Intents - II

Intents Explícitos [Kotlin]

La forma de operar con los **Intent Explícitos** en Kotlin es fundamentalmente similar a cómo se hace en Java. Disponemos tanto de los métodos **startActivity ()** como **startActivityForResult ()**, con las mismas reglas. Por tanto, para el ejemplo de la espera de respuesta de una *actividad* podríamos hacer:

```
private val SUBACTIVITY_1 = 1
```

Y después, simplemente realizamos el *intent*.

```
buttonToAc2.setOnClickListener{ it: View!
    fun onClick(v: View?) {
        val intent = Intent(applicationContext, Activity2::class.java)
        intent.putExtra( name: "Veces", finalVeces)
        startActivityForResult(intent, SUBACTIVITY_1)
    }
}
```

Nota: el método está deprecado, pero nos da igual. Estamos aprendiendo.

De la misma forma que en Java, cuando queramos terminar la *actividad*, podremos indicar el estado en el que la *actividad* ha finalizado (correctamente, cancelada, etc.) y pasar parámetros a la *actividad* principal mediante *intents*, tal y como se ve a continuación.

```
buttonOk.setOnClickListener { it: View!
    val intent = Intent(applicationContext, MainActivity::class.java)
    intent.putExtra( name: "Veces", veces)
    setResult(RESULT_OK, intent)
    finish()
}

buttonCancel.setOnClickListener{ it: View!
    setResult(RESULT_CANCELED)
    finish()
}
```

Por último, recogemos el resultado del **startActivityForResult ()** mediante el **evento**:

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    when (requestCode) {
        SUBACTIVITY_1 -> if (resultCode == RESULT_OK) {
            // Todo ha ido bien
            val veces = data?.getStringExtra( name: "Veces")
        } else if (resultCode == RESULT_CANCELED) {
            // Tratamos el error
        }
        SUBACTIVITY_2 -> if (resultCode == RESULT_OK) {
            // ...
        }
    }
}
```

Práctica 30

[Kotlin] Realiza la práctica **07 – Intents Explícitos [Kotlin]**.

Práctica 31

[Kotlin] Realiza la práctica **08 – Intents Explícitos II [Kotlin]**.

Intents Implícitos [Java]

Como ya se ha comentado, existe otra forma de iniciar una *actividad*: un intent **implícito**. En este caso, se solicita que un componente anónimo de una app sin determinar se encargue de realizar el trabajo. Esto es similar a colocar un aviso en un tablón de anuncios. Escribes qué quieres hacer y la información necesaria para hacerlo, y *alguien responderá* a ese anuncio. Date cuente que no sabes con antelación quién se encargará de responder al aviso.

Para poder hacer un *intent implícito* necesitas indicar la **acción** que quieres realizar y pasarle la **URI** con los datos que necesita. Si es necesario, puedes enviar más información en el *intent* como un **extra**.

Existen muchas **acciones** diferentes y están catalogadas. Es posible preparar una app para que responda a una o más acciones; además, puede que varias apps sean capaces de responder a la misma acción. Por norma general, es el sistema el que elige qué app es la más adecuada para responder a una acción, excepto si queremos darle la opción al usuario de elegir qué app quiere utilizar mediante un **chooser**. Las acciones más comunes son:

Estados	
ACTION_ANSWER	Abre la app que maneja las llamadas de teléfono entrantes
ACTION_CALL	Muestra el marcador de teléfonos e inmediatamente inicia una llamada al número indicado por la URI.
ACTION_DELETE	Inicia una actividad para eliminar los datos referenciados por la URI del intent
ACTION_DIAL	Muestra el marcador de teléfonos con un número premarcado que es el pasado por la URI
ACTION_EDIT	Inicia una actividad para editar los datos referenciados por la URI del intent
ACTION_INSERT	Inicia una actividad para insertar nuevos datos en el Cursor referenciados por la URI del intent
ACTION_PICK	Lanza una actividad para escoger un elemento del proveedor de contenidos especificado por la URI del intent
ACTION_SEARCH	Inicia una actividad de búsqueda
ACTION_SENDTO	Inicia una actividad para enviar un mensaje al contacto especificado por la URI del intent.
ACTION_SEND	Inicia una actividad para enviar una serie de datos mediante el intent.
ACTION_VIEW	Es la acción más común. Solicita que los datos referenciados por URI del intent sean mostrados de la forma más adecuada posible. q

En cualquier caso, tocará **investigar por internet** exactamente qué acción y cómo se pasan parámetros para cada cosa que queramos hacer. No es lo mismo solicitar el uso de la cámara de fotos que abrir un vídeo de música en YouTube en nuestro navegador predeterminado.

Como ya hemos comentado antes, a veces hay varias apps capaces de responder a la misma acción. Si nos limitamos a realizar el *intent* sin más, es el sistema quien decide la app que responde al *intent*. En cambio, si queremos que el usuario decida, podemos usar un **chooser**.

En el siguiente ejemplo solicitamos que 'alguien' nos muestre una dirección web. Si iniciásemos el *intent*, el sistema abriría el navegador predeterminado. En cambio, al crear un *chooser* a partir del *intent*, el sistema nos mostraría un pop-up con todas las apps capaces de responder a la acción ACTION_VIEW y nos dejaría escoger una.

