Bloque 3. Ejercicio Resuelto Intent Implícito

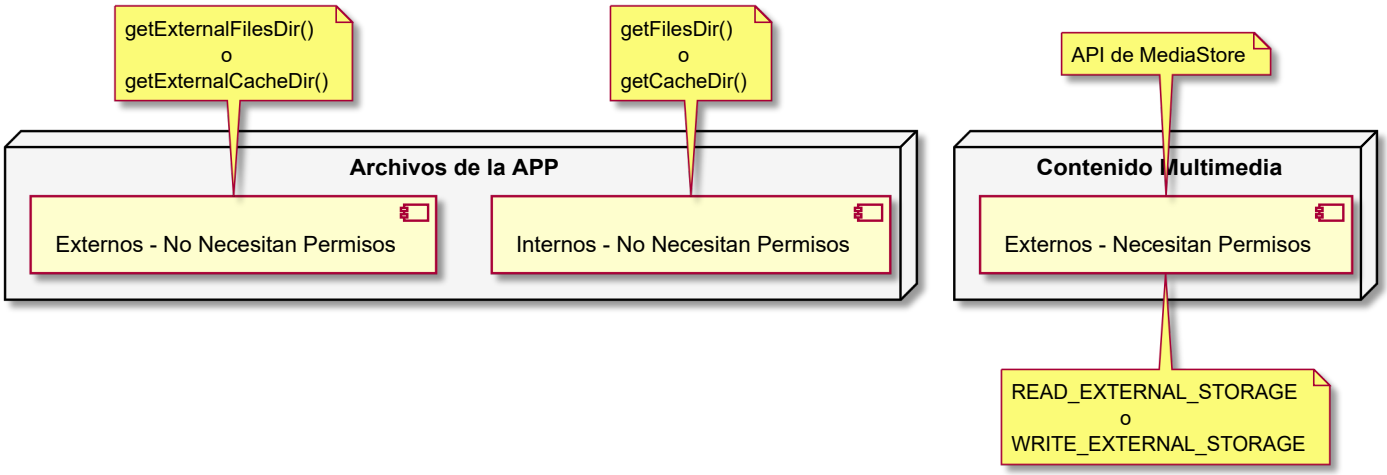
Descargar este ejercicio

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**.

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
<?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`

```kotlin
        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 **línea3** lo que hace es cargar el layout que queremos asociar con la actividad principal y asignarlo al contexto de la aplicación, **Linea4** el método findViewById busca la vista en el layout del contexto y la hincha (crear el objeto del tipo correspondiente asociado a esa vist, en este caso Button). **Linea 7 y 8** crean el Inten 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

//////////////FALTA ACABAR ESTE PUNTO DE EJERCICIO
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 del dispositivo.

PASO III. Lógica Botón Abrir

Veamos el código y pasemos a las explicaciones:

La