Tema 4: Utilización de librerías multimedia

¿Qué aprenderás?

Conocer las posibilidades multimedia que ofrecen los dispositivos Android.

- Construir una aplicación multimedia capaz de reproducir archivos de audio y vídeo..

¿Sabías que…?

- La clase MediaPlayer permite reproducir contenidos multimedia, desde archivos MP3 hasta vídeos.
- Es posible reproducir contenidos multimedia en streaming, es decir, directamente desde Internet sin necesidad de descargar previamente ningún archivo.

4.1. Reproducción de objetos multimedia

El término multimedia tiene sus orígenes en los años 90 y se refiere a varias áreas que mezclan la informática y las telecomunicaciones con el objetivo de utilizar distintos medios y tecnologías para guardar, mostrar, enviar y recibir información. Los medios pueden ser variados, desde texto, gráficos e imágenes, hasta sonidos, vídeo o animaciones.

Las librerías de Android permiten capturar y reproducir imágenes, sonidos y vídeo en diferentes formatos. Además, las fuentes desde donde se pueden reproducir los archivos multimedia abarcan el sistema de archivos del dispositivo móvil pero también la reproducción en streaming de contenidos que se encuentran en Internet.

Una aplicación Android puede integrar con facilidad audio, vídeo e imágenes. La clase principal que se utiliza para procesar audio y vídeo es MediaPlayer.

Antes de utilizar la clase MediaPlayer debe incluirse en el fichero *manifest* la declaración de los permisos de uso. El primero de los permisos está relacionado con el uso de conexiones de datos, en el caso de que el contenido al que se va a acceder se encuentre en Internet. Como ya se vio en el capítulo anterior (punto 3.2 “Comunicaciones en Android”), la declaración para establecer el permiso de uso es la siguiente:

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

En el caso de que la aplicación requiera desbloquear el dispositivo, ya sea porque se encuentro en modo sleeping (en reposo) o bien bloqueado por el usuario, el permiso para desbloquear el dispositivo es el siguiente:

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

4.2. La clase MediaPlayer

La clase MediaPlayer se encuentra en el paqueta android.media y permite procesar tanto audio como vídeo desde diferentes fuentes. Las fuentes posibles son las siguientes:

- Desde un archivo en memoria interna (local): en este caso el contenido está ubicado en un archivo en la carpeta */res/raw* o bien en la carpeta */assets*. Este fichero se empaquetará junto al resto de la aplicación en un archivo apk.
- Desde un archivo en memoria externa: el contenido está ubicado generalmente en una tarjeta de memoria SD.
- Desde la web: al contenido se accede mediante una URL (dirección de Internet).

Es importante destacar que cada fabricante organiza el almacenamiento del dispositivo de la manera que considera más oportuna. Así, es habitual que un dispositivo haga referencia a su memoria interna cuando en realidad se trata de una ampliación de memoria externa. Por este motivo, la búsqueda de contenidos en memoria interna o externa puede resultar confusa o incluso infructuosa, dependiendo del dispositivo en cuestión.

El ciclo de vida un objeto de la clase MediaPlayer es el conjunto de estados y transiciones entre estos estados que atraviesa dependiendo de las acciones de los usuarios. Por lo general, a partir del estado *idle*, es decir, en reposo:

1. Se ejecutará en primer lugar el método *setDataSource()* que establece la fuente multimedia a reproducir.
2. A continuación, se ejecuta el método *prepare()* que deja el objeto en estado *Prepared*, a punto para iniciar la reproducción de audio o vídeo.
3. La reproducción de audio o vídeo comienza cuando se ejecuta el método *start()*; es posible ejecutar el método *pause()* para pausar la reproducción del contenido multimedia o bien el método *stop()* para detenerla.
4. Finalmente, el método *release()* permite liberar y finalizar el ciclo de vida del objeto MediaPlayer.

El código siguiente nos muestra cómo realizar la declaración de un objeto de la clase MediaPlayer e inicializarlo, mediante los métodos *setDataSource()* y *prepare()*:

```
MediaPlayer mp = new MediaPlayer();
try {
    mp.setDataSource(this, Uri.parse("android.resource://" +
                            getPackageName()+ "/" + R.raw.archivo));
    mp.prepare();
} catch (Exception e) {
}
```

Otra posibilidad es utilizar el método *create()* que de manera más sencilla aúna las llamadas a los métodos *setDataSource()* y *prepare()* tal y como se muestra a continuación:

```
MediaPlayer mp = MediaPlayer.create(this, R.raw.archivo);
```

En ambos casos, se está asignando el objeto mp de la clase MediaPlayer al archivo multimedia ***archivo*** que se encuentra ubicado en la carpeta */res/raw* del proyecto.En los siguientes puntos, veremos la utilización de este objeto para reproducir contenido multimedia.

