad adecuados.

La clase más habitual para reproducir vídeo es la clase **VideoView**, y será ésta sobre la que nos centraremos. Además, la acompañaremos de la clase **MediaController** para proporcionar un reproductor o controlador de reproducción con controles de reproducción, pausa, avance y retroceso, con una apariencia sencilla pero correcta.

La clase **VideoView** muestra un archivo de vídeo. La clase VideoView puede cargar imágenes de varias fuentes (como recursos o proveedores de contenido), se ocupa de calcular su medición a partir del vídeo para que se pueda usar en cualquier administrador de diseño y proporciona varias opciones de visualización, como escalado y tintado.

VideoView no conserva su estado completo al pasar al segundo plano o al cambiar la orientación del dispositivo. En particular, no restaura el estado de reproducción actual, posición de reproducción, pistas seleccionadas o cualquier pista de subtítulos agregada a través de **addSubtitleSource()**. Las aplicaciones deben guardarlas y restaurarlas por sí mismas en

Activity.onSaveInstanceState(Bundle) y **Activity.onRestoreInstanceState(Bundle)**. En el último ejercicio resuelto del capítulo trabajaremos sobre este importante aspecto.

Existen otras clases y librerías para reproducción de vídeo, como **ExoPlayer**, pero se puede considerar **VideoView** como el referente principal.

En los ejercicios resueltos que proporcionaremos a continuación, los vídeos a reproducir unas veces pertenecerán a la aplicación (incorporados en carpeta **res/raw**, y otras veces serán vídeos de internet (servidor externo) o incluso, podrán ser vídeos que tengamos incorporados en un servidor web local (por ejemplo con el Apache que proporciona XAMPP).



[Ministerio de Educación y
Formación Profesional \(Uso
Educativo nc\)](#)

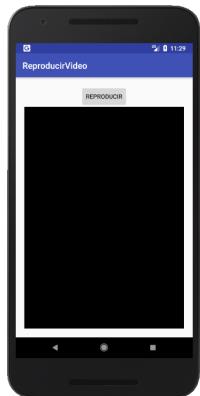
2.1.- Ejercicio Resuelto 2.

Ejercicio Resuelto

Crear una aplicación que disponga de un botón y un **VideoView** de forma que al pulsar al botón reproduzca, de manera aleatoria, dos posibles vídeos que previamente habremos incorporado en la aplicación (ver figura 1).

Los posibles vídeos serán los mostrados en la figura 2 y figura 3.

Figura 1



[Google Developers](#) (Uso educativo
nc)

Figura 2



[Google Developers](#) (Uso educativo
nc)

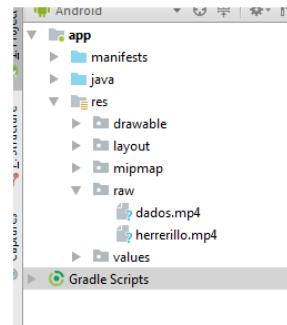
Figura 3



[Google Developers](#) (Uso educativo
nc)

Los vídeos están sacados de Pixabay.com. Hay que tratar que no sean muy pesados para no sobrecargar la aplicación. Los alumnos podrán seleccionar los que quieran para practicar con este ejercicio.

[Mostrar retroalimentación](#)



[Google Developers \(Uso educativo nc\)](#)

En primer lugar habrá que incorporar los vídeos en la carpeta raw de la carpeta de recursos. La carpeta raw por defecto no está creada, pero podemos hacerlo mediante clic derecho sobre la carpeta res > New > Android Resource Directory. En el desplegable "Resource type" elegimos "raw", y ya le pondrá automáticamente también "raw" como "Directory name". Tras esto, copiamos los ficheros que en nuestro caso tendrán extensión .mp4. Quedará como se muestra en la imagen.

