Tema 3.8 - Intents y Contracts

Descargar estos apuntes pdf o html

Índice

Introducción

- **Intents**
  - Documentación oficial: **Intents**
  - Documentación oficial: **Interactuando con otras Apps**
  - Vídeo Intents: **Philipp Lackner (Inglés)**
- **Permissions**
  - Documentación oficial: **Permisos**
  - Vídeo Permisos 1: **Android Developers**
  - Vídeo Permisos 2: **Android Developers**
  - Vídeo Permisos: **Philipp Lackner (Inglés)**
- **Permissions in Compose con Accompanist Library** (Experimental)
  - Vídeo: **Martin Kiperszmid (Castellano)**
  - Vídeo: **Stevdza-San (Inglés)**
  - Documentación librería: **Accompanist**

Una **Intent** es un objeto de mensajería que puedes usar para solicitar una acción de otro componente de una app. Si bien las intents facilitan la comunicación entre componentes de varias formas, existen tres casos de uso principales:

1. Iniciar una actividad.
2. Iniciar un servicio.
3. Transmitir una emisión o broadcast.

Nosotros en este tema vamos a centrarnos en el primer caso de uso que es, **Iniciar una actividad**. Como se puede deducir es una funcionalidad del API de Android que nos permite iniciar una actividad desde otra actividad incluso si la actividad pertenece a otra aplicación diferente como por ejemplos hacer una llamada telefónica, enviar un correo electrónico, abrir una página web, etc.

> 🔗 **Dercarga**: El proyecto con el código delos ejemplos de este tema en el siguiente **enlace**

Tipos de Intents

Intents Explícitos

Especifican qué aplicación las administrará, ya sea incluyendo el **nombre del paquete de la app de destino** o el nombre de clase del componente completamente calificado. Normalmente, el usuario usa una intent explícita para iniciar un componente en su propia aplicación porque conoce el nombre de clase de la actividad o el servicio que desea iniciar. Por ejemplo, puedes utilizarla para iniciar una actividad nueva en respuesta a una acción del usuario o iniciar un servicio para descargar un archivo en segundo plano.

Abriendo otra aplicación desde nuestra aplicación

Para abrir otra aplicación desde nuestra aplicación, debemos crear un objeto **Intent** y especificar el **nombre del paquete de la aplicación de destino**. Por ejemplo, si queremos abrir la aplicación de **Chrome** desde nuestra aplicación, debemos crear un objeto **Intent** y especificar el nombre del paquete de la aplicación de destino que en este caso es `com.android.chrome`. Para ello usaremos el método `setPackage()` del objeto **Intent**. Una vez creado el objeto **Intent** lo pasaremos como parámetro al método `startActivity()`. Como el método `startActivity()` es un método de la clase **Context**, necesitamos un objeto de esta clase para poder llamar al método. Para ello usaremos el `LocalContext.current` que nos proporciona **Jetpack Compose**. El código sería el siguiente...

```kotlin
// IntentExplicito.kt

fun Context.openChrome() {
    // Creamos un Intent con la acción ACTION_MAIN
    // que es abrir una actividad principal de la aplicación Chrome.
    Intent(Intent.ACTION_MAIN).also {
        it.`package` = "com.android.chrome"
        // Lanza ActivityNotFoundException si no está instalada la aplicación.
        startActivity(it)
    }
}
@Composable
fun IntentOpenChrome() {
    val context = LocalContext.current
    Box(modifier = Modifier.fillMaxSize(),
        contentAlignment = androidx.compose.ui.Alignment.Center) {
        Button(onClick = { context.openChrome() }) {
            Text(text = "Open Chrome")
        }
    }
}
```

Saber el nombre del paquete de la app de destino

Para saber el nombre del paquete de la app de destino, podemos usar la herramienta **adb.exe** (Android Debug Bridge) instalada junto al SDK de Android. Para ello debemos abrir una consola de comandos e ir a la carpeta `platform-tools` donde esté instalado el SDK para nuestro usuario. (Ej: `C:\Users\alumno\AppData\Local\Android\Sdk\platform-tools>`). Otras opción es añadir esta carpeta al **Path** de nuestro usuario para que podamos ejecutar los comandos desde cualquier ubicación, incluso desde la ventana del terminal de AndroidStudio.
Una vez tenemos acceso a la herramienta **adb.exe** seguiremos estos pasos:

1. Arrancaremos la máquina virtual de nuestro emulador de Android **desde el mismo Android Studio** o desde un terminal si sabemos el nombre de la máquina. Por ejemplo, si nuestro emulador se llama **Pixel_3a_API_33** ejecutaremos el comando:

   ```
   C:\Users\alumno\AppData\Local\Android\Sdk\tools\emulator.exe -avd Pixel_3a_API_33
   ```

