Programación multimedia y dispositivos móviles

UT-7. Intents - Explícitas

https://developer.android.com/guide/

¿Qué son los Intents de Android?

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:

- **Iniciar una actividad**: para iniciar una nueva instancia de una Activity deberás pasar un objeto Intent. Este Intent describe la actividad que se debe iniciar y contiene los datos necesarios para ello.
- **Iniciar un servicio**: Un Service es un componente que realiza operaciones en segundo plano sin una interfaz de usuario.
- **Transmitir una emisión**: Una emisión es un aviso que cualquier aplicación puede recibir. El sistema transmite varias emisiones de eventos, como cuando se inicia el sistema o comienza a cargarse el dispositivo. Puedes transmitir una emisión a otras apps pasando una Intent.

¿Qué son los Intents de Android?

Los Intents permiten intercambiar datos entre aplicaciones o componentes de aplicaciones, como por ejemplo las actividades. También pueden ser usados para iniciar actividades o servicios. Otra posible aplicación de los Intents es solicitar al sistema que se realice una determinada acción con ciertos datos; el propio Android se encargará de buscar la aplicación más cualificada para realizar el trabajo.

Por lo tanto, los Intents se convierten en un mecanismo para la transmisión de mensajes que puede ser utilizado tanto en el seno de una única aplicación como para comunicar aplicaciones entre sí.

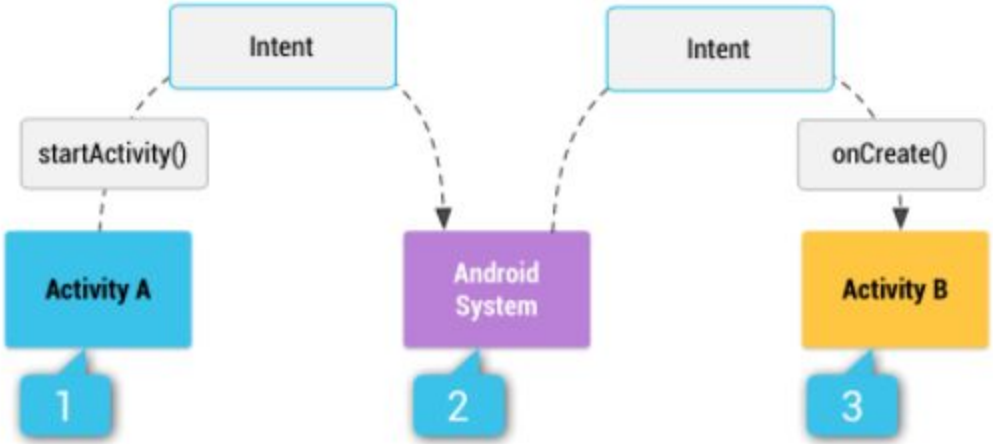
El uso de Intents es un principio fundamental en el desarrollo de aplicaciones para Android. Permite el desacoplamiento de componentes de la aplicación, de tal forma que cualquiera de ellos pueda ser sustituido fácilmente. También permite de manera simple extender la funcionalidad de nuestras aplicaciones, reutilizando actividades presentes en aplicaciones de terceros o incluso aplicaciones nativas de Android.

Tipos de Intents

Existen dos tipos de intents:

- Las **intents explícitas** 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.
- Las **intents implícitas** 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 deseas mostrar al usuario una ubicación en un mapa, puedes usar una intent implícita para solicitar que otra aplicación apta muestre una ubicación específica en un mapa.

