

Bloque 3. Ejercicio Resuelto Intent Implícito

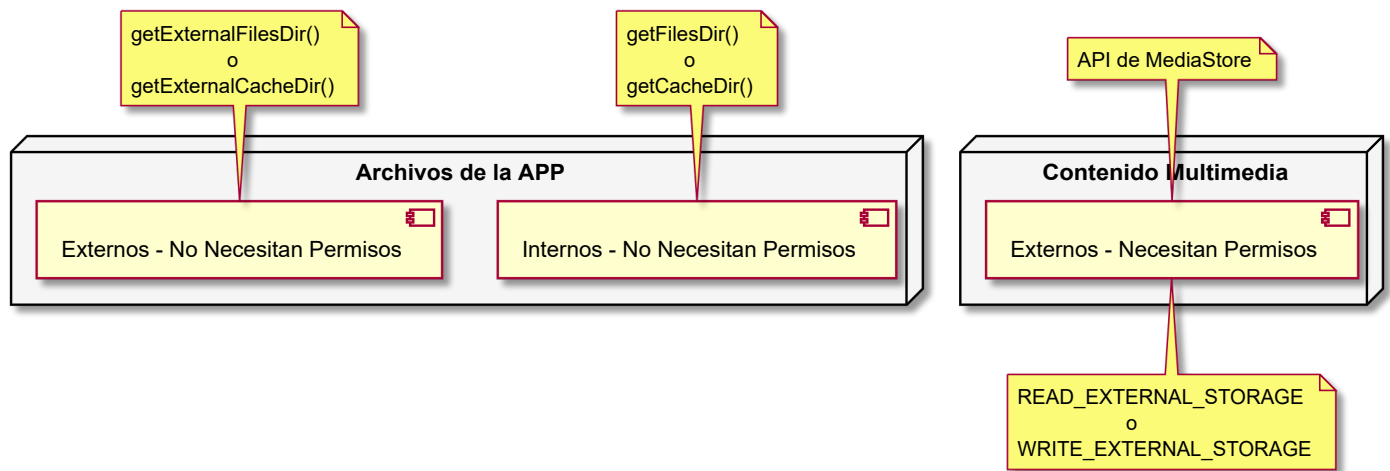
[Descargar este ejercicio](#)

Almacenamiento de información en Android

Android utiliza un [sistema de archivos](#) que es similar a los sistemas de archivos basados en discos de otras plataformas. El sistema proporciona varias opciones para que guardes los datos de tu app:

- **Almacenamiento específico de la app:** Almacena archivos diseñados solo para tu app, ya sea en directorios dedicados dentro de un volumen de **almacenamiento interno** o en directorios dedicados diferentes dentro del **almacenamiento externo**. Usa los directorios del almacenamiento interno para guardar información sensible a la que otras apps no deberían acceder.
- **Almacenamiento compartido:** Almacena archivos que tu app pretenda compartir con otras apps, incluidos archivos multimedia, documentos y otros.
- **Preferencias:** Almacena datos primitivos y privados en pares clave-valor.
- **Bases de datos:** Almacena datos estructurados en una base de datos privada mediante la biblioteca de persistencias Room.

⚠ Para darles a los usuarios más control sobre sus archivos y acotar el desorden, las aplicaciones orientadas a Android 10 (nivel de API 29) y versiones posteriores reciben acceso con alcance determinado al almacenamiento externo, o **almacenamiento específico**, de forma predeterminada. Estas apps solo tienen acceso al directorio específico de la aplicación en el almacenamiento externo, así como a tipos específicos de contenido multimedia que creó la app.



✎ Los almacenamientos externos, sean archivos pertenecientes a la App o contenido multimedia serán accesibles por otras aplicaciones. Los archivos de la App se eliminarán cuando se borre esta, tanto los internos como los externos.

