



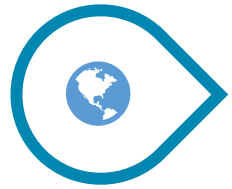
# USO DE LA CÁMARA CON ANDROID

DESARROLLO DE APLICACIONES MÓVILES AVANZADAS

**LUIS HUMBERTO RIVAS RODRÍGUEZ**

INGENIERO EN SISTEMAS INFORMÁTICOS  
Y MÁSTER EN DIRECCIÓN ESTRATÉGICA DE EMPRESAS.

## Agenda



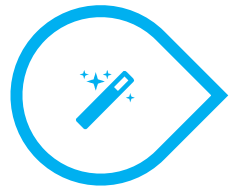
# 1 Generalidades

Conceptos

## MediaStore

Funcionalidades

2



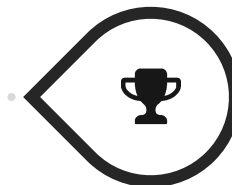
# 3 Desafío de Clases

Ejemplo Práctico

## Cierre

Valoraciones Finales

4



# RESULTADOS DE APRENDIZAJE

Competencias a desarrollar al finalizar la sesión.



## **INTERPRETA**

Los componentes para usar la cámara dentro de una interfaz gráfica en Android.



## **COMPRENDE**

La relación que tienen las librerías y su importancia en el desarrollo de aplicaciones basadas en Android.



## **CONSTRUYE**

Aplicaciones Android haciendo uso de los componentes avanzados como son las librerías en Android.

# CONTEXTO DE USO

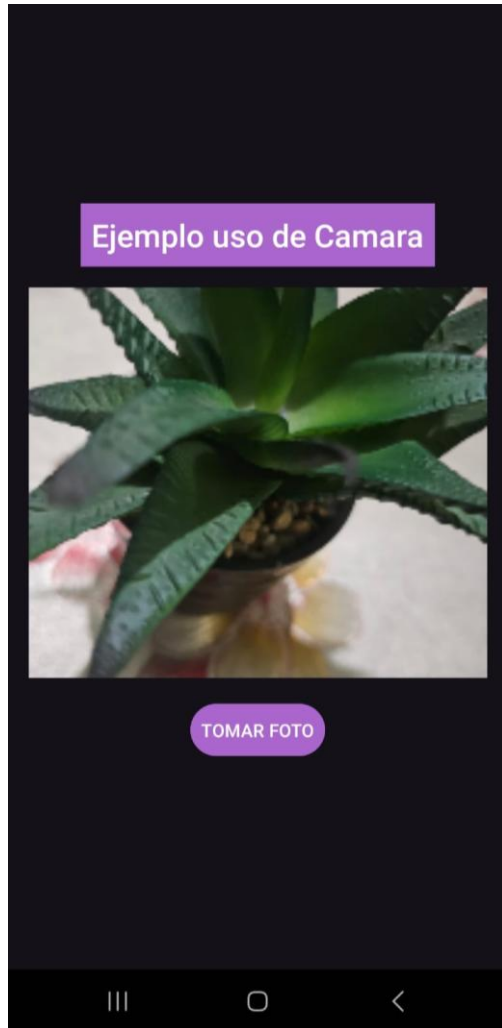
Requerimientos principales



- En algún momento se requiere hacer uso del hardware de nuestro dispositivo para utilizar funciones de una aplicación.
- Para esta sesión nos vamos a enfocar en la clase **MediaStore** y el uso de Activities de sistema que ya trae Android.

# MEDIAPLAYER

## COMPONENTE



- El marco de trabajo de contenido multimedia de Android admite la gestión del hardware de nuestras aplicaciones haciendo uso de las clases que ya proporciona nativamente Android.
- La clase [MediaStore.ACTION\\_IMAGE\\_CAPTURE](#) nos permitirá combinarla con [ActivityResultContracts](#) para generar un proceso como lo hacen muchas aplicaciones que utilizamos hoy en día por ejemplo un cliente de correo electrónico que necesita adjuntar una imagen, una aplicación de mensajería instantánea que requiere enviar una imagen y así con las diferentes aplicaciones disponibles.