Uso de intents

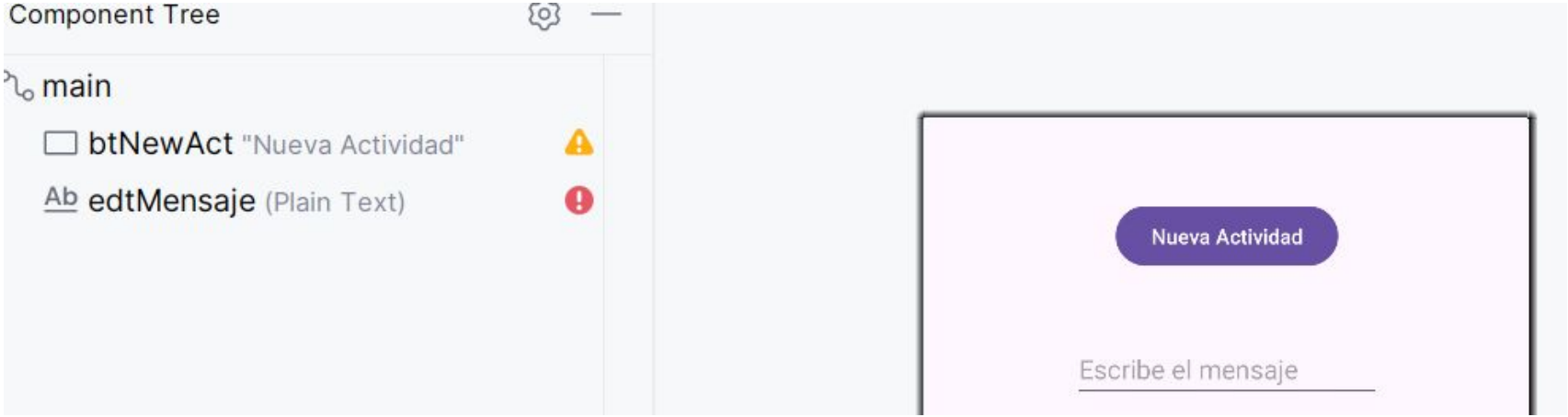


Uso de intents

1. La actividad A crea una Intent con una descripción de acción y la pasa a startActivity()
2. Según sea la intent:
    a. Cuando el objeto Intent nombra un componente de actividad específico de forma explícita, el sistema inicia ese componente al instante.
    b. Cuando usas una intent implícita, 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.
3. El sistema inicia la actividad coincidente (actividad B) invocando su método onCreate() y pasándolo a la Intent.

Mi primera aplicación con Intent

Creamos un nuevo proyecto en el que en la actividad principal incluiremos un EditText para poder introducir un texto y un botón desde el que abriremos la nueva actividad



Mi primera aplicación con Intent

Sobre la carpeta del proyecto, pulsamos botón derecho y creamos una nueva actividad (actividad en blanco). En mi caso la he llamado “SegundaActivity”. En esta 2ª activity vamos a incluir un texto para recibir el mensaje de la primera actividad y un botón para cerrar esta actividad.

Mi primera aplicación con Intent

El objetivo es bastante sencillo, desde la actividad principal al pulsar al botón de “Nueva Actividad” se recogerá el mensaje que se haya escrito en el EditText y se enviará a la 2ª actividad para que se visualice. Veamos cómo queda el código en la actividad principal:

```
val myEdt = findViewById<EditText>(R.id.edtMensaje)
val myBt = findViewById<Button>(R.id.btNewAct)

myBt.setOnClickListener{ it: View!
    val myIntent = Intent( packageContext: this, SeconActivity::class.java)
    myIntent.putExtra( name: "mensaje", myEdt.text.toString())
    startActivity(myIntent)
}
```

Mi primera aplicación con Intent

Más allá de del onClick del botón, y del casteo y captura del texto del EditText lo que tenemos es:

- Definimos una intent dónde asignamos 2 valores: el contexto (mi actividad/this) y la actividad que queremos abrir (que en mi caso es la segunda actividad). Esta es una intent explícita porque indica claramente que actividad queremos abrir.
- Al objeto Intent le asignamos el valor del texto recogido. Esto lo hacemos con el método putExtra que podéis comprobar al indicarlo en el código que tiene claramente polimorfismo o sobrecarga. Conceptualmente en este método definimos un “extra” con la clave que queramos y luego le asignamos el valor que corresponda. En la actividad que recibe la intent tocará hacer uso de este dato.
  - Los “Extras” viajan en un objeto Bundel dentro del Intent
- Por último simplemente iniciamos la actividad indicando la Intent que queremos lanzar.

