Manejo de Audio y Almacenamiento Externo en Android con Java

Utilizaremos la clase **MediaRecorder** y lo dirigiremos a un archivo dentro del directorio de **Almacenamiento Externo Compartido** (o un directorio específico de la app en la externa, si se requiere que se elimine con la desinstalación).

1. Permisos Esenciales (¡Cruciales!)

Esta operación necesita dos permisos peligrosos que deben ser solicitados al usuario en tiempo de ejecución.

A. AndroidManifest.xml

XML

```xml
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    android:maxSdkVersion="28" />
```

B. Permisos en Tiempo de Ejecución (Runtime)

Debes verificar y solicitar ambos permisos: **RECORD_AUDIO** y **WRITE_EXTERNAL_STORAGE** (o manejar la lógica de **Scoped Storage** si apuntas a API 29+).

Java

```java
// Ejemplo simplificado de verificación de permisos combinada
private static final int PERMISSION_REQUEST_CODE = 201;

private boolean checkAndRequestPermissions() {
    int recordAudio = ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO);
    // Para API < 29, verificamos WRITE_EXTERNAL_STORAGE
    int writeStorage = ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE);

    List<String> listPermissionsNeeded = new ArrayList<>();
```

```
    if (recordAudio != PackageManager.PERMISSION_GRANTED) {
        listPermissionsNeeded.add(Manifest.permission.RECORD_AUDIO);
    }
    if (Build.VERSION.SDK_INT <= Build.VERSION_CODES.P && writeStorage != PackageManager.PERMISSION_GRANTED) {
        listPermissionsNeeded.add(Manifest.permission.WRITE_EXTERNAL_STORAGE);
    }

    if (!listPermissionsNeeded.isEmpty()) {
        ActivityCompat.requestPermissions(this,
                listPermissionsNeeded.toArray(new String[0]),
                PERMISSION_REQUEST_CODE);
        return false;
    }
    return true;
}
// ... Sobrescribir onRequestPermissionsResult para manejar la respuesta
```

---

2. Preparar el Fichero de Destino en la Memoria Externa

El mejor lugar para guardar archivos de audio públicos es el directorio **Environment.DIRECTORY_MUSIC** o **Environment.DIRECTORY_RECORDINGS**.

Java

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

private void configurarRutaAudioExterno() throws IOException {
```

```
    // 1. Obtiene el directorio de Música/Grabaciones compartidas.
    // Usamos getExternalStoragePublicDirectory (obsoleto, pero común en Java antiguo)
    // Para API 29+, se debe usar MediaStore.
    File storageDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC);

    // Asegura que el directorio exista
    if (!storageDir.exists()) {
        storageDir.mkdirs();
    }

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

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

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

---

3. Implementación de MediaRecorder (Grabar Audio)

El código para startRecording(), stopRecording(), releaseRecorder() y playRecording() es **exactamente el mismo** que se usó para la Memoria Interna, con la única diferencia de la ruta establecida en mediaRecorder.setOutputFile(audioFilePath);.

Pasos clave:

1. Llamar a **configurarRutaAudioExterno()** para obtener la ruta externa.
2. Configurar MediaRecorder usando el audioFilePath generado.
3. Llamar a mediaRecorder.prepare() y mediaRecorder.start().

4. Llamar a mediaRecorder.stop() y **release()** al finalizar.

Java

```
public void startRecording() {
    if (!checkAndRequestPermissions()) {
        return;
    }

    // El resto de la lógica es la misma:
    try {
        configurarRutaAudioExterno(); // <--- Aquí se usa la ruta externa
        mediaRecorder = new MediaRecorder();
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        mediaRecorder.setOutputFile(audioFilePath); // <--- Asigna la ruta externa
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);

        mediaRecorder.prepare();
        mediaRecorder.start();

        Toast.makeText(this, "Grabando audio en Almacenamiento Externo...", Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
        Log.e("AudioRecorder", "Fallo: " + e.getMessage());
        // Manejar errores...
    }
}
// ... stopRecording() y playRecording() siguen la misma lógica interna.
```

---

4. Notificar al Sistema de Medios (¡Específico de Almacenamiento Externo!)

Dado que el archivo se ha escrito en una ubicación compartida, el sistema de medios (como la galería de música) no lo detectará inmediatamente. Debes **forzar un escaneo** para que el archivo aparezca en otras aplicaciones.

Java

```
public void notifyMediaScanner(String path) {
    File f = new File(path);
    // Notifica al Media Scanner para que indexe el nuevo archivo.
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    Uri contentUri = Uri.fromFile(f);
    mediaScanIntent.setData(contentUri);
    this.sendBroadcast(mediaScanIntent);

    Log.d("AudioScanner", "Notificación enviada para: " + path);
}

// Llama a esto después de stopRecording() si la grabación fue exitosa.
// notifyMediaScanner(audioFilePath);
```

---

5. Consideraciones Clave para la Memoria Externa

- **Permisos de Almacenamiento:** Este es el punto más sensible. Debes obtener la aprobación del usuario en *runtime*. Si el usuario niega el permiso, la grabación fallará.
- **Scoped Storage (API 29+):** Si tu targetSdkVersion es 29 o superior, el uso de Environment.getExternalStoragePublicDirectory() está obsoleto y se requiere la **MediaStore API** para escribir archivos públicos de forma segura. Si quieres evitar MediaStore y seguir usando WRITE_EXTERNAL_STORAGE, debes establecer android:requestLegacyExternalStorage="true" en el Manifest (una solución temporal).
- **Persistencia:** Recuerda que el audio permanecerá en el dispositivo del usuario **incluso si desinstalan tu app**.