activity_main.xml

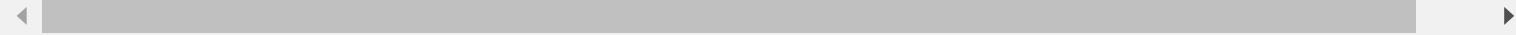
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      android:padding="20dp"
9      tools:context="com.example.dell.reproducirvídeo.MainActivity">
10
11     <Button
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:id="@+id/btnReproducir"
15         android:text="Reproducir"
16         android:layout_gravity="center"/>
17
18     <!-- Incorporamos etiqueta de View VideoView para cargar sobre ella el vídeo -->
19     <VideoView
          android:layout_width="match_parent"
```

```
20     android:layout_height="wrap_content"
21     android:id="@+id/vwReproductor"
22     android:layout_gravity="center"/>
23
24 </LinearLayout>
25
```

MainActivity.java

```
1 import android.net.Uri;
2 import androidx.appcompat.app.AppCompatActivity;
3 import android.import android.os.Bundle;
4 import android.view.View;
5 import android.widget.Button;
6 import android.widget.VideoView;
7
8 public class MainActivity extends AppCompatActivity {
9     Button btnReproducir;
10    VideoView vwReproductor;
11
12    @Override
13
14    protected void onCreate(Bundle savedInstanceState) {
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.activity_main);
17        btnReproducir=(Button)findViewById(R.id.btnReproducir);
18        vwReproductor=(VideoView)findViewById(R.id.vwReproductor);
19        btnReproducir.setOnClickListener(new View.OnClickListener() {
20
21            @Override
22
23            public void onClick(View v) {
24                int aleatorio=(int)Math.floor(Math.random()*2);
```

```
25     int identificadorRecurso;
26     if(aleatorio==0)
27         identificadorRecurso = R.raw.dados;
28     else
29         identificadorRecurso = R.raw.herrerillo;
30     String rutaVideo = "android.resource://com.example.dell.reproducirvídeo/" + identificadorRecurso;
31     //En la línea anterior, en lugar de poner com.example.dell.reproducirvídeo", podemos concatenar
32     // con getPackageName(), la cual devuelve un String con el nombre del paquete del proyecto
33     Uri uri = Uri.parse(rutaVideo);
34     vwReproductor.setVideoURI(uri);
35     vwReproductor.start();
36 }
37 });
38 }
39 }
```



2.2.- Ejercicio Resuelto 3.

Ejercicio Resuelto

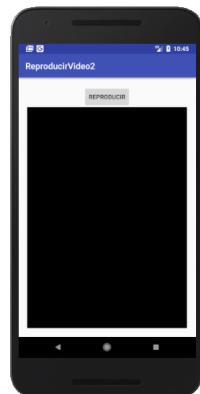
Incorporar en el ejercicio anterior la clase **MediaController**, la cual dotará al reproductor de un aspecto más completo, incorporando implícitamente controles de reproducir, parar, ir hacia delante e ir hacia atrás, e incluso arrastrar el cursor de la línea de reproducción para avanzar o adelantar todo lo que queramos la reproducción.

La actividad inicialmente tendrá el mismo aspecto (figura 1).

Además al clicar sobre el botón también se reproducirá aleatoriamente cualquiera de los videos (figura 2)

La diferencia es que si clicamos sobre la imagen del video, ya me aparecen unos controles de reproducción proporcionados por la clase MediaController (figura 3).

Figura 1. Aspecto inicial



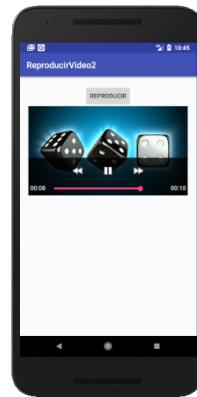
[Google Developers](#) (Uso educativo
nc)

Figura 2. Al clicar sobre el botón



[Google Developers](#) (Uso educativo
nc)

Figura 3. Al clicar sobre el vídeo



[Google Developers](#) (Uso educativo
nc)

Mostrar retroalimentación

Los recursos serán los mismos. Además el fichero **activity_main.xml** será exactamente igual. Los cambios se harán solo en **MainActivity.java**.

MainActivity.java