Mi primera aplicación con Intent

Ahora vemos como queda el código de la segunda actividad:

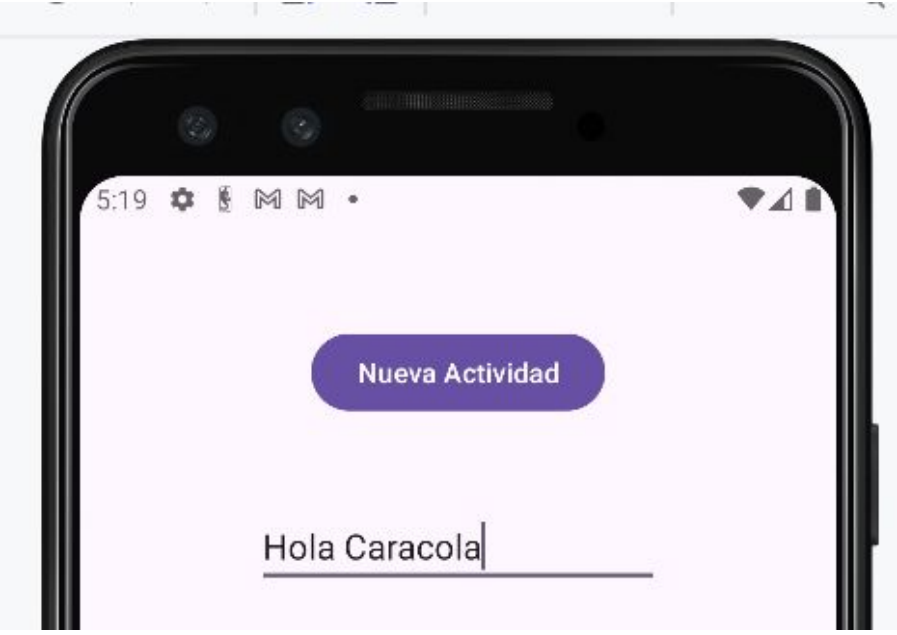
```
val misExtras :Bundle = getIntent().extras!!
val mensaje = misExtras!!.getString( key: "mensaje")

findViewById<TextView>(R.id.tvMensaje).text = mensaje

findViewById<Button>(R.id.btSalir).setOnClickListener{ it: View!
    this.finish()
}
```