Los volúmenes extraíbles, como las **tarjetas SD**, aparecen en el sistema de archivos como parte del almacenamiento externo. Android representa estos dispositivos mediante una ruta de acceso, como **/sdcard**. Para acceder a este almacenamiento, además de pedir permiso se tendrá que declarar un content provider en el manifest.

Ejercicio

Realizar una aplicación que contenga tres botones. El primer botón tiene abrir el navegador con la siguiente búsqueda *"iesdoctorbalmis"*. El segundo botón nos permitirá abrir una aplicación externa que hayáis instalado previamente. Y el tercer botón lanzará una canción mp3 que se encuentre en la tarjeta sd del dispositivo.

PASO I. Preparar la vista de la aplicación

Para ello tendremos que modificar el archivo **activity_main.xml** con lo necesario para crear los tres botones que estén más o menos centrados. Una posible solución podría ser la siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/navegador"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:text="@string/abrir_navegador"/>
    <Button
        android:id="@+id/abrir"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:text="@string/abrir_aplicaci_n"/>
    <Button
        android:id="@+id/escuchar"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:text="@string/escuchar_canci_n"/>
</LinearLayout>
```

PASO II. Codificar la ejecución del primer botón

El método siguiente es el que corresponde con la codificación del primer botón. Tendremos que añadirlo dentro del método `onCreate` de la `MainActivity.kt`

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    3 setContentView(R.layout.activity_main)
    4 findViewById<Button>(R.id.navegador).setOnClickListener (View.OnClickListener {
    5         val intent = Intent(Intent.ACTION_WEB_SEARCH).apply {
                putExtra(SearchManager.QUERY, "iesdoctorbalmis")
    7         }
    8         if (intent.resolveActivity(packageManager) != null) startActivity(intent)
    })
}
```

🔗 la **línea 3** lo que hace es cargar el layout que queremos asociar con la actividad principal y asignarlo al contexto de la aplicación, **Línea 4** el método `findViewById` busca la vista en el layout del contexto y la hincha (crear el objeto del tipo correspondiente asociado a esa vista, en este caso `Button`). **Línea 7 y 8** crean el `Intent` de búsqueda en la web e inicia la actividad

que lo resuelva, siempre y cuando haya alguna actividad instalada en el dispositivo que lo pueda resolver.

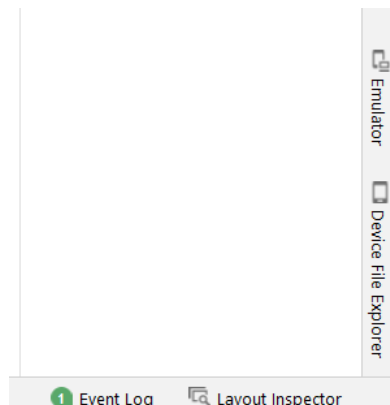
PASO III. Abrir aplicación externa

El código siguiente lanza el intent que permite abrir una app que está instalada en el dispositivo. Se tendrá que tener conocimiento del dominio para indicarlo en el nombre del componente **Línea3**.

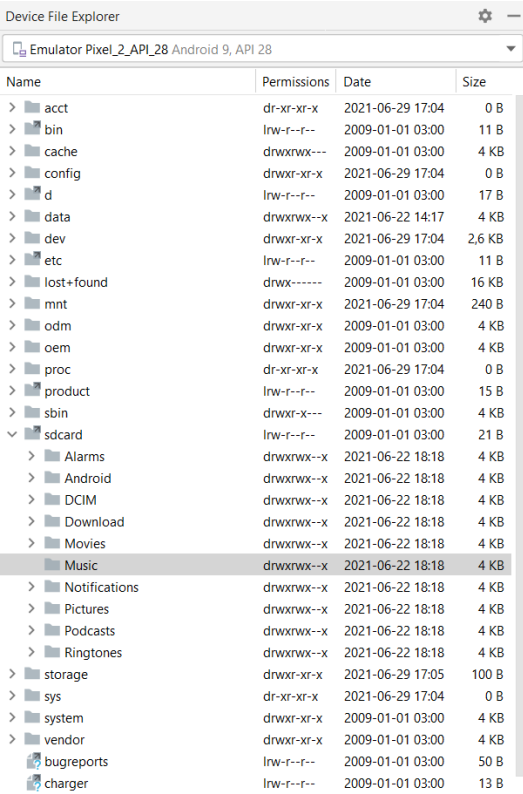
```
findViewById<Button>(R.id.abrir).setOnClickListener {  
    val intent= Intent();  
    3 intent.setComponent(ComponentName("com.ejemplos.myapplication",  
        "com.ejemplos.myapplication.MainActivity"))  
    if (intent.resolveActivity(packageManager) != null) startActivity(intent)  
}
```