```
1 package com.example.dell.reproducirvideo2;
2 import android.net.Uri;
3 import androidx.appcompat.app.AppCompatActivity;
4 import android.os.Bundle;
5 import android.view.View;
6 import android.widget.Button;
7 import android.widget.MediaController;
8 import android.widget.VideoView;
9
10 //En este ejercicio, con respecto al anterior, incorporaremos la clase MediaController la cual dotará al repr
11 //de un aspecto más completo, incorporando implicitamente controles de reproducir, parar, ir hacia delante e
12 //hacia atrás, e incluso podemos arrastrar el cursor de la linea de reproducción para avanzar o adelantar tod
13 //que queramos la reproducción
14 public class MainActivity extends AppCompatActivity {
15     Button btnReproducir;
16     VideoView vwReproductor;
17     MediaController mediaController;
18
19     @Override
20
21     protected void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_main);
24         btnReproducir=(Button)findViewById(R.id.btnReproducir);
25         vwReproductor=(VideoView)findViewById(R.id.vwReproductor);
26         mediaController=new MediaController(this); //No poner getApplicationContext, ya que fallará
27         btnReproducir.setOnClickListener(new View.OnClickListener() {
28
29             @Override
```

```
29
30     public void onClick(View v) {
31         int aleatorio=(int)Math.floor(Math.random()*2);
32         int identificadorRecurso;
33         if(aleatorio==0)
34             identificadorRecurso = R.raw.dados;
35         else
36             identificadorRecurso = R.raw.herrerillo;
37         String rutaVideo = "android.resource://com.example.dell.reproducirvideo/" + identificadorRecu
38         Uri uri = Uri.parse(rutaVideo);
39         vwReproductor.setVideoURI(uri);
40         vwReproductor.setMediaController(mediaController);
41         mediaController.setAnchorView(vwReproductor);
42         vwReproductor.start();
43     }
44 });
45 }
46 }
47 }
```

2.3.- Ejercicio Resuelto 4.

Ejercicio Resuelto

En este ejercicio, con respecto a los anteriores, vamos a proveer a nuestra aplicación de la capacidad de ser capaz de restaurar el estado de reproducción actual y posición, de forma que si cambiamos la orientación del dispositivo, en lugar de situarse al principio de la reproducción, sigue por donde estaba.

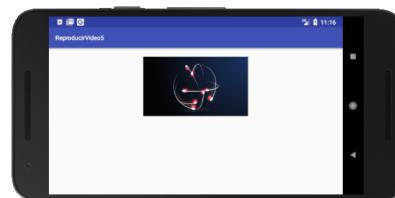
Esto viene a propósito de la nota mencionada anteriormente, en el capítulo explicativo sobre **VideoView**:

VideoView no conserva su estado completo al pasar al segundo plano o al cambiar la orientación del dispositivo. En particular, no restaura el estado de reproducción actual, posición de reproducción, pistas seleccionadas o cualquier pista de subtítulos agregada a través de **addSubtitleSource()**. Las aplicaciones deben guardarlas y restaurarlas por sí mismas en **Activity.onSaveInstanceState(Bundle)** y **Activity.onRestoreInstanceState(Bundle)**.

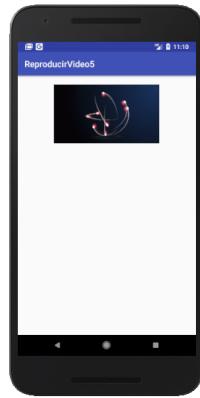
Dichos métodos son muy importantes porque harán que si giramos de orientación el dispositivo, la reproducción continuará por el punto donde estaban. Si no lo ponemos comenzará a reproducir desde el principio (se aconseja hacer la prueba).

Al reproducir el vídeo.

Al girar la pantalla, se reanudará en el mismo punto en el que estaba cuando se giró.

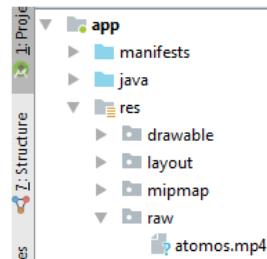


[Google Developers](#) (Uso educativo nc)



[Google Developers](#) (Uso educativo nc)

Mostrar retroalimentación



Incorporaremos el vídeo (puedes elegir uno que tenga un peso razonable de algún banco de recursos) en la carpeta **raw**.

[Google Developers](#) (Uso educativo nc)

activity_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
4  xmlns:tools="http://schemas.android.com/tools"
5  android:layout_width="match_parent"
6  android:layout_height="match_parent"
7  android:orientation="vertical"
8  tools:context="com.example.dell.reproducirvídeo5.MainActivity">
9
10 <VideoView
11     android:id="@+id/vV"
12     android:layout_width="wrap_content"
13     android:layout_height="wrap_content"
14     android:layout_margin="20dp"
15     android:layout_gravity="center" />
16
17 </LinearLayout>
```