# PERMISOS – PARTE 1

## CONTEXTUALIZACIÓN

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

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

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES"
        tools:ignore="SelectedPhotoAccess" />
    <uses-permission android:name="android.permission.READ_MEDIA_VIDEO"/>


```

- Se debe asignar los permisos a utilizar en la aplicación.
- Además, se deberá crear un Provider para la gestión de los archivos y la memoria interna del dispositivo.

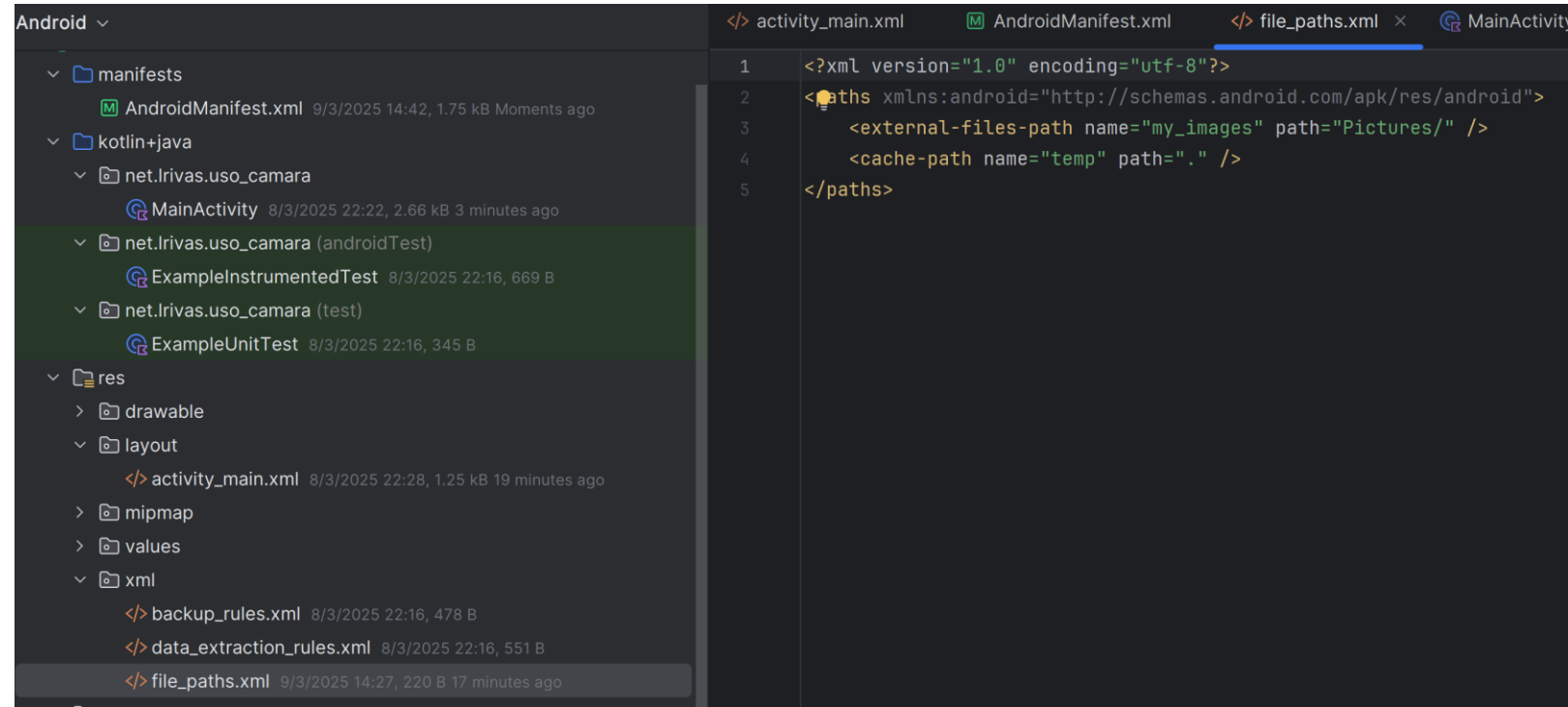
```
<provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="net.lrivass.uso_camara.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths" />
</provider>