```
Intent intent = new Intent();
intent.setAction( Intent.ACTION_VIEW );
String URL = webText.getText().toString(); // Do not forget the https:
URL = URL.contains("http://")? URL :  "https://" + URL;
intent.setData( Uri.parse( URL ) );

Intent chooser = intent.createChooser(intent, getText( R.string.txt_intent_web ));

if ((chooser.resolveActivity(getPackageManager())) != null) {
    startActivity(chooser);
    Toast.makeText(this, getText( R.string.txt_ok ), Toast.LENGTH_LONG).show();
} else {
    Toast.makeText(this, getText( R.string.txt_error_2 ), Toast.LENGTH_LONG).show();
}
```

Al igual que un *intent explícito*, en los *intent implícitos* también es posible tratar una respuesta enviada por la actividad que has iniciado. Para ello, simplemente hay que utilizar el método **startActivityForResult ()** y tratar el evento de la misma forma.

Intents Implícitos [Kotlin]

Básicamente, el funcionamiento es el mismo. A modo de ejemplo:

```
val intent = Intent ( Intent.ACTION_VIEW, Uri.parse( URL ) );
val chooser = intent.createChooser(intent, getText( R.string.txt_intent_web ));

if ((chooser.resolveActivity(getPackageManager())) != null) {
        startActivity(chooser);
        Toast.makeText(this, getText( R.string.txt_ok ), Toast.LENGTH_LONG).show();
    } else {
        Toast.makeText(this, getText( R.string.txt_error_2 ), Toast.LENGTH_LONG).show();
    }
}
```

Práctica 31

[Java] Realiza la práctica **09 – Intents Implícitos [Java]**.

Práctica 32

[Java] Realiza la práctica **09 – Intents Implícitos [Kotlin]**.

Práctica 33

[Java] Realiza una app que sea capaz de acceder a la agenda del móvil y cargue una lista con los nombres y números de teléfono. Al pulsar a un número, la app deberá de mostrar el marcador de teléfonos con el número premarcado. (No llames directamente al número).

Práctica 34

[Kotlin] Realiza una app que sea capaz de acceder a la agenda del móvil y cargue una lista con los nombres y números de teléfono. Al pulsar a un número, la app deberá de mostrar el marcador de teléfonos con el número premarcado. (No llames directamente al número).

EXTRA

[Java/Kotlin] Crea una app capaz de reproducir vídeo captado por la cámara del móvil en un VideoView. Este ejercicio propuesto es para que investigues un rato si ya has terminado otras tareas. No es complicado de hacer porque el móvil emulado por el Android tiene cámara y una 'sala virtual' que puede captar, pero hacer que todo eso funcione es complicado. Que te diviertas.