MainActivity.java

```
1 package com.example.dell.reproducirvídeo5;
2 import android.media.MediaPlayer;
3 import android.net.Uri;
4 import androidx.appcompat.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.util.Log;
7 import android.widget.MediaController;
8 import android.widget.VideoView;
9 /* En este ejercicio, con respecto a los anteriores, vamos a proveer a nuestra aplicación de la capacidad
10 de ser capaz de restaurar el estado de reproducción actual y posición, de forma que si cambiamos la orientación
11 del dispositivo, en lugar de situarse al principio de la reproducción, sigue por donde estaba.
12 */
13
14 public class MainActivity extends AppCompatActivity {
15     private VideoView vV;
```

```
16     private int posicion = 0;
17
18     @Override
19
20     protected void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.activity_main);
23         vV = (VideoView) findViewById(R.id.vV);
24         establecerVideo();
25     }
26
27     private void establecerVideo() {
28         // Especificamos la ruta al vídeo que reproduciremos
29         //En mi caso getPackageName() devolverá la cadena "com.example.dell.reproducirvídeo5"
30         String ruta = "android.resource://" + getPackageName() + "/" + R.raw.atomos;
31         Uri uri = Uri.parse(ruta);
32         // Para darle mayor funcionalidad, incorporaremos controles multimedia.
33         MediaController mediaController = new MediaController(this);
34         // Establecemos los controles multimedia a mi VideoView
35         vV.setMediaController(mediaController);
36         try {
37             // Establecemos la URI del vídeo a mi VideoView
38             vV.setVideoURI(uri);
39             // Asignamos el foco a mi VideoView
40             vV.requestFocus();
41         } catch (Exception e) {
42             Log.e("Error", e.getMessage());
43             e.printStackTrace();
44         }
45
46         // Establecemos un escuchador de eventos o listener que controle si el vídeo está ya preparado para s
47         vV.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
48
49             @Override
50
51             public void onPrepared(MediaPlayer mp) {
52                 //Hacemos que el vídeo se reproduzca continuamente
53                 mp.setLooping(true);
54                 if (posicion == 0) { //Si estamos en el principio, el vídeo comenzará la reproducción
```

```
55         vV.start();
56     } else { //Si no está en principio, entonces lo pausaremos
57         vV.pause();
58     }
59 }
60 });
61 }
62 /* Ahora incorporaremos la sobreescritura de los métodos onSaveInstanceState() y onRestoreInstanceState().
63 Dichos métodos son muy importantes porque harán que si giramos de orientación el dispositivo, la repro
64 continuará por el punto donde estaban.
65 Si no lo ponemos comenzará a reproducir desde el principio (se aconseja hacer la prueba)
66 */
67
68 @Override
69
70 public void onSaveInstanceState(Bundle savedInstanceState) {
71     super.onSaveInstanceState(savedInstanceState);
72     // Con onSaveInstanceState almacenamos la posición por donde iba cuando se cambió la orientación del
73     savedInstanceState.putInt("miPosicion", vV.getCurrentPosition());
74     vV.pause();
75 }
76
77 @Override
78
79 public void onRestoreInstanceState(Bundle savedInstanceState) {
80     super.onRestoreInstanceState(savedInstanceState);
81     // Con onRestoreInstanceState recuperamos la posición por donde iba cuando se cambió la orientación d
82     posicion = savedInstanceState.getInt("miPosicion");
83     vV.seekTo(posicion);
84 }
85 }
```



3.- WebView.