<activity
```

# PERMISOS – PARTE 2

## COMPONENTES

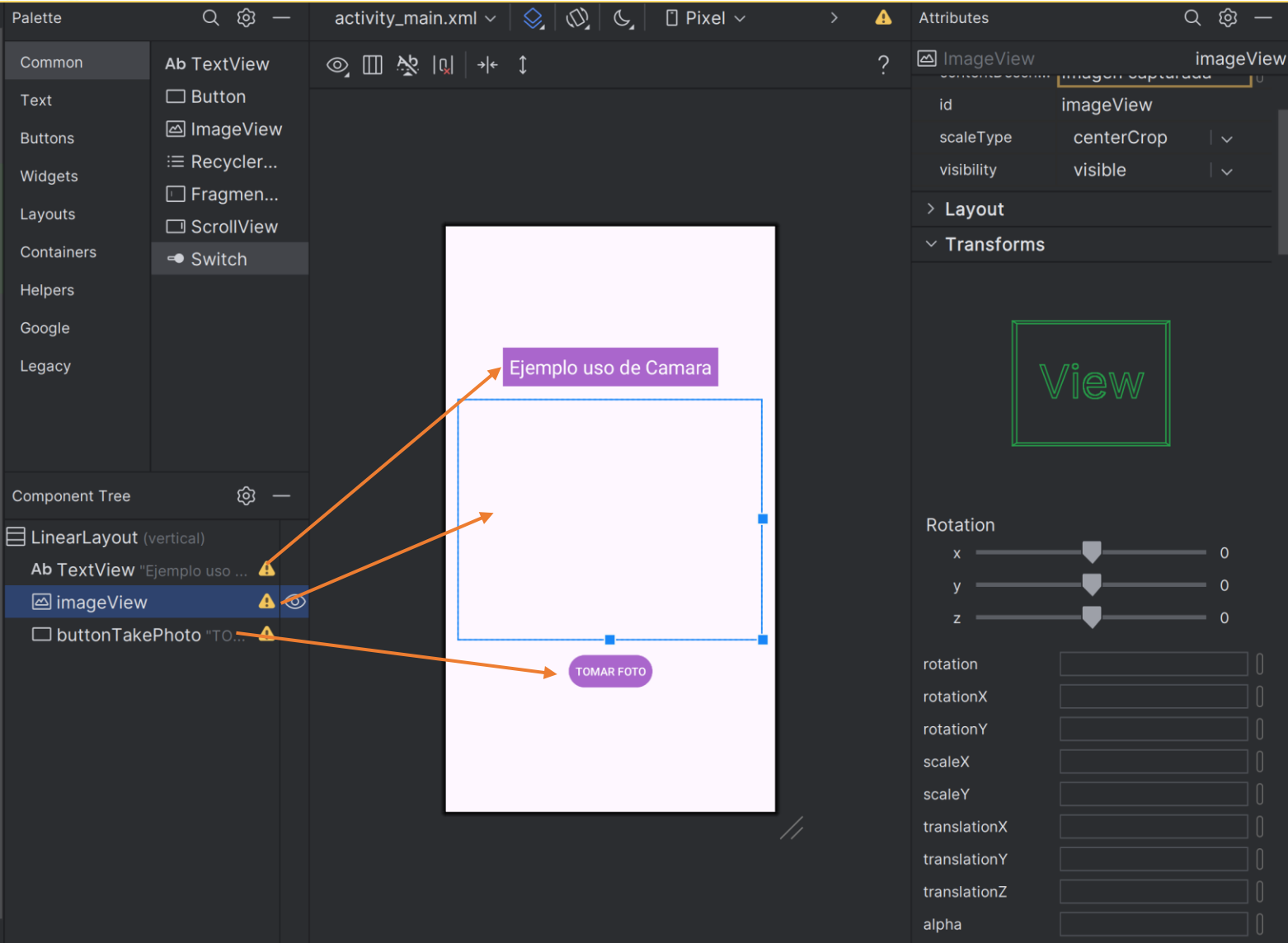
- Se debe crear el archivo que gestiona el provider, para ello se debe crear el directorio especial xml y dentro de él, un archivo con el contenido que se muestra.



```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-files-path name="my_images" path="Pictures/" />
    <cache-path name="temp" path="." />
</paths>
```

# DISEÑO DE LA INTERFAZ

## USO DEL CONTROL



- Jugaremos con la opción de tomar una foto y luego ver la imagen que se tomó con el dispositivo.



# DÁNDOLE FUNCIONAMIENTO

## CONTROL

- Se debe definir ciertas variables de trabajo y los permisos tanto para usar la cámara como para tomar una fotografía. No olvidar incluir el método `registerForActivityResult`.

