Manejo de Audio y Almacenamiento Interno en Android con Java

Para grabar audio, utilizaremos la clase **MediaRecorder** de Android. Para guardarlo, simplemente le indicaremos a MediaRecorder que escriba en una ruta dentro del directorio de la Memoria Interna.

1. Permisos y Componentes Esenciales

A. AndroidManifest.xml

Solo necesitamos permisos para grabar audio y usar el micrófono.

XML

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

<uses-feature android:name="android.hardware.microphone" android:required="false" />
```

B. Permisos en Tiempo de Ejecución (Runtime)

El permiso **RECORD_AUDIO** es un **permiso peligroso**, por lo que debe ser solicitado al usuario en *runtime* (como se explicó en la presentación de permisos).

Java

```java
private static final int RECORD_AUDIO_PERMISSION_CODE = 200;

private boolean checkAudioPermission() {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO)
            != PackageManager.PERMISSION_GRANTED) {
        // Solicita el permiso
```

```
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.RECORD_AUDIO},
                RECORD_AUDIO_PERMISSION_CODE);
        return false;
    }
    return true; // Permiso ya concedido
}
// ... Sobrescribir onRequestPermissionsResult para manejar la respuesta
```

---

2. Preparar el Fichero de Destino en la Memoria Interna

Preparamos la ruta completa del archivo de audio en la Memoria Interna (getFilesDir()).

Java

```
// MainActivity.java
private MediaRecorder mediaRecorder;
private String audioFilePath = null;

private void configurarRutaAudio() {
    // 1. Obtiene el directorio de Memoria Interna
    File storageDir = getFilesDir();

    // 2. Crea un nombre de archivo único
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(new Date());
    String fileName = "AUDIO_" + timeStamp + ".3gp"; // .3gp es un formato común y eficiente

    // 3. Establece la ruta absoluta
    audioFilePath = storageDir.getAbsolutePath() + File.separator + fileName;
```

```
    Log.d("AudioRecorder", "Ruta de grabación: " + audioFilePath);
}
```

---

3. Implementación de MediaRecorder (Grabar Audio)

La grabación de audio requiere varios pasos de configuración antes de empezar y liberar los recursos después.

A. Iniciar la Grabación

```
Java


public void startRecording() {
    if (!checkAudioPermission()) {
        Toast.makeText(this, "Permiso de audio denegado.", Toast.LENGTH_SHORT).show();
        return;
    }

    // 1. Configura la ruta del archivo
    configurarRutaAudio();

    // 2. Inicializa y configura MediaRecorder
    mediaRecorder = new MediaRecorder();
    try {
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC); // Fuente: Micrófono
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP); // Formato de salida
        mediaRecorder.setOutputFile(audioFilePath); // ¡Ruta de Memoria Interna!
```

```
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB); // Codificador de audio

        // 3. Prepara y empieza
        mediaRecorder.prepare();
        mediaRecorder.start();

        Toast.makeText(this, "Grabando audio...", Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
        Log.e("AudioRecorder", "Fallo en prepare() o start(): " + e.getMessage());
        releaseRecorder(); // Asegura la liberación de recursos en caso de error
    }
}
```

B. Detener la Grabación y Liberar Recursos

Java

```
public void stopRecording() {
    if (mediaRecorder != null) {
        try {
            mediaRecorder.stop(); // Detiene la grabación
            Toast.makeText(this, "Grabación finalizada en Memoria Interna.", Toast.LENGTH_LONG).show();
        } catch (RuntimeException e) {
            // Manejar la excepción si stop() es llamado sin start() o si el archivo es inválido
            Log.e("AudioRecorder", "Fallo al detener la grabación: " + e.getMessage());
            // Opcional: Borrar el archivo inválido
            new File(audioFilePath).delete();
            audioFilePath = null;
        } finally {
            releaseRecorder();
        }
```

```
    }
}

private void releaseRecorder() {
    if (mediaRecorder != null) {
        mediaRecorder.release(); // Libera los recursos del hardware
        mediaRecorder = null;
    }
}
```

---

4. Reproducir el Audio Grabado

Para reproducir el archivo de audio guardado, usaremos la clase **MediaPlayer**.

Java

```
private MediaPlayer mediaPlayer;

public void playRecording() {
    if (audioFilePath == null) {
        Toast.makeText(this, "No hay audio grabado para reproducir.", Toast.LENGTH_SHORT).show();
        return;
    }

    releasePlayer(); // Asegura que no haya otro reproductor activo

    mediaPlayer = new MediaPlayer();
    try {
        // 1. Establece la fuente de datos (la ruta de Memoria Interna)
```

```
        mediaPlayer.setDataSource(audioFilePath);

        // 2. Configura el listener de finalización
        mediaPlayer.setOnCompletionListener(mp -> releasePlayer());

        // 3. Prepara y reproduce
        mediaPlayer.prepare();
        mediaPlayer.start();

        Toast.makeText(this, "Reproduciendo audio...", Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
        Log.e("AudioPlayer", "Fallo al reproducir: " + e.getMessage());
    }
}

private void releasePlayer() {
    if (mediaPlayer != null) {
        mediaPlayer.release();
        mediaPlayer = null;
    }
}
```

---

5. Consideraciones Clave

- **Recursos de Hardware:** Tanto MediaRecorder como MediaPlayer utilizan recursos de hardware (micrófono, códecs). Es **crítico** llamar a **release()** en ambos objetos cuando termines de usarlos, o podrías causar fugas de memoria o bloquear el hardware para otras aplicaciones.

- **Estado de Vida de la Actividad:** Llama a **stopRecording()** y **releasePlayer()** en los métodos de ciclo de vida apropiados, como **onPause()** o **onDestroy()**, para liberar el micrófono si el usuario sale de tu actividad.
- **Hilos:** Las operaciones de I/O de audio (grabación, reproducción) deben realizarse en el **hilo principal**. Sin embargo, **mediaPlayer.prepare()** puede ser lento. Para evitar bloqueos, considera usar **mediaPlayer.prepareAsync()** y un **OnPreparedListener**.