UA02. FUNCIONAMIENTO DE APLICACIONES

2.1. Ciclo de vida de una aplicación

Para el desarrollo nativo de aplicaciones en Android no basta con ser un buen conocedor del código (básicamente Java) y tener buena pericia en fundamentos de programación. La peculiaridad de los dispositivos móviles, la variabilidad que entre estos existen (como ya se estudió en el capítulo anterior) y, sobre todo, la multiplicidad de recursos que son capaces de aportar estos dispositivos, hacen necesario que antes de iniciar las tareas de codificación se conozcan los fundamentos y la estructura de componentes en los que se basa una aplicación en Android.

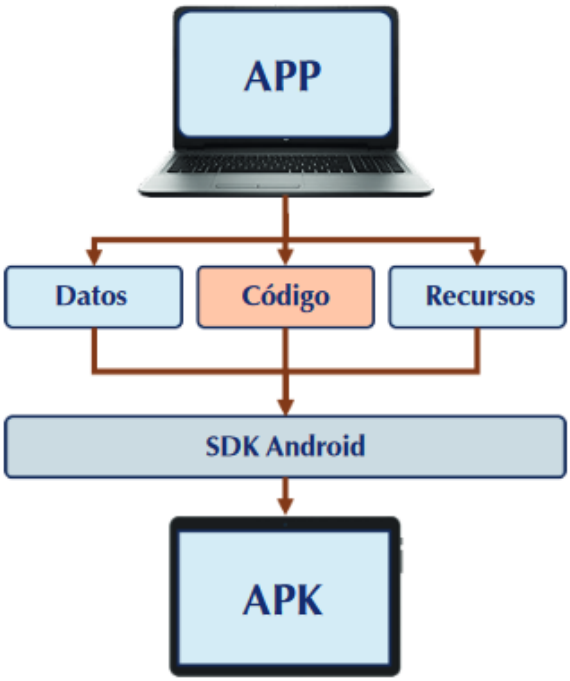
Además, el correcto manejo de algunos de estos componentes (Android Manifest, clase R, permisos...) es importante si se desea que las aplicaciones desarrolladas tengan alguna proyección de mayor alcance que el límite que pueda imponer el propio equipo informático y las herramientas de desarrollo.

Fundamentos

Android es un sistema operativo multiusuario basado en Linux, donde cada aplicación es considerada propiedad de un usuario distinto. El sistema asigna un identificador de usuario diferente a cada aplicación, por lo que los archivos incluidos en cada una tendrán permisos solo para ese usuario y solo él podrá tener acceso a ellos.

Esto hace que cada aplicación tenga su propio sistema de seguridad, con usuarios diferentes para cada una de ellas, permisos propios para cada usuario con máquinas virtuales propias y un proceso de Linux propio. De tal manera que Android utiliza el

principio de menor privilegio, dando los permisos justos a cada aplicación. Todo ello lo convierte en un sistema operativo seguro.



No obstante, una aplicación puede solicitar permisos para acceder a datos (agenda), recursos (SD, Bluetooth) o funcionalidades (SMS, cámara) del dispositivo. Estos permisos asociados a la aplicación se conceden en la instalación de la misma o en el momento de usarlos (para versiones más modernas de Android), y los otorga el usuario.

Las aplicaciones Android están escritas en Java, un lenguaje de programación orientado a objetos, apoyado en determinados aspectos por XML. El SDK (kit de desarrollo de software) de Android compila el código, datos y recursos, incluyéndolo todo en un fichero APK.

A diferencia de otros sistemas operativos, en Android cada aplicación es un proceso separado, el usuario no decide cuándo se finaliza una aplicación, esta decisión reside en la gestión de la pila de aplicaciones del propio sistema Android, en función de las necesidades de memoria en cada momento.

Del mismo modo, cuando el usuario vuelve a solicitar la aplicación, el sistema se encarga de crear el proceso para mostrar la aplicación en pantalla.

Dependiendo del estado en el cual se encuentre el proceso, el sistema asigna una prioridad que determina la probabilidad de que la aplicación sea cerrada. Podemos verlo en la siguiente tabla:

Prioridad	Estado de proceso	Estado de la actividad
Bajo o nula	Primer Plano, tiene el foco	Created Started Resumed