```
class MainActivity : AppCompatActivity() {

    private lateinit var btnTomarFoto: Button
    private lateinit var imgFoto: ImageView
    private lateinit var currentPhotoPath: String

    private val takePictureLauncher = registerForActivityResult(ActivityResultContracts.TakePicture()) { success ->
        if (success) {
            imgFoto.setImageURI(Uri.parse(currentPhotoPath))
            Toast.makeText(context: this, text: "Foto tomada y guardada", Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText(context: this, text: "Error al tomar la foto", Toast.LENGTH_SHORT).show()
        }
    }

    private val requestPermissionLauncher = registerForActivityResult(ActivityResultContracts.RequestPermission()) { isGranted ->
        if (isGranted) {
            realizarProcesoFotografia()
        } else {
            Toast.makeText(context: this, text: "Permiso de cámara denegado", Toast.LENGTH_SHORT).show()
        }
    }
}
```

# DÁNDOLE FUNCIONAMIENTO

## CONTROL

```
private fun realizarProcesoFotografia() {
    when {
        ContextCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED -> {
            val photoFile = createImageFile()
            val photoUri = FileProvider.getUriForFile(
                context: this,
                authority: "net.lrivas.ejemplocamara.fileprovider",
                photoFile
            )
            currentPhotoPath = photoFile.absolutePath
            takePictureLauncher.launch(photoUri)
        }
        shouldShowRequestPermissionRationale(Manifest.permission.CAMERA) -> {

            Toast.makeText( context: this, text: "Se necesita el permiso de cámara para tomar fotos", Toast.LENGTH_LONG).show()
            requestPermissionLauncher.launch(Manifest.permission.CAMERA)
        }
        else -> {

            requestPermissionLauncher.launch(Manifest.permission.CAMERA)
        }
    }
}
```

# DÁNDOLE FUNCIONAMIENTO

## CONTROL

- Estos métodos acompañados de los códigos de petición y los permisos solicitados en el archivo AndroidManifest permitirán gestionar la memoria y la cámara.
- Una vez realizado esto, quedará los métodos para tomar la foto y guardarla en la memoria del dispositivo.

```
private fun createImageFile(): File {  
    val timeStamp = SimpleDateFormat( pattern: "yyyyMMdd_HHmmss", Locale.getDefault()).format(Date())  
    val storageDir = getExternalFilesDir( type: "Pictures")  
    return File.createTempFile(  
        prefix: "JPEG_${timeStamp}_",  
        suffix: ".jpg",  
        storageDir  
    ).apply {  
        currentPhotoPath = absolutePath  
    }  
}
```

# DÁNDOLE FUNCIONAMIENTO

## CONTROL

```
private fun crearArchivo(): File {
    val timeStamp = SimpleDateFormat( pattern: "yyyyMMdd_HH:mm:ss", Locale.getDefault()).format(Date())
    val storageDir = getExternalFilesDir( type: "Pictures")
    return File.createTempFile(
        prefix: "JPEG_${timeStamp}_",
        suffix: ".jpg",
        storageDir
    ).apply {
        currentPhotoPath = absolutePath
    }
}
```

- El método **realizarProcesoFotografia()** permite invocar la activity de la cámara.
- Se auxilia de **crearArchivo()** para generar una nomenclatura única de la foto tomada.

```
private fun realizarProcesoFotografia() {
    when {
        ContextCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED -> {
            val photoFile = createImageFile()
            val photoUri = FileProvider.getUriForFile(
                context: this,
                authority: "net.lrvivas.uso_camara.fileprovider",
                photoFile
            )
            currentPhotoPath = photoFile.absolutePath
            takePictureLauncher.launch(photoUri)
        }
        shouldShowRequestPermissionRationale(Manifest.permission.CAMERA) -> {
            Toast.makeText( context: this, text: "Se necesita el permiso de cámara para tomar fotos", Toast.LENGTH_LONG).show()
            requestPermissionLauncher.launch(Manifest.permission.CAMERA)
        }
        else -> {
            requestPermissionLauncher.launch(Manifest.permission.CAMERA)
        }
    }
}
```



- Tomando como referencia los contenidos visto en la sesión, elabore una aplicación que permita utilizar el componente de cámara del dispositivo, guardar la foto en la memoria interna y mostrar la foto obtenida.



# VALORACIONES FINALES

Comentarios sobre el tema.

1

## **MediaStore.ACTION\_IMAGE\_CAPTURE**

Es la clase encargada de gestionar la actividad de la cámara en una aplicación android.

2

## **Provider**

Es un componente de seguridad que permite a Android delimitar el uso de los recursos de archivos en nuestras aplicaciones.

3

## **Creativo**

Saber distribuir y seleccionar los diferentes métodos y propiedades de una clase, permitirá al programador trabajar de forma rápida y eficiente



# ¿PREGUNTAS?

DESARROLLO DE APLICACIONES MÓVILES AVANZADAS

**LUIS HUMBERTO RIVAS RODRÍGUEZ**

INGENIERO EN SISTEMAS INFORMÁTICOS  
Y MÁSTER EN DIRECCIÓN ESTRATÉGICA DE EMPRESAS.