Manejo de la Cámara y Almacenamiento Externo en Android con Java

Este proceso se basa en el uso de un **Intent implícito** para delegar la tarea de tomar la foto a la aplicación de cámara del sistema, mientras nosotros solo nos encargamos de preparar la ubicación para el archivo.

1. Permisos Requeridos (y la Seguridad)

Para esta tarea, necesitamos tres cosas: usar la cámara, guardar el archivo en el almacenamiento externo y, a partir de Android 10 (API 29+), asegurar la compatibilidad con el sistema de archivos moderno.

A. AndroidManifest.xml

Declara los permisos necesarios:

XML

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
  android:maxSdkVersion="28" /> <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 (Runtime)

Al igual que en la presentación anterior, debes solicitar los permisos de CAMERA y WRITE_EXTERNAL_STORAGE al usuario en tiempo de ejecución.

---

2. Preparar el Fichero de Destino en la Memoria Externa

Necesitamos crear un archivo temporal en un directorio público para que la aplicación de la cámara sepa dónde guardar la imagen de alta resolución. Usaremos el directorio Pictures.

Java

```
// MainActivity.java
private File archivoDeFoto;
private String currentPhotoPath;

private File crearArchivoDeImagen() 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 público de fotos del dispositivo.
    // Usamos getExternalFilesDir(Environment.DIRECTORY_PICTURES) para almacenar la imagen
    // en un subdirectorio privado de la app en External Storage (se borra al desinstalar).
    // Si quieres que persista, usa Environment.getExternalStoragePublicDirectory().
    File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);

    // 3. Crea el archivo temporal.
    File image = File.createTempFile(
        imageFileName,  /* Prefijo */
        ".jpg",         /* Sufijo */
        storageDir      /* Directorio */
```

```
    );

    // Guarda la ruta del archivo: necesaria para el Intent de la cámara y para cargar la imagen después.
    currentPhotoPath = image.getAbsolutePath();
    return image;
}
```

---

3. Configurar FileProvider (res/xml/file_paths.xml)

La aplicación de la cámara necesita una **URI** (un identificador de contenido) para saber dónde guardar la imagen, pero no puede usar el File directamente a partir de Android N (API 24). Debemos usar **FileProvider**.

res/xml/file_paths.xml (Crea este archivo en la carpeta res/xml):

XML

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

---

4. Llamar a la Cámara usando Intent

Ahora utilizamos el Intent y el FileProvider para lanzar la cámara y especificar la URI de destino.

Java

```
private static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    // Asegúrate de tener los permisos necesarios (paso 1).
    if (!checkCameraPermission()) { // Asume que tienes una función para verificar permisos
        // Solicita permisos si no están
        return;
    }

    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    // Asegúrate de que haya una actividad de cámara para manejar el Intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        File photoFile = null;
        try {
            // 1. Crea el archivo de destino.
            photoFile = crearArchivoDeImagen();
        } catch (IOException ex) {
            // Error al crear el archivo
            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 aplicación de la cámara finaliza, se llama a onActivityResult. Si la foto fue exitosa, la imagen ya estará guardada en la ruta 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 se tomó con éxito. La imagen ya está en currentPhotoPath.

        // 1. Muestra una vista previa (opcional: requiere reducir la escala).
        mostrarImagenEnImageView(currentPhotoPath);

        // 2. Notifica a la Galería (si usaste un directorio público compartido)
        // galleryAddPic(currentPhotoPath);
    }
}

private void mostrarImagenEnImageView(String path) {
    // Usar Bitmaps es intensivo en memoria. Se recomienda escalarlos.
    // ImageView imageView = findViewById(R.id.mi_imageView);
```

```
    try {
        Bitmap bitmap = BitmapFactory.decodeFile(path);
        // imageView.setImageBitmap(bitmap);
        Toast.makeText(this, "Foto cargada con éxito.", Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

---

6. Resumen de Buenas Prácticas

- **FileProvider (API 24+):** Es **obligatorio**. Nunca pases el objeto File directamente a la cámara; usa la URI generada por FileProvider para compartir el archivo de forma segura.
- **Permisos:** Siempre verifica y solicita los permisos de CAMERA y WRITE_EXTERNAL_STORAGE (en dispositivos pre-Android 10) antes de lanzar el Intent.
- **Rendimiento:** Las imágenes de cámara son grandes. Utiliza **BitmapFactory.Options** para reducir la escala de la imagen (*scaling*) antes de cargarla en un ImageView para evitar errores de **OutOfMemoryError**.
- **Almacenamiento:** Prefiere usar getExternalFilesDir(Environment.DIRECTORY_PICTURES) si la imagen solo la necesita tu app. Si la foto debe ser visible en la galería del usuario, usa la API moderna de **MediaStore** (es la forma correcta para Android 10+).