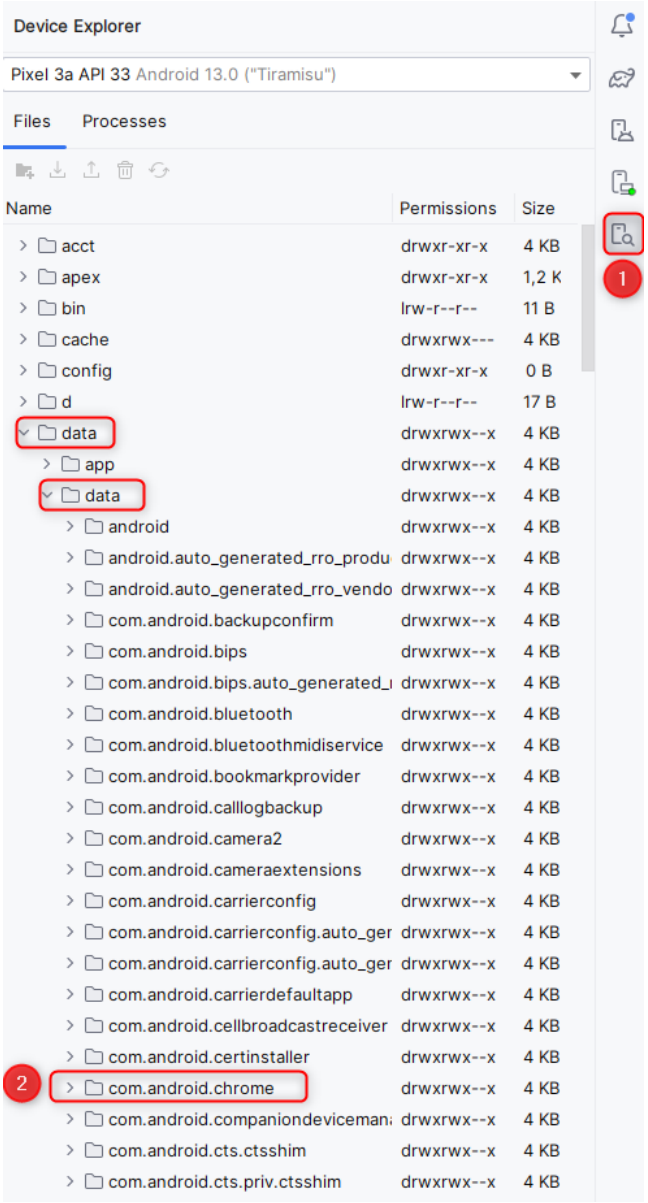
2. Con el emulador arrancado, ejecutaremos el comando `adb shell` para acceder a la consola del emulador. Por ejemplo:

```
C:\Users\alumno\AppData\Local\Android\Sdk\platform-tools>adb shell
emu64x:/ $
```

3. Ahora ejecutaremos el comando `pm list packages` para listar todos los paquetes de aplicaciones instaladas en el emulador y si queremos buscar una aplicación en concreto podemos usar el comando `grep`. Por ejemplo, si queremos saber el nombre del paquete de la app de **Chrome**, ejecutaremos el comando:

```
emu64x:/ $ pm list packages | grep chrome
package:com.android.chrome
emu64x:/ $
```

Otra forma más sencilla es a través de **Android Studio**. Una vez arrancado el emulador, abriremos la ventana **Device File Explorer** situado a la derecha como se muestra en la imagen de ejemplo y navegaremos hasta la carpeta `data/data` donde se encuentran todas las aplicaciones instaladas en el emulador. Ahora solo tenemos que buscar la carpeta de la aplicación que nos interese que se corresponde con el nombre del paquete de la aplicación.



Intents Implícitos

Son los que más interesantes para nosotros. Puesto que no siempre sabemos que aplicaciones vamos a tener instaladas.

No nombran el componente específico, pero, en cambio, declaran una acción general para realizar, lo cual permite que un componente de otra aplicación la maneje. Por ejemplo, si quieres enviar un correo electrónico, no necesitas saber qué aplicación de correo electrónico tiene el usuario instalada; simplemente debes enviar una intent implícita para que cualquier aplicación de correo electrónico pueda responderla. Es más, puedes preguntarle al sistema si existe una actividad que pueda responder a tu intent antes de iniciarla y ofrecerle la posibilidad al usuario de elegir qué aplicación usar.

Enviar un correo electrónico escogiendo la aplicación

Veamos un ejemplo de un **Intent implícito** para solicitar al sistema enviar un correo electrónico con un texto plano. Además, al Intent le pasaremos una serie de parámetros como el asunto, el texto del correo y los destinatarios. Para ello usaremos el método `putExtra()` del objeto **Intent**. El código sería el siguiente...

Aunque no es necesario, podemos indicar a Android que tipo de intents vamos a usar en nuestra aplicación al final de Manifest de la aplicación con la etiqueta `<queries>`.

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

    ...