4.3. Diagrama de estados de MediaPlayer

El control de reproducción de los archivos de audio y vídeo se realiza a través de una serie de estados. En este diagrama, los óvalos representan los diferentes estados y las flechas las operaciones de reproducción (inicio, pausa, avance, detención, …).

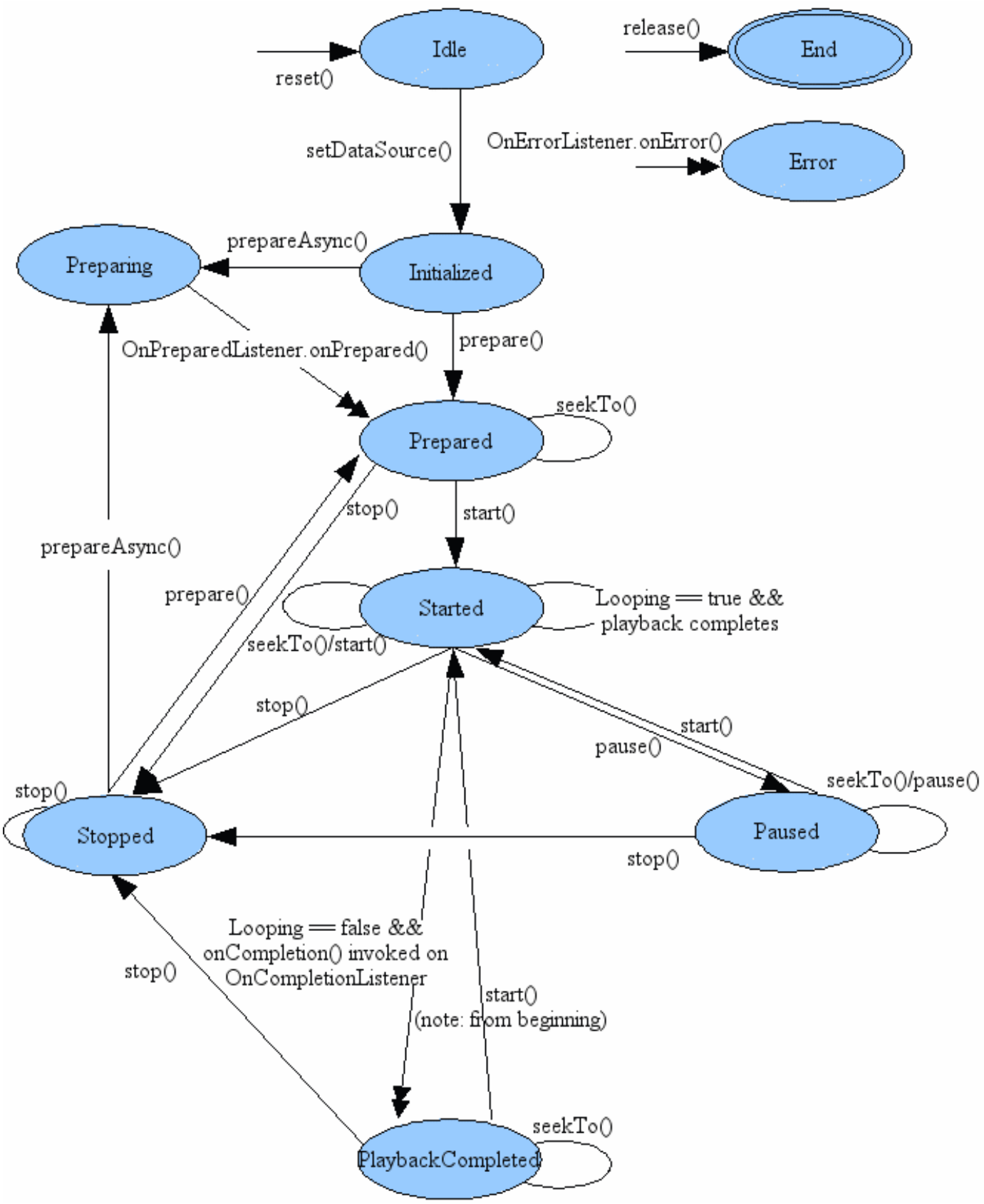

Figura: Diagrama de estados de la clase MediaPlayer

4.4. Métodos de la clase MediaPlayer

Vamos a ver un pequeño ejemplo de código que utiliza la clase AsyncTask para conectarse a un servidor remoto.

```
public class MainActivity extends AppCompatActivity {
    MediaPlayer mp = new MediaPlayer();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mp = MediaPlayer.create(this, R.raw.avicii);
        mp.start(); // inicia reproducción de audio

    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        mp.stop(); // detiene la reproducción de audio
    }
}
```

En este ejemplo, se utiliza el método *start()* para iniciar la reproducción del contenido multimedia; en este caso, un archivo de audio llamado *avicii.mp3* y ubicado en */res/raw/avicii.mp3*.

