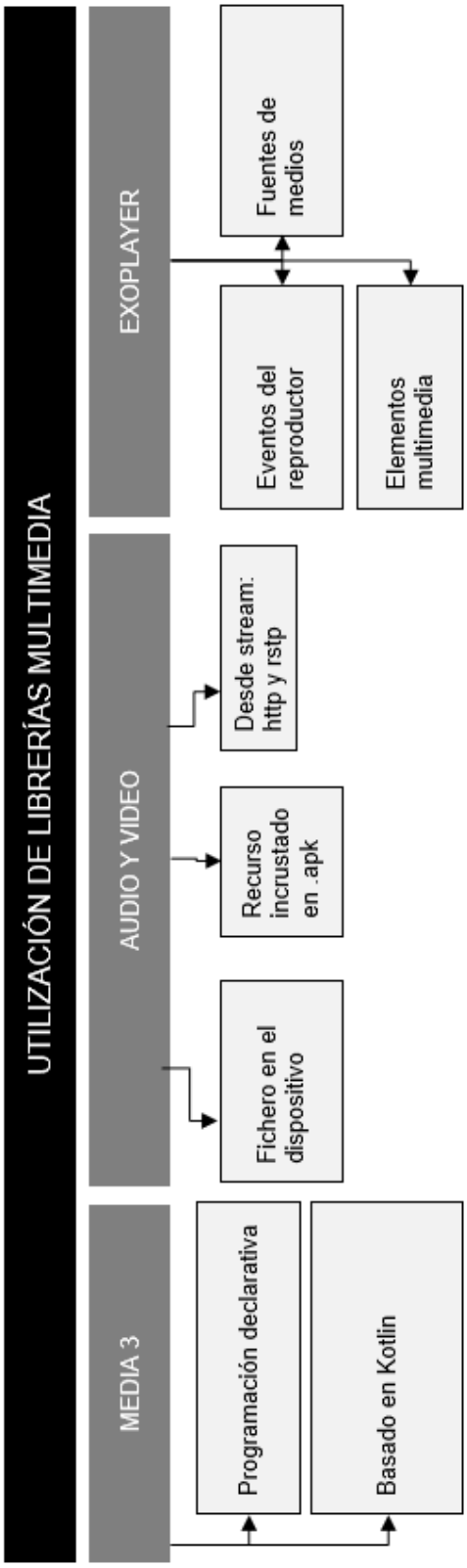
Programación multimedia y dispositivos móviles

Utilización de librerías multimedia

Índice

Esquema


UTILIZACIÓN DE LIBRERÍAS MULTIMEDIA
EXOPLAYER
Fuentes de medios
Eventos del reproductor
Elementos multimedia
AUDIO Y VIDEO
Desde stream: http y rstp
Recurso incrustado en .apk
Fichero en el dispositivo
MEDIA 3
Programación declarativa
Basado en Kotlin

Material de estudio

7.1. Introducción y objetivos

Jetpack Media3 es la nueva plataforma para las bibliotecas multimedia, diseñada para que las aplicaciones de Android ofrezcan experiencias avanzadas de audio y video.

Los objetivos que se persiguen en el siguiente tema son:

- Conocer las distintas clases que gestionan formatos multimedia en Android.
- Conocer los distintos formatos multimedia soportados.
- Conocer la librería Jetpack Media3
- Utilizar esta librería para reproducir archivos de audio y vídeo.
- Aprender a modificar usar ExoPlayer.

7.2. Conceptos y arquitectura de las aplicaciones multimedia

El término multimedia hace referencia al uso combinado de diferentes medios: texto, imagen, sonido, animación y vídeo.

Los dispositivos móviles utilizan de forma combinada estos medios, y permiten la interacción con el usuario. De hecho, muchos dispositivos móviles se utilizan casi únicamente para la reproducción de este tipo de archivos, por lo que es importante poder reproducirlos con nuestras aplicaciones.

Podemos **reproducir audio y vídeo desde orígenes distintos:**

- Desde un fichero almacenado en el dispositivo.

- Desde un recurso incrustado en el paquete de la aplicación (fichero *.apk*).
- Desde un *stream* que es leído desde una conexión de red. En este punto admite dos posibles protocolos: http:// y rstp://

7.3. Clases de API para acceder a servicios multimedia

Media3 proporciona una arquitectura sencilla, con amplias posibilidades de personalización, alta confiabilidad y optimizaciones adaptadas a las capacidades del dispositivo, simplificando así el manejo de la fragmentación.

Media3 ofrece varios componentes clave para casos de uso de reproducción multimedia, en el siguiente diagrama se puede ver como se conjugan los componentes en una aplicación multimedia típica.

