Manejo de la Cámara y Almacenamiento Interno en Android con Java

A diferencia de la Memoria Externa, guardar la imagen directamente en la Memoria Interna:

1. **No requiere el permiso WRITE_EXTERNAL_STORAGE**.
2. Asegura que la imagen sea **privada** para tu app.
3. Garantiza que la imagen se **borre** al desinstalar la app.

1. Permisos Requeridos y Configuración

Solo necesitas el permiso para usar la cámara y la configuración de FileProvider para compartir la ruta de forma segura.

A. AndroidManifest.xml

XML

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

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

<application ...>
    <provider
        android:name="androidx.core.content.FileProvider"
        android:authorities="${applicationId}.provider"
        android:exported="false"
        android:grantUriPermissions="true">
```

```
        <meta-data
            android:name="android.support.FILE_PROVIDER_PATHS"
            android:resource="@xml/file_paths" />
    </provider>
</application>
```

B. Permisos en Tiempo de Ejecución

Solo necesitas solicitar el permiso **CAMERA** en tiempo de ejecución (como se explicó en una presentación anterior).

---

2. Configurar FileProvider para Memoria Interna (res/xml/file_paths.xml)

Para permitir que la aplicación de la cámara acceda a tu almacenamiento interno de forma segura, debes modificar el archivo XML de rutas del FileProvider.

res/xml/file_paths.xml

XML

```
<?xml version="1.0" encoding="utf-8"?>
<paths>
    <files-path
        name="internal_files"
        path="/" />
</paths>
```

💡 **files-path** mapea directamente al directorio devuelto por Context.getFilesDir().

---

3. Preparar el Fichero de Destino en la Memoria Interna

Creamos el archivo en el directorio privado de la app (getFilesDir()) antes de llamar a la cámara.

Java

```
// MainActivity.java
private String currentPhotoPath; // Para almacenar la ruta absoluta del archivo

private File crearArchivoDeImagenInterna() throws IOException {
    // 1. Crea un nombre de archivo único basado en la marca de tiempo.
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";

    // 2. Obtiene el directorio de destino: getFilesDir() para Memoria Interna.
    File storageDir = getFilesDir();

    // 3. Crea el archivo temporal en el directorio de Memoria Interna.
    File image = File.createTempFile(
        imageFileName,  /* Prefijo */
        ".jpg",         /* Sufijo */
        storageDir      /* Directorio */
    );

    // Guarda la ruta para referencia posterior.
    currentPhotoPath = image.getAbsolutePath();
    return image;
}
```

---

4. Llamar a la Cámara usando Intent

El procedimiento es idéntico al de la Memoria Externa, pero la Uri que generamos apunta a la Memoria Interna gracias al

FileProvider configurado.

Java

```java
private static final int REQUEST_IMAGE_CAPTURE = 2; // Usamos un código diferente

private void dispatchTakePictureIntentInterna() {
    // 1. **Verificar Permiso de CÁMARA aquí** (usando la lógica de runtime).
    // if (!checkCameraPermission()) { return; }

    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        File photoFile = null;
        try {
            // Llama a la función que crea el archivo en Memoria Interna.
            photoFile = crearArchivoDeImagenInterna();
        } catch (IOException ex) {
            ex.printStackTrace();
        }

        if (photoFile != null) {
            // 2. Obtiene la URI del archivo a través de FileProvider.
            Uri photoURI = FileProvider.getUriForFile(this,
                getApplicationContext().getPackageName() + ".provider",
                photoFile);

            // 3. Pasa la URI al Intent para que la cámara guarde allí la imagen.
            takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
```

```
            // 4. Lanza el Intent esperando un resultado.
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
        }
    }
}
```

---

5. Manejar el Resultado y Cargar la Imagen

Cuando la cámara regresa con éxito (RESULT_OK), la imagen ya estará guardada en la ruta de Memoria Interna (currentPhotoPath).

Java

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        // La foto está guardada de forma privada en la ruta: currentPhotoPath.

        // 1. Muestra la imagen (requiere reducir la escala).
        mostrarImagenInterna(currentPhotoPath);

        // La imagen NO necesita ser escaneada por la galería (es privada).
    }
}

private void mostrarImagenInterna(String path) {
    // Nota: ¡Asegúrate de escalar el Bitmap antes de cargarlo a la UI!
    // Bitmap bitmap = BitmapFactory.decodeFile(path);
```

```
    // ... Cargar el bitmap en tu ImageView.

    Toast.makeText(this, "Foto privada guardada en: " + path, Toast.LENGTH_LONG).show();
}
```

---

6. Ventajas del Uso de Memoria Interna

- **Simplificación de Permisos:** Solo necesitas CAMERA. El permiso de almacenamiento peligroso (WRITE_EXTERNAL_STORAGE) se evita por completo.
- **Privacidad:** La imagen no es accesible por otras aplicaciones ni aparece en la Galería del usuario.
- **Limpieza Automática:** Cuando la aplicación se desinstala, la imagen se elimina automáticamente, manteniendo limpio el dispositivo del usuario.