Kotlin

MINI APUNTES ANDROID 1

Programación Multimedia y Dispositivos Móviles

3 DE NOVIEMBRE DE 2024

ERIK AMO TOQUERO

Contenido

Componentes de una App de Android

Activity – Actividad

Derivan de la clase Activity y ofrecen una pantalla para que el usuario interactúe. Cada Activity va asociada a una ventana, la cual puede ser del mismo tamaño de la pantalla o ser un contenido flotante.

Una app consta de varias activities, con una principal. Se puede navegar entre activities. Siguiendo la pila LIFO para detener las anteriores.

View – Vista

Derivan de View y suele definir componentes con los que los usuarios interactúan. Son parecidos a los de JavaSwing y JavaFX

Layout – Diseño

Derivan de ViewGroup. Son contenedores que contiene vistas y otros ViewGroups. Se pueden definir en ficheros de XML o en instancias durante la ejecución.

Service – Servicio

Derivan de Service y permiten realizar operaciones en segundo plano. Pueden definir aplicaciones en primer plano (aplicaciones de música) y pueden estar en ejecución sin acciones del usuario. Son como los servicios de Windows.

Content Provider – Proveedor de Contenido

Es un mecanismo estándar que permite acceder a datos de otras aplicaciones, como la lista de contactos.

Intent – Intención

Es la voluntad de realizar alguna opción en el teléfono, como realizar una llamada o llamar a una web. Se utiliza para lanzar un servicio, comunicarnos y lanzar una actividad distinta

Broadcast Receiver – Receptor de anuncios

Permite actuar con eventos externos

Fragment – Fragmento

Aparecen en la UI de tablets, y son bloques independientes en las actividades. Un ejemplo es la aplicación de Ajustes, dividida en 2 fragmentos.

Ciclo de vida de una actividad

Las aplicaciones Android pasan por 3 estados:

- **En ejecución:** La aplicación está en primer plano
- **Pausada:** Hay otra aplicación en primer plano, pero esta es visible y se mantiene en memoria.
- **Detenida:** La actividad pasa a segundo plano. El sistema la puede borrar cuando quiera.

La ejecución Android sigue el siguiente orden:

onCreate()

El método onCreate() se invoca de forma implícita (por el sistema) o explícita (por el usuario). Se instancia el objeto de la aplicación.

onStart()

El método onStart() inicializa la instancia de la aplicación y la pone en primer plano

onResume()

Con el paso del tiempo de vida de la aplicación, es muy posible que vaya a perder la prioridad en pantalla y deje de estar en primer plano. El método onResume() permite ejecutar código cuando la aplicación recupere el primer plano.

Una vez pasa por esos tres métodos, está en primer plano.

onPause()

Si la actividad va a pasar a segundo plano por cualquier motivo (Una llamada de teléfono, por ejemplo), se invoca esta función para poder guardar datos. Puede volver a onResume() si vuelve al primer plano

onStop()

Una vez que la aplicación deja de ser visible, se ejecuta el método onStop(). De aquí puede pasar dos cosas:

1. El proceso de la aplicación muera para liberar memoria: Se re ejecutará desde onCreate()
2. El usuario entra de nuevo a la actividad sin que este proceso haya muerto: implementa el método *onRestart()* y vuelve a onStart()

onDestroy()

Cuando la aplicación es destruida por el sistema o el usuario la finaliza, se accede al método onDestroy(), donde se termina la aplicación

Apertura de Actividades desde otras Actividades

Se utilizan los Intents para intentar abrir una nueva actividad en el contexto actual. Una vez se crea el Intent, se utiliza el método startActivity.

```
var mi_intencion=Intent(this,NombreNuevaActivity::class.java)
startActivity(mi_intencion)
```

- **this** define el contexto, la pantalla actual en este caso
- El siguiente valor que busca es el tipo de la clase que va a abrir el programa. Con él busca en los manifiestos de Android a qué actividad se refiere.