    <queries>
        <intent>
            <action android:name="android.intent.action.SEND" />
            <data android:mimeType="text/plain" />
        </intent>
    </queries>
</manifest>
```

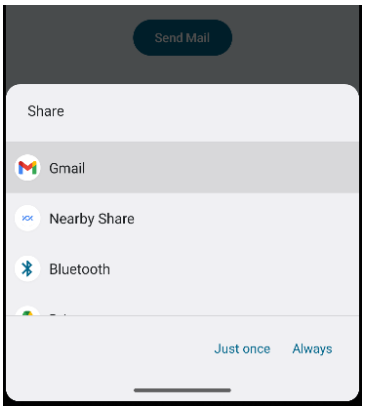
```kotlin
// IntentsImplicitos.kt

fun Context.sendMail(
    correos: Array<String>,
    asunto: String,
    texto: String,
    forzarEleccion: Boolean = false
) {
    val intent = Intent(Intent.ACTION_SEND).apply {
        type = "text/plain"
        // Añadimos los datos del correo.
        putExtra(Intent.EXTRA_EMAIL, correos)
        putExtra(Intent.EXTRA_SUBJECT, asunto)
        putExtra(Intent.EXTRA_TEXT, texto)
    }
    val chooser = if (forzarEleccion) {
        val title: String = resources.getString(R.string.enviar_correo)
        Intent.createChooser(intent, title)
    }
    else null

    if (intent.resolveActivity(packageManager) != null) {
        startActivity(chooser ?: intent)
    }
}

@Composable
fun IntentSendMail() {
    val context = LocalContext.current
    Box(modifier = Modifier.fillMaxSize(),
        contentAlignment = androidx.compose.ui.Alignment.Center) {
        Button(onClick = { context.sendMail(
            correos = arrayOf("correo@alu.edu.gva.es"),
            asunto = "Asunto del correo",
            texto = "Texto del correo"
        ) }) {
            Text(text = "Send Mail")
        }
    }
}
```

Al usar `ACTION_SEND` con esos parámetros. `Android` nos ofrecerá varias opciones para enviar el correo. Si seleccionamos `Gmail` + `Always` la siguiente vez que queramos enviar un correo, **no nos preguntará**.



Realmente el sistema **Android busca el componente apropiado** para iniciar comparando el contenido de la intent con los **filtros de intents** declarados en el archivo de manifiesto de otras aplicaciones en el dispositivo. Si la intent coincide con un filtro de intents, el sistema inicia ese componente y le entrega el objeto Intent. **Si varios filtros de intents son compatibles**, el sistema muestra un cuadro de diálogo para que el usuario pueda elegir la aplicación que se debe usar.

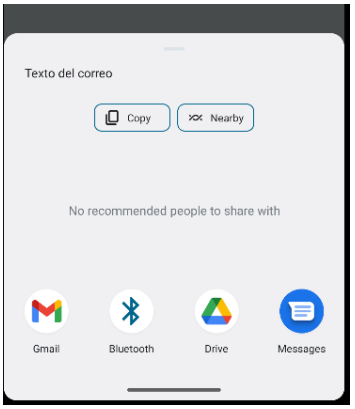
En nuestro caso, la App debe tener una actividad que acepte los parámetros que le pasamos y tener un filtro similar al siguiente:

```
<activity android:name=".ClaseQueDefineLaActividad">
    <android:exported="true"> <!-- Necesario para que otras apps puedan usarla -->
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
    </intent-filter>
</activity>
```

Puedes consultar la **documentación oficial** para más información.

> 📌 **Nota**: Si no declaras ningún filtro de intent para una actividad, esta solo se puede iniciar con un intent explícito.

Si queremos que el sistema nos pregunte siempre que aplicación queremos usar para enviar el correo independientemente de si la hemos predeterminado o no. Deberemos usar el método `createChooser()` del objeto **Intent** como se ve en el código de ejemplo.



Abrir una localización en Google Maps

Veamos algunos ejemplos más de **Intents implícitos** extraídos de la documentación oficial **aquí** o **aquí** ...

En este caso vamos a visualizar una geolocalización y la aplicación de `Maps` tiene un filtro que acepta este tipo de intents. Fíjate que en este caso hemos controlado la excepción `ActivityNotFoundException` que se lanza si no tenemos instalada la aplicación de `Maps` mostrando un mensaje temporal.

```
fun Context.buscaEnMaps(lugar: String) {
    val intent = Intent(Intent.ACTION_VIEW).apply {
        data = Uri.parse("geo:0,0?q=$lugar")
    }
    try {
        startActivity(intent)
    } catch (e: ActivityNotFoundException) {
        Toast.makeText(this, "No se puede abrir Maps", Toast.LENGTH_SHORT).show()
    }
}

@Composable
fun IntentBuscaEnMaps() {
    val context = LocalContext.current
    Button(onClick = {
        context.buscaEnMaps("I.E.S Doctor Balmis, +Alicante")
    }) {
        Text(text = "Ver Balmis en Maps")
    }
}
```