DAMM08 UF2. Programació multimèdia.

MediaPlayer.

La Class MediaPlayer ens permet reproduir àudio i/o vídeo.

S'ha de tenir en compte que l'àudio i/o el vídeo a reproduir poden estar ubicats en el mateix projecte (carpeta raw dels recursos), en la memòria secundària del dispositiu (disc SD) o en una ubicació d'Internet.

Reproduir àudio des de la carpeta raw.

Per reproduir àudio des de la pròpia aplicació s'ha de crear la carpeta raw:

- Botó dret a sobre de la carpeta res.
- New → Android Resource Directory.
- En Resource Type escull raw i Acceptar

Posteriorment arrossegar l'arxiu a la carpeta:

L'objecte MediaPlayer s'ha de crear com una propietat de la class i igualant-lo a null ja que el seu estat inicial és que no té carregat cap arxiu:

```
var mMediaPlayer: MediaPlayer? = null
```

Per carregar l'arxiu d'àudio s'ha de crear l'objecte passant l'id de l'àudio. En el nostre cas R.raw.canco1.

Per fer un play, s'ha de cridar al mètode start().

```
fun playSound(sound: Int) {
    if (mMediaPlayer == null) {
        mMediaPlayer = MediaPlayer.create( context: this, sound)
        mMediaPlayer!!.isLooping = true
        mMediaPlayer!!.start()
    } else mMediaPlayer!!.start()
}
```

Per pausar l'àudio s'ha de cridar al mètode pause:

```
if (mMediaPlayer?.isPlaying == true) mMediaPlayer?.pause()
```

Per aturar la reproducció de l'àudio s'ha de cridar al mètode stop. S'ha de tenir en compte que si volem descarregar l'àudio hauríem de cridar al mètode release.

```
if (mMediaPlayer != null) {
    mMediaPlayer!!.stop()
    mMediaPlayer!!.release()
    mMediaPlayer = null
}
```

Reproduir àudio des d'una direcció d'Internet.

Per reproduir àudio des d'una URL, primer hem de carregar la direcció.

```
val uri = Uri.parse( uriString: "https://dhits.frilab.com:8443/dhits")
```

Posteriorment hem de passar la uri al mètode setDataSource de l'objecte MediaPlayer. Amb el mètode SetContentType indiquem que el tipus de contigut (per exemple música).

Abans de fer un start s'ha de fer un prepare per a que el nostre dispositiu agafi el flux de dades de la url i per últim start:

```
fun playContentUri(uri: Uri) {
    var mMediaPlayer: MediaPlayer? = null
    try {
        mMediaPlayer = MediaPlayer().apply { this: MediaPlayer
            setDataSource(application, uri)
            setAudioAttributes(AudioAttributes.Builder() AudioAttributes.Builder
                .setUsage(AudioAttributes.USAGE_MEDIA) AudioAttributes.Builder!
                .setContentType(AudioAttributes.CONTENT_TYPE_MUSIC)
                .build()
            )
            prepare()
            start()
        }
    } catch (exception: IOException) {
        mMediaPlayer?.release()
        mMediaPlayer = null
    }
}
```

Reproduir vídeo.

La vista VideoView permet visualitzar vídeos 3gp i mp4 en Android. Podem definir-la al nostre layout:

```
<VideoView
    android:id="@+id/videoView1"
    android:layout_width="match_parent"
    android:layout_height="400dp"
    android:layout_margin="10dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/titolvideo" />
```

I posteriorment al codi assignar-li el video a reproduir:

```
// Indico la URL del vídeo (en aquest cas un recurs)
videoView!!.setVideoURI(
    Uri.parse( uriString: "android.resource://"
        + packageName + "/" + R.raw.spreedsheet_odoo16))
```

VideoView permet mostrar els controls per moure'ns pel vídeo emprant la class MediaController.

```
// Agafo el videoView de la vista i instancio els controls
videoView = binding.videoView1
mediaControls = MediaController( context: this)
// Indico que el video i els controls estan associats.
mediaControls!!.setAnchorView(videoView)
videoView!!.setMediaController(mediaControls)
```

Finalment, fem un li passem el focus de l'aplicació al videoView i fem start.

```
// Indico que el video és qui té el focus de tot el layout.
videoView!!.requestFocus()

videoView!!.start()
```

Per últim, si volem controlar si el vídeo ha finalitzat utilitzarem el listener setOnCompletionListener.

```
// S'executa quan el vídeo finalitza
videoView!!.setOnCompletionListener {   it: MediaPlayer!
    Toast.makeText( context: this,  text: "Vídeo finalitzat", Toast.LENGTH_LONG).show()
}
```

MediaPlayer té els següents mètodes interessants:

- start(): Visualitzar un VideoView.
- pause(): Pausar un vídeo.
- stopPlayback(): Atura el vídeo.
- setVideoURI (Uri uri): Indiquem la url on es troba el vídeo. Pot ser un recurs local, un vídeo ubicat a la memòria del dispositiu o un vídeo d'Internet.
- seekTo(int millis): Salta mil·lisegons.
- setMediaController(MediaController controller): Permet indicar quins és el controlador de reproducció.
- getDuration(): Torna la duració del vídeo.
- getCurrentPosition(): Torna la posició actual del vídeo.
- isPlaying(): Torna un booleà que indica si un vídeo s'està executant.
- SetAnchorView: Permet indicar la posició dels controls respecte al VideoView.
- setOnPreparedListener: És un listener que s'executa quan el vídeo està preparat per reproduir-se.
- SetOnErrorListener: És un listener que s'executa si un error succeeix mentre es reprodueix el vídeo.
- SetOnCompletionListener: És un listener que s'executa quan el vídeo acaba de reproduir-se.

Para saber más:

- https://www.geeksforgeeks.org/videoview-in-kotlin/
- https://dev4phones.wordpress.com/2020/05/29/reproducir-un-video-de-youtube-dentro-de-una-app-android-usando-kotlin/