Actividades en Manifest

Las actividades en Android Manifest se ven rodeadas de los tags <manifest/>. Todas las actividades tienen 2 valores: name y exported. Name hace referencia a la clase, y Exported es una booleana que controla si otras aplicaciones externas pueden abrir la actividad. **Es importante que exported sea true en la pantalla principal, puesto que el sistema es el que abre la aplicación.**

```
<activity
    android:name=".TerceraActividad"
    android:exported="false" />
<activity
    android:name=".MainActivity"
    android:exported="true" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".SegundaActividad"
    android:exported="false" >
</activity>
```

Crear la Actividad – Layouts

Al iniciar la nueva actividad tiene que tener el layout definido en el método onCreate(). Se pueden tener distintos layouts, preferiblemente uno por cada actividad. Dentro de Android Studio, está la sección de crear una nueva Activity, con la opción de crear y asignar un layout.

Layout es un archivo xml en la carpeta layout, que se parsea a una clase “estática” llamada R. La clase R contiene la definición de todos los recursos de la aplicación

Una vez se ha creado el layout, en el método onCreate hay que llamar a la función setContentView(id del layout)

```
setContentView(R.layout.actividad)
```

Librerías de Compatibilidad

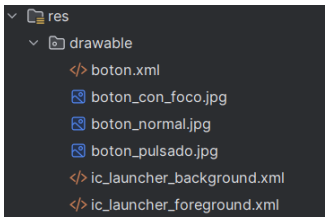
Son librerías que buscan dar compatibilidad entre aplicaciones nuevas y dispositivos antiguos. Actualmente, estas librerías se han sustituido por la librería AndroidX: Una librería preinstalada que incluye todas las clases que se pueden utilizar en versiones desactualizadas de Android.

Algunas librerías son AppCompatActivity (base de las actividades), Views y Room (Acceso a Datos).

Recursos

Los recursos son una serie de archivos y contenido estático al que la app tiene acceso. Estos recursos son externalizados y la aplicación guarda un código único de ellos, almacenado en la clase R.

Los recursos se deben guardar en el subdirectorio de res que corresponda:



Contenido de la Carpeta res

res/ tiene varias subcarpetas para diferenciar qué guarda cada una:

- **Drawable:** Contenido que se puede mostrar (Imágenes, iconos y vectores)
- **Layout:** Los layouts de la aplicación
- **Mipmap:** Su uso es parecido al de drawable, cambiando en que los recursos de mipmap tienen copias del mismo recurso para las distintas resoluciones del dispositivo
- **Values:** Guarda valores estáticos e inmutables de la aplicación. Pueden ser desde Strings hasta arrays
- **Xml:** Guarda archivos xml

Acceso a Recursos

Se puede acceder a los recursos de varias formas:

- Desde código: Hace uso de la clase R con sintaxis R.tipo.nombre

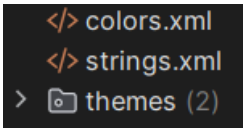
```
var imView:ImageView = findViewById(R.id.imageView);
imView.setImageResource(R.drawable.boton)
```

- Desde XML: Hace referencia al archivo con la sintaxis @tipo/nombre. Si aparece en la sintaxis @+tipo/nombre, significa que está añadiendo el recurso

```
android:label="@string/app_name"
android:id="@+id/imageView"
```

Recurso “values”

Si entramos en “values”, encontraremos algo tal que así:



Cada archivo xml sirve para separar los contenidos entre ellos.

Dentro de cada archivo, aparecen los valores que se pueden usar en el código más tarde. Pueden ser de varios tipos:

- Integer
- String
- Bool
- Item (Tipo de contenido de arrays)
- Integer-array (Array de Enteros)
- String-array (Array de Strings)
- Array: (Array tipado)

Los arrays en values se rellenan de la siguiente forma:

<tipoarray name=”array”>

<item></item>

...

</tipoarray>

```
<integer-array name="nombre">
    <item>12</item>
    <item>2</item>
    <item>3</item>
</integer-array>
var array:IntArray = resources.getIntArray(R.array.nombre);
```

Acceso a la API desde Android Studio

Ante cualquier duda, se puede acceder a la referencia de la clase que estamos utilizando Ctrl + Click derecho sobre la clase o Ctl + Q para ver la documentación.

Toda API de Android Studio es visible de forma local desde el IDE