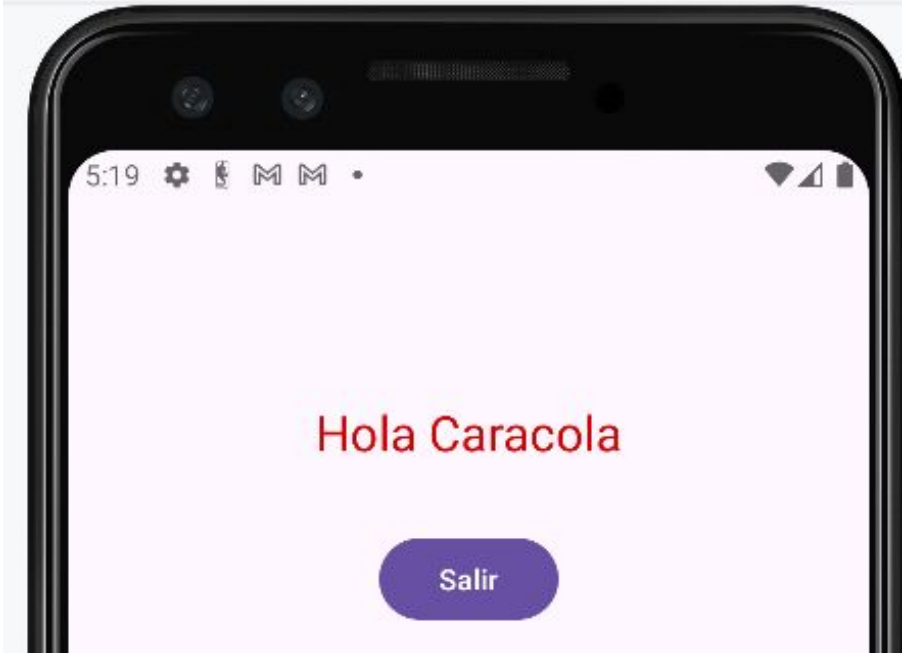
Mi primera aplicación con Intent

Además del método salir para cerrar la actividad lo que tenemos es:

- Instanciamos una variable. Esta es de tipo Bundel y recojemos los “Extras” que vienen dentro del Intent .
- Recuperamos en una variable el “Mensaje” con la información enviada por la primera actividad y lo asignamos al TextView.

Mi primera aplicación con Intent





Objetos Parcelable y Bundle

Los objetos Parcelable y Bundle están diseñados para usarse entre actividades con intents y para almacenar el estado transitorio en los cambios de configuración.

Android recomienda que uses la clase Bundle para el envío de primitivas que el SO conozca en objetos Intent. La clase Bundle está muy optimizada para asociar y desasociar el uso de parcelas.

En algunos casos, es posible que necesites un mecanismo para enviar objetos compuestos o complejos entre actividades. En esos casos, la clase personalizada debe implementar un objeto Parcelable.

https://developer.android.com/guide/components/activities/parcelables-and-bundles?hl=es

Parcelable vs Serializable

La serialización en Android es bastante lenta y no se recomienda para el envío de objetos entre Activities. Para suplir esta carencia, Android creo el objeto Parcelable.

- Cuando se usa memoria, Parcelable tiene un rendimiento más alto que Serializable.
- Serializable generará una gran cantidad de variables temporales durante la serialización, lo que causará un Garbage Collecting frecuente.
- Además un objeto Parcelable puede ser enviado mediante un intent en Android.

Cómo obtener un resultado de una actividad

Una actividad iniciada por medio de la función startActivity es independiente de su actividad padre y por lo tanto no proporcionará ningún tipo de información a ésta cuando finalice.

Sin embargo podemos tener la necesidad de que la actividad inicial reciba información de la actividad abierta y en base a esa respuesta tener comportamientos distintos.

En Android otra posibilidad es iniciar una actividad como una subactividad que esté conectada a su actividad padre. Una subactividad que finalice provocará siempre la activación de un evento en su actividad padre.

Cómo obtener un resultado de una actividad

Para abrir una actividad esperando un resultado se utiliza la clase ActivityResultLauncher dónde tenemos el método launch() para abrir la 2ª actividad (pasando una intent igual que cuando no se espera resultado), Cambian los métodos pero poco más para abrir la 2ª actividad.

Para capturar los datos enviados en la 2ª actividad se hace exactamente igual que cuando la actividad se ha abierto con el método starActivity.

Cómo obtener un resultado de una actividad

Sin embargo, dado que la 1ª actividad está esperando información de la 2ª actividad, en esta si tenemos que hacer varios ajustes para que la actividad inicial sepa que ha pasado.

Cuando la subactividad esté preparada para terminar, llamaremos a setResult antes de la llamada finish para devolver un resultado a la actividad padre.

El método setResult requiere dos parámetros: el código de resultado y el propio resultado, que se representará mediante un Intent. Generalmente el código de resultado tendrá el valor Activity.RESULT_OK o Activity.RESULT_CANCELED, aunque podríamos utilizar nuestros propios códigos de resultado, ya que este primer parámetro puede tomar como valor cualquier número entero.

Además, de forma similar a cuando pasamos un intent al abrir una actividad, podemos pasar un intent "de vuelta" con la información que queramos.