La reproducción de sonido se inicia en el momento en el que se inicia la app, en el método *onCreate()*. El método *stop()* se llama cuando se finaliza la ejecución de la app: en el método *onDestroy()*.

En este ejemplo, la reproducción del archivo de audio se detiene y la ejecución de la app finaliza; en caso contrario, debería llamarse al método *prepare()* antes de volver a llamar al método *start()*. Esto es debido a que tras la llamada al método *stop()* el objeto MediaPlayer queda en estado **Stopped** y debe llamarse al método *prepare()* para que pase al estado **Prepared**.

4.5. Reproducción de vídeo

De manera análoga a como se realiza la reproducción de archivos de audio, la API de Android proporciona el componente VideoView y la clase MediaController, que combinados permiten visualizar archivos de vídeo desde diversas fuentes.

MediaController proporciona los botones de control de la reproducción (inicio, pausa, retroceso, avance y parada). Mediante el método *setMediaController()* se asocian los controles de reproducción a un componente VideoView que muestra el vídeo. El siguiente código permite reproducir un vídeo desde una fuente online utilizando esta técnica.

```
public class MainActivity extends AppCompatActivity {
    private VideoView vidView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        vidView = (VideoView)findViewById(R.id.miVideo);
        String vidAddress = "http://techslides.com/demos/sample-
videos/small.mp4";
        Uri vidUri = Uri.parse(vidAddress);
        vidView.setVideoURI(vidUri);

        // botones de control de reproducción
        MediaController vidControl = new MediaController(this);
        vidControl.setAnchorView(vidView);
        vidView.setMediaController(vidControl);

        // inicio del vídeo
        vidView.start();
    }
}
```

4.6. Formatos de audio y vídeo

El formato de los archivos de audio y vídeo establece la manera en que la información está organizada y codificada. Independientemente del origen de estos archivos, su formato debe ser compatible para que el contenido pueda ser reproducido.

Por un lado, los clips de audio son archivos pensados para ser utilizados por un reproductor de sonido. Existen varios formatos:

1. Formatos de audio sin compresión: la información se guarda tal y como se graba; entre los formatos más populares se encuentran WAV, AIFF y AU.
2. Formatos de audio sin pérdida: la información de audio se comprime con algoritmos y se puede recuperar sin pérdida de partes del mismo. El formato más popular es FLAC.
3. Formatos de audio con pérdida: la información se comprime y se codifica, perdiendo parte de la información original. El formato más popular es MP3.

En la actualidad están disponibles diversidad de formatos de vídeo. La diferencia entre los distintos formatos está principalmente en la calidad, que por lo general está adaptada al medio de reproducción: no se requiere la misma calidad para visualizar un vídeo en una plataforma de vídeos online como Youtube o Vimeo, que para visualizar el mismo vídeo en una pantalla digital de alta resolución.

En la compresión de los archivos de vídeo hay que diferenciar entre el formato contenedor y el códec utilizado. El formato contenedor engloba el vídeo y el audio del mismo (puede contener varias pistas de audio en diversos idiomas) y los formatos más habituales son AVI, MPG, MOV, H264 y WMV. El códec de vídeo es el programa responsable de realizar la compresión y descomprensión digital y está directamente relacionado con la calidad del archivo comprimido. Entre los códecs más populares se encuentra H264, XVID y DIVX.

Además de audio y vídeo, los contenidos también incluyen gráficos, fuentes de texto, imágenes y animaciones, entendiendo por animación la presentación de varias imágenes por segundo generando al observador sensación de movimiento. Además de los formatos de imagen para impresión (alta calidad) como EPS, TIFF o PSD, los formatos de imagen para la web utilizan técnicas de compresión con pérdida para producir archivos de menor tamaño.

Entre los formatos de imagen para la web más populares encontramos los siguientes:

- Formato JPG, tiene una buena calidad de imagen y buena velocidad de descarga y visualización.
- Formato GIF: tiene baja calidad de imagen, pero alta velocidad de descarga.
- Formato PNG: tiene excelente calidad de imagen, pero baja calidad de descarga.
- Formato BMP: tiene escasa compresión y baja velocidad de descarga; su uso está extendido.