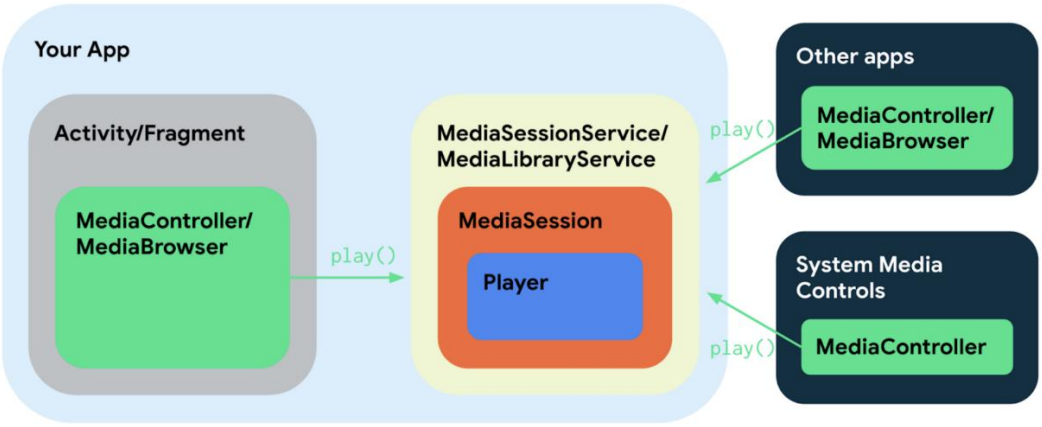


Figura 1: Componentes de una aplicación multimedia. Fuente: https://developer.android.com/

Una de las características más destacadas de Media3 frente a las API multimedia anteriores es que ya no es necesario crear conexiones entre los distintos componentes. La nueva clase MediaSession es compatible con cualquier clase que implemente la interfaz Player, incluyendo la interfaz de usuario. Tanto ExoPlayer

como MediaController son ejemplos de clases que cumplen con esta interfaz, lo que permite una interacción mucho más simple entre los componentes.

Media Player

Es el componente que nos permitirá reproducir los ficheros multimedia.

- Player: es una interface que define las capacidades de alto nivel de un reproductor multimedia (play, stop, pause, buscar, etc.)
- ExoPlayer: Es una implementación en Media3 de la interface Player.

Media Session

Una sesión multimedia proporciona una forma universal de interactuar con un reproductor multimedia. Esto permite a una aplicación avisar la reproducción de medios a fuentes externas y recibir solicitudes de control de reproducción de fuentes externas.

- MediaSession: permiten a tu aplicación interactuar con un reproductor de audio o vídeo. Anuncian la reproducción de medios de forma externa y reciben comandos de reproducción de fuentes externas. Necesitan un Player para ejecutar comandos y obtener el estado actual.
- MediaSessionService: mantiene una sesión multimedia con su reproductor asociado en un servicio separado de la Activity principal de tu aplicación para facilitar la reproducción en segundo plano.
- MediaController: se utiliza generalmente para enviar instrucciones desde fuera de tu aplicación. Los comandos se envían al reproductor viculado de la MediaSession asociada.
- MediaLibraryService: es similar a un MediaSessionService, salvo que incluye APIs adicionales para que puedas servir tu biblioteca de contenidos a las aplicaciones cliente.
- MediaBrowser: Permite al usuario navegar por la biblioteca de contenidos de una aplicación multimedia y seleccionar los elementos que desea reproducir.

UI Components

Media3 proporciona componentes de interfaz de usuario predeterminados para ver vídeo y controlar la reproducción.

- PlayerView: Es vista predeterminada para mostrar el vídeo y los controles de reproducción

Componentes de edición

Media3 incluye la API Transformer para edición multimedia: Procesamiento de audio y vídeo, como añadir filtros y efectos, manejo de formatos especiales, combinar varios archivos de entrada y exportación del resultado final a un archivo.

- Transformer: iniciar y detener transformaciones, así como para comprobar las actualizaciones del progreso de una transformación en ejecución.
- Effects: colección de efectos de audio y vídeo para aplicar a un elemento multimedia.
- EditedMediaItem: representa un elemento multimedia a procesar y las ediciones a aplicarle.

7.4. Audio y vídeo

En este apartado vamos a presentarte las características de los objetos MediaPlayer. Un objeto MediaPlayer puede presentar gran variedad de estados:

- Inicializados sus recursos (initialized).
- Preparando la reproducción (preparing).
- Preparado para reproducir (prepared).
- Reproduciendo (started).
- En pausa (paused).
- Parado (stopped).
- Reproducción completada (playback completed).
- Finalizado (end).
- Con error (error).