Mi segunda aplicación con Intent

Vamos a seguir con el mismo proyecto para abrir una subactividad desde una actividad y controlar el resultado de vuelta.

Mi segunda aplicación con Intent

Vamos a seguir con el mismo proyecto para abrir una subactividad desde una actividad y controlar el resultado de vuelta.

Mi segunda aplicación con Intent

En la clase de la actividad principal lo primero que vamos a hacer es declarar una variable de la clase que necesitamos:

```
lateinit var myActivityResultLauncher: ActivityResultLauncher<Intent>
```

Ahora vemos y comentamos cómo debe quedar el código en la actividad principal para abrir y recibir respuesta de la subactividad (dentro del onCreate)

Mi segunda aplicación con Intent

```
myActivityResultLauncher = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult()) {
    result: ActivityResult? ->
        if (result!!.resultCode == Activity.RESULT_OK) {
            // Acciones si todo ha ido OK
            val miIntentResultado = result.data
            findViewById<TextView>(R.id.tvMensBack).text = miIntentResultado!!.extras!!.getString( key: "mensajeBack")
        }else{
            // Acciones si todo ha ido KO
            Toast.makeText( context: this, text: "Ha fallado la vuelta", Toast.LENGTH_LONG).show()
        }
}
```

Recordemos, esto es una función lambda

que recibe un parámetro

Mi segunda aplicación con Intent

En la documentación explica, más o menos, lo que hacen estos métodos:

https://developer.android.com/training/basics/intents/result?hl=es-419#kotlin

A mi me cuesta entenderlo, pero más o menos tenemos:

- ActivityResultContracts.StartActivityForResult(): establece un “contrato” para que la nueva actividad que abrimos no sea independiente, sino que sepa que está “unida” a esta.
- Las llaves: recordemos que esas llaves en Kotlin es en realidad una función que es el último parámetro de la función registerForActivityResult. Es por tanto un callback, una función que será a la que se llame cuando se retorne. La función está expresa en lambda y result es el parámetro de entrada que nos dicen que es de tipo “ActivityResult”)

Mi segunda aplicación con Intent

El método para abrir la 2ª actividad apenas tendría cambios

```
myBt.setOnClickListener{ it: View!
    val myIntent = Intent( packageContext: this, SeconActivity::class.java)
    myIntent.putExtra( name: "mensaje", myEdt.text.toString())
    //startActivity(myIntent)
    myActivityResultLauncher.launch(myIntent)
}
```

Mi segunda aplicación con Intent

Por otra en la clase de la 2ª actividad, el método onCreate no tenemos que hacer ningún cambio.

Codificamos un método para el botón de de salir:

- ❏ Instanciamos un objeto intent
- ❏ Recogemos el contenido del EditText, y lo asignamos al objeto intent
- ❏ Asignamos el objeto intent y la salida OK a la devolución con el método setResult
- ❏ Cerramos la actividad

Mi segunda aplicación con Intent

```
findViewById<Button>(R.id.btSalir).setOnClickListener{ it: View!
    val myResultado = Intent()
    myResultado.putExtra( name: "mensajeBack", findViewById<EditText>(R.id.edtMensBack).text.toString())
    setResult(RESULT_OK, myResultado)

    this.finish()
}
```

Mi segunda aplicación con Intent

Documentación

https://developer.android.com/guide/components/intents-filters?hl=es-419

http://www.jtech.ua.es/dadm/restringido/android/sesion02-apuntes.html

https://code.tutsplus.com/es/tutorials/what-are-android-intents--cms-29335

https://programmerclick.com/article/3037483926/

https://es.stackoverflow.com/questions/20236/diferenecia-de-serializable-y-parcelable

https://developer.android.com/training/basics/intents/result?hl=es-419#launch

https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es

Documentación

https://stackoverflow.com/questions/62671106/onactivityresult-method-is-deprecated-what-is-the-alternative

https://proandroiddev.com/is-onactivityresult-deprecated-in-activity-results-api-lets-deep-dive-into-it-302d5cf6edd