Recursos y enlaces

- Guías para desarrolladores de Android

- Reproducción de medios en Android

Conceptos clave

- **Formato**: Un formato es un estándar que define la forma en que la información está codificada. Existen diferentes formatos multimedia, tanto para audio como vídeo. Entre los formatos más populares se encuentran MP3, MPEG y MP4.
- **Multimedia:** La tecnología multimedia combina distintos medios como audio, vídeo y televisión. Un dispositivo multimedia es aquél capaz de integrar y controlar diversos contenidos a través de pantallas, lectores de discos, conexión de datos a Internet y sintetizadores de audio y vídeo.
- **Streaming:** Técnica de transmisión de contenidos multimedia que permite al usuario consumir los contenidos mientras se están descargando. Se utiliza el protocol RTP (Realtime Transport Protocol) para visualizar los contenidos.

Test de autoevaluación

1. ¿En qué estado debe encontrarse un objeto de la clase MediaPlayer para poder iniciar la reproducción de contenidos?
   a) Prepared.
   b) Started.
   c) Ready.
   d) Iddle

2. ¿Cuál de los siguientes métodos debemos ejecutar para devolver el objeto multimedia al estado Prepared?
   a) stop().
   b) prepareAsync().
   c) release().
   d) start().

3. ¿Qué fuentes puede utilizar un objeto de la clase MediaPlayer para reproducir contenidos multimedia?
   a) El sistema de archivos de memoria interna.
   b) El sistema de archivos de una memoria externa, por ejemplo, una tarjeta SD.
   c) Un servidor de datos accesible a través de Internet.
   d) Todas las respuestas anteriores son correctas.

4. ¿Qué clase proporciona una interfaz de usuario con botones de control de reproducción a un componente de tipo VideoView?
   a) MediaPlayer.
   b) MediaController.
   c) MediaButtons.
   d) Ninguna de las respuestas anteriores es correcta.

Ponlo en práctica

Actividad 1

Crea un nuevo proyecto en Android Studio que utilice la clase MediaPlayer y mediante tres botones permitan iniciar, pausar y detener la reproducción de un archivo mp3.

Solucionarios

Test de autoevaluación

1. ¿En qué estado debe encontrarse un objeto de la clase MediaPlayer para poder iniciar la reproducción de contenidos?
   a) **Prepared.**
   b) Started.
   c) Ready.
   d) Iddle

2. ¿Cuál de los siguientes métodos debemos ejecutar para devolver el objeto multimedia al estado Prepared?
   a) stop().
   b) **prepareAsync().**
   c) release().
   d) start().

3. ¿Qué fuentes puede utilizar un objeto de la clase MediaPlayer para reproducir contenidos multimedia?
   a) El sistema de archivos de memoria interna.
   b) El sistema de archivos de una memoria externa, por ejemplo, una tarjeta SD.
   c) Un servidor de datos accesible a través de Internet.
   d) **Todas las respuestas anteriores son correctas.**

4. ¿Qué clase proporciona una interfaz de usuario con botones de control de reproducción a un componente de tipo VideoView?
   a) MediaPlayer.
   b) **MediaController.**
   c) MediaButtons.
   d) Ninguna de las respuestas anteriores es correcta.

Ponlo en práctica

Actividad 1

Crea un nuevo proyecto en Android Studio que utilice la clase MediaPlayer y mediante tres botones permitan iniciar, pausar y detener la reproducción de un archivo mp3.

Solución:

La aplicación utiliza tres botones. El botonPlay llama al método start() para iniciar la reproducción del archivo de sonido; en este caso, el fichero avicii.mp3 ubicado en la carpeta el proyecto /res/raw/avicii.mp3.

El botonPause simplemente pausa la reproducción de sonido mediante el método pause(). Por su parte, el botonStop detiene la reproducción mediante el método stop() y a continuación llama al método prepare() para que el objeto MediaPlayer regrese al estado Prepared y pueda volver a ser reproducido mediante start() (en el caso de que el usuario presione dicho botón).

```
public class MainActivity extends AppCompatActivity {
    private Button botonPlay;
    private Button botonPause;
    private Button botonStop;
    MediaPlayer mp = new MediaPlayer();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        botonPlay = findViewById(R.id.btPlay);
        botonPause = findViewById(R.id.btPause);
        botonStop = findViewById(R.id.btStop);

        mp = MediaPlayer.create(this, R.raw.avicii);
        botonPlay.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                try {
                    mp.start();
                } catch (Exception e){
                }
            }
```

```
        });
        botonPause.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mp.pause();
            }
        });
        botonStop.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mp.stop();
                try {
                    mp.prepare();
                } catch (Exception e) {}
            }
        });
    }
}
```