Es muy importante saber en qué estado se encuentra un objeto, dado que algunos de los métodos solo pueden ser llamados desde ciertos estados. Por ejemplo, no podemos ponerlo en reproducción (método start()) si no está en estado preparado. O no podremos ponerlo en pausa (pause()), si está parado. Si llamamos a un método no admitido para un determinado estado, se producirá un error de ejecución.

La siguiente imagen nos muestra los métodos que podemos invocar desde cada uno de los estados y cuál es el nuevo estado al que iremos tras invocarlo.

ESTADO SALIDA	ESTADO ENTRADA							
	Idle	**Initialized**	**Preparing**	**Prepared**	**Started**	**Paused**	**Stopped**	**Playback Completed**
Initialized	setDataSource							
Preparing		prepareAsync					prepareAsync	
Prepared		prepare	onPrepared	seekTo			prepare	
Started				start	seekTo start	start		start
Paused					pause	seekTo pause		
Stopped				stop	stop	stop	stop	stop
Playback completed					onCompletion			seekTo
End	release	release	release	release	release	release	release	release
Error	onError	onError	onError	onError	onError	onError	onError	onError

7.5. Exoplayer

Un reproductor multimedia es un componente a nivel de la app que permite la reproducción de archivos de video y audio. Estos archivos se pueden almacenar de forma local o transmitir por Internet. Jetpack Media3 proporciona una interfaz Player que define una funcionalidad básica, como la capacidad de reproducir, pausar, buscar y mostrar información de la pista.

ExoPlayer es la implementación predeterminada de esta interfaz en Media3. En comparación con la API de MediaPlayer de Android, agrega ventajas adicionales, como la compatibilidad con varios protocolos de transmisión, procesadores de audio y video predeterminados, y componentes que controlan el almacenamiento en búfer de contenido multimedia.

Agregar dependencias de ExoPlayer

La forma más fácil de comenzar a usar AndroidX Media3 es agregar dependencias de Gradle en las bibliotecas que necesitas en el archivo build.gradle del módulo de tu app.

Por ejemplo, para depender de ExoPlayer con compatibilidad con la reproducción de DASH y componentes de IU, puedes agregar dependencias en los módulos de la siguiente manera:

```
implementation("androidx.media3:media3-exoplayer:1.5.0")
implementation("androidx.media3:media3-exoplayer-dash:1.5.0")
implementation("androidx.media3:media3-ui:1.5.0")
```

donde 1.5.0 es la versión que prefieras (consulta las notas de la versión para encontrar la versión más reciente). Todos los módulos deben tener la misma versión.

Además, debes habilitar la compatibilidad con Java 8 en todos los archivos build.gradle que dependen de ExoPlayer. Para ello, agrega lo siguiente a la sección Android:

```
compileOptions {
  targetCompatibility JavaVersion.VERSION_1_8
}
```

Crear el reproductor

Puedes crear una instancia de ExoPlayer con ExoPlayer.Builder, que proporciona una variedad de opciones de personalización. El siguiente código es el ejemplo más simple para crear una instancia.

```
val player = ExoPlayer.Builder(context).build()
```

Se debe acceder a las instancias de ExoPlayer desde un solo subproceso de la aplicación. En la gran mayoría de los casos, este debe ser el subproceso principal de la aplicación. Usar el subproceso principal de la aplicación es un requisito cuando se usan los componentes de la IU de ExoPlayer.

Conectar el reproductor a una vista

La biblioteca de ExoPlayer proporciona una variedad de componentes de IU precompilados para la reproducción de contenido multimedia. Entre los mencionados, se incluye PlayerView, que encapsula un PlayerControlView, un SubtitleView y un Surface en los que se renderiza el video. Se puede incluir un PlayerView en el archivo XML del diseño de tu aplicación. Por ejemplo, para vincular el reproductor a la vista, haz lo siguiente:

```
// Bind the player to the view.
playerView.player = player
```

Completa la playlist y prepara el reproductor

En ExoPlayer, cada elemento multimedia está representado por un MediaItem. Para reproducir un elemento multimedia, debes compilar un MediaItem correspondiente, agregarlo al reproductor, prepararlo y llamar a play para iniciar la reproducción:

```
// Build the media item.
val mediaItem = MediaItem.fromUri(videoUri)
// Set the media item to be played.
player.setMediaItem(mediaItem)
// Prepare the player.
player.prepare()
// Start the playback.
player.play()
```