Caso práctico

- Acabo de hablar con nuestro cliente -informa Ada al equipo-. La empresa que ha solicitado la aplicación para las visitas guiadas, ha pedido que algunos de los vídeos no se reproduzcan directamente desde la app sino que la reproducción se haga desde una plataforma externa donde están actualmente alojados.

- ¿Eso nos perjudica? -pregunta Juan.

- ¡Todo lo contrario! -exclama Ada, quien les explica los beneficios de la petición del cliente.

La reproducción de los vídeos desde una plataforma externa otorga varias ventajas:



Mikhail ([Pexels](#))

- ✓ Es posible actualizar los contenidos en la plataforma externa manteniendo los enlaces, y por tanto sin necesidad de actualizar la app.
- ✓ No es necesario guardar los medios en los dispositivos locales donde la app se está ejecutando.
- ✓ Por otro lado, permite contabilizar todas las reproducciones y aumentar los ratios en términos de visitas.

- Para conseguirlo -aclara Ada al equipo tras indicarles las ventajas de la petición del cliente-, vamos a implementar la clase **WebView** en la aplicación.

La clase **WebView** es una extensión de la clase **View** de Android que nos permitirá mostrar páginas web como parte del diseño de nuestra actividad, de manera perfectamente integrada. No incluye ninguna característica de un navegador web completamente desarrollado, como controles de navegación o una barra de direcciones. Todo lo que hace **WebView**, de forma predeterminada, es mostrar una página web.

Un escenario común en el que usar **WebView** es útil es cuando desea proporcionar información en su aplicación que podría necesitar actualizar, como un acuerdo de usuario final o una guía del usuario. Dentro de su aplicación Android, puede crear una Actividad que contenga una **WebView** y luego usarla para mostrar su documento alojado en línea.

Otro escenario en el que **WebView** puede ayudar es si nuestra aplicación proporciona datos a un usuario que requiera una conexión a Internet para recuperar datos, como el correo electrónico. En este caso, puede ser más fácil construir un **WebView** en la aplicación Android que muestra una página web con todos los datos del usuario, en lugar de realizar una solicitud de red, luego analizar los datos y representarlos en un diseño de Android. En su lugar, podemos diseñar una página web que se adapte a los dispositivos Android y luego implementar un **WebView** en nuestra aplicación que cargue la página web.

3.1.- Añadiendo un **WebView** a nuestra aplicación.

Lo primero será añadir una etiqueta <**WebView**> en el layout donde queramos incorporar el **WebView**.

Código

```
1 | <?xml version="1.0" encoding="utf-8"?>
2 | <WebView  xmlns:android="http://schemas.android.com/apk/res/android"
3 |   android:id="@+id/webview"
4 |   android:layout_width="fill_parent"
5 |   android:layout_height="fill_parent"
6 | />
```

A continuación, para cargar la página en el **WebView**, utilizaremos el método **loadUrl()**.

Código

```
1 | WebView myWebView = (WebView) findViewById(R.id.webview);
2 | myWebView.loadUrl("http://www.example.com");
```

Además, previamente, para que la conexión se realice será necesario incorporar en el fichero **AndroidManifest.xml** el permiso de acceso a Internet (**android.permission.INTERNET**):

```
1 | <manifest ... >
2 |   <uses-permission android:name="android.permission.INTERNET" />
3 |   ...
4 | </manifest>
```

Igualmente, en caso de que vayamos a acceder a contenido no seguro (no https) es necesario añadir también este atributo a la etiqueta application en el archivo de manifiesto:

```
1 | android:usesCleartextTraffic="true"
```

3.2.- Habilitar JavaScript.

Por defecto la carga del JavaScript asociado a una página web está deshabilitado en un **WebView**. Para habilitarlo podremos hacerlo a través de la clase **WebSettings** mediante el método **setJavaScriptEnabled()**.

Código

```
1 | WebView myWebView = (WebView) findViewById(R.id.webview);
2 | WebSettings webSettings = myWebView.getSettings();
3 | webSettings.setJavaScriptEnabled(true);
```

Nota: En las versiones más modernas de API, además tenemos que incorporar las líneas:

```
1 | myWebView.setWebViewClient(new WebViewClient());
2 | myWebView.setWebChromeClient(new WebChromeClient());
```