PASO IV. Acceso a la tarjeta SD

Después de crear el proyecto debemos acceder al Device File Explorer, hay una pestaña en la parte inferior derecha del IDE.



Esto nos abrirá un explorador que da acceso a las carpetas de los dispositivos/emuladores que estén arrancados en ese momento. Accederemos a la carpeta Music de sdcard y con el menú contextual podremos subir o descargar elementos al/del dispositivo. Procede a añadir uno archivo .mp3 (se pasan dos como recursos del ejercicio).



Name	Permissions	Date	Size
> acct	dr-xr-xr-x	2021-06-29 17:04	0 B
> bin	lrw-r--r--	2009-01-01 03:00	11 B
> cache	drwxrwx---	2009-01-01 03:00	4 KB
> config	drwxr-xr-x	2021-06-29 17:04	0 B
> d	lrw-r--r--	2009-01-01 03:00	17 B
> data	drwxrwx--x	2021-06-22 14:17	4 KB
> dev	drwxr-xr-x	2021-06-29 17:04	2,6 KB
> etc	lrw-r--r--	2009-01-01 03:00	11 B
> lost+found	drwx-----	2009-01-01 03:00	16 KB
> mnt	drwxr-xr-x	2021-06-29 17:04	240 B
> odm	drwxr-xr-x	2009-01-01 03:00	4 KB
> oem	drwxr-xr-x	2009-01-01 03:00	4 KB
> proc	dr-xr-xr-x	2021-06-29 17:04	0 B
> product	lrw-r--r--	2009-01-01 03:00	15 B
> sbin	drwxr-x---	2009-01-01 03:00	4 KB
▼ sdcard	lrw-r--r--	2009-01-01 03:00	21 B
> Alarms	drwxrwx--x	2021-06-22 18:18	4 KB
> Android	drwxrwx--x	2021-06-22 18:18	4 KB
> DCIM	drwxrwx--x	2021-06-22 18:18	4 KB
> Download	drwxrwx--x	2021-06-22 18:18	4 KB
> Movies	drwxrwx--x	2021-06-22 18:18	4 KB
> Music	drwxrwx--x	2021-06-22 18:18	4 KB
> Notifications	drwxrwx--x	2021-06-22 18:18	4 KB
> Pictures	drwxrwx--x	2021-06-22 18:18	4 KB
> Podcasts	drwxrwx--x	2021-06-22 18:18	4 KB
> Ringtones	drwxrwx--x	2021-06-22 18:18	4 KB
> storage	drwxr-xr-x	2021-06-29 17:05	100 B
> sys	dr-xr-xr-x	2021-06-29 17:04	0 B
> system	drwxr-xr-x	2009-01-01 03:00	4 KB
> vendor	drwxr-xr-x	2009-01-01 03:00	4 KB
bugreports	lrw-r--r--	2009-01-01 03:00	50 B
charger	lrw-r--r--	2009-01-01 03:00	13 B

PASO V. Lógica Botón Escuchar canción

Teniendo en cuenta la teoría sobre el almacenamiento que usan los dispositivos Android, que nos encontramos al principio del tema. En este caso se va a acceder al contenido multimedia, por lo que se va a tener que solicitar permiso de acceso a la tarjeta externa, además de declarar un content provider. Veamos el código y pasemos a las explicaciones:

Pedir permisos

Este código es el encargado de mostrar los diálogos que permiten al usuario otorgar el permiso solicitado. Se comenta en el tema, por lo que no se va a decir nada más al respecto.

```

RESPUESTA_PERMISOS = 111
@RequiresApi(Build.VERSION_CODES.Q)
fun solicitarPermisosYEscucharCancion() {
    if (checkSelfPermission(READ_EXTERNAL_STORAGE) ==
        PackageManager.PERMISSION_DENIED) {
        requestPermissions(arrayOf(READ_EXTERNAL_STORAGE),
            RESPUESTA_PERMISOS)
    } else escucharCancion()
}

override fun onRequestPermissionsResult(
    requestCode: Int, //codigo de identificación del resultado
    permissions: Array<out String>, //array con los nombres
    //de los permisos
    grantResults: IntArray //array de 0 y -1 (permitido,
    //no permitido) en orden) {
    super.onRequestPermissionsResult(requestCode, permissions,
        grantResults)
    when (requestCode) {
        RESPUESTA_PERMISOS -> {

            if (grantResults[0] == PackageManager.PERMISSION_GRANTED)
                Toast.makeText(
                    applicationContext,
                    permissions[0] + " Permiso concedido",
                    Toast.LENGTH_SHORT
                ).show()
                escucharCancion()
            }
        }
    }
}

```

Declarar el content provider

Para las versiones de SDK de Android mayores a la 24, no se puede acceder directamente a la memoria externa que no corresponda a la aplicación, por lo que será necesario realizar los siguientes pasos:

1. Crear una carpeta xml en los recursos, `res/xml` y dentro un fichero, por ejemplo `provider_paths`. Donde añadiremos el siguiente código para referenciar al directorio raíz.

```

<?xml version="1.0" encoding="utf-8"?>
<paths>
    <external-path name="external_files" path="."/>
</paths>

```

2. Añadir en el Manifest un provider que haga referencia al recurso creado. Dentro de la etiqueta aplicación y teniendo en cuenta no olvidar los permisos de `READ_EXTERNAL_STORAGE`.

```

<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/provider_paths" />
</provider>

```

Codificar el intent

```

class MainActivity : AppCompatActivity() {

    @RequiresApi(Build.VERSION_CODES.Q)
    override fun onCreate(savedInstanceState: Bundle?) {
        ...

7       findViewById<Button>(R.id.escuchar).setOnClickListener
        {
9           solicitarPermisosYEscucharCancion()
        }
    }
12    private fun escucharCancion() {
        //Ruta de la carpeta que usa android para guardar recursos
        //de la aplicación, consta del dominio y es privada a esta
15    var directorio=getExternalFilesDir(null)?.absolutePath
        //Cogemos el inicio de la ruta, que corresponde con
        //los archivos multimedia 'storage/emulated/0/'
        directorio=directorio?.substring(0,directorio?.indexOf("Android"))
        val data = File(directorio + "music/minions.mp3")
        val uri = FileProvider.getUriForFile(
            this@MainActivity,
22        BuildConfig.APPLICATION_ID + ".provider",
            data
24        )
        val intent = Intent(Intent.ACTION_VIEW)
        intent.setDataAndType(uri, "audio/.mp3")
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
        startActivity(intent)
    }
    ...
}

```



Al pulsar el botón de escuchar canción se llamará el método `solicitarPermisosYEscucharCancion()` , **Línea 9** que a su vez llamará al método `escucharCancion()` cuando se hayan aceptado los permisos. En este último método **Línea 12**

a 24, se crea un objeto para crear la uri con la dirección de acceso usando el provider declarado antes **Línea 15**. No se debe olvidar añadir el Flags con los permisos de lectura de Uri, **Línea 22**.