ExoPlayer admite playlists directamente, por lo que es posible preparar el reproductor con varios elementos multimedia para que se reproduzcan uno tras otro:

```
// Build the media items.
val firstItem = MediaItem.fromUri(firstVideoUri)
val secondItem = MediaItem.fromUri(secondVideoUri)
// Add the media items to be played.
player.addMediaItem(firstItem)
player.addMediaItem(secondItem)
// Prepare the player.
player.prepare()
// Start the playback.
player.play()
```

Cómo controlar el reproductor

Una vez que se prepara el reproductor, se puede controlar la reproducción llamando a métodos en el reproductor. Estos son algunos de los métodos más utilizados:

- play y pause inician y pausan la reproducción.
- seekTo permite buscar dentro del contenido multimedia.
- hasPrevious, hasNext, previous y next permiten navegar por la playlist.
- setRepeatMode controla si se reproduce contenido multimedia en bucle y cómo se hace.
- setShuffleModeEnabled controla la reproducción aleatoria de la playlist.
- setPlaybackParameters ajusta la velocidad de reproducción y el tono de audio.

Es importante liberar el reproductor cuando ya no sea necesario para liberar recursos limitados, como decodificadores de video, para que los usen otras aplicaciones. Para ello, llama a `ExoPlayer.release`

Eventos del reproductor

Los eventos, como cambios de estado y errores de reproducción, se informan a los `Player.Listener`. Para registrar un objeto de escucha de modo que reciba dichos eventos debemos de escribir:

```
// Add a listener to receive events from the player.
player.addListener(listener)
```

Player.Listener tiene métodos predeterminados vacíos, por lo que solo debes implementar los métodos que te interesen.

Elementos multimedia

La API de playlist se basa en instancias de MediaItem, que se pueden compilar de forma conveniente con MediaItem.Builder. Dentro del reproductor, un MediaSource.Factory convierte un MediaItem en un MediaSource reproducible. Sin configuración personalizada, una DefaultMediaSourceFactory realiza esta conversión, que puede compilar fuentes de contenido multimedia complejas correspondientes a las propiedades del elemento multimedia.

Fuentes de medios

En ExoPlayer, cada elemento multimedia está representado por un MediaItem. Sin embargo, de forma interna, el reproductor necesita instancias de MediaSource para reproducir el contenido. El reproductor crea estos elementos a partir de elementos multimedia con un MediaSource.Factory.

De forma predeterminada, el reproductor usa un DefaultMediaSourceFactory, que puede crear instancias de las siguientes implementaciones de MediaSource de contenido:

- DashMediaSource para DASH.
- SsMediaSource para SmoothStreaming.
- HlsMediaSource para HLS.
- ProgressiveMediaSource para archivos multimedia normales
- RtspMediaSource para RTSP.

DefaultMediaSourceFactory también puede crear fuentes de medios más complejas según las propiedades de los elementos multimedia correspondientes.

A fondo

JetPack Media 3

https://developer.android.com/media/media3

Documentación en línea. (2024)

Documentación oficial sobre Jetpack Media3.

Formatos compatibles Media3 Explayer

https://developer.android.com/media/media3/exoplayer/supported-formats?hl=es-419 Documentación en línea. (2024)

Documentación oficial sobre Media3 Exoplayer y los formatos compatibles.

Dispositivos compatibles

https://developer.android.com/media/media3/exoplayer/supported-devices?hl=es-419 Documentación en línea. (2024)

Las versiones mínimas de Android necesarias para los casos de uso principales de ExoPlayer.

Entrenamientos

Entrenamiento 1

- Desarrollar una app Android Studio con Kotlin y JepPack Compose en el que se inserten las dependencias de Media3 y Exoplayer.
- Desarrollo paso a paso y solución: https://developer.android.com/media/media3/exoplayer/hello-world?hl=es-419

Entrenamiento 2

- Desarrollar una app Android Studio con Kotlin y JetPack Compose utilizando Media3 y exoplayer.
- Desarrollo paso a paso: https://developer.android.com/media/media3/exoplayer/demo-application?hl=es-419
- Solución: https://github.com/androidx/media/tree/release/demos/main/src/main/java/androidx/media3/demo/main

Entrenamiento 3

- Desarrollar una App en Android de Streaming de video.
- Desarrollo paso a paso y solución: https://www.youtube.com/watch?v=_0F8BU6Jbko

Entrenamiento 4

- Desarrollar una app Android y JetPack Compose para reproducir videos locales.
- Desarrollo paso a paso: y solución: https://www.youtube.com/watch?v=JX1fwti2LI4

Entrenamiento 5

- Desarrollar una app Android y JetPack Compose con media3 con reproducción background y servicios.
- Desarrollo paso a paso: https://www.youtube.com/watch?v=IOHwiK7-J0s
- Solución: https://github.com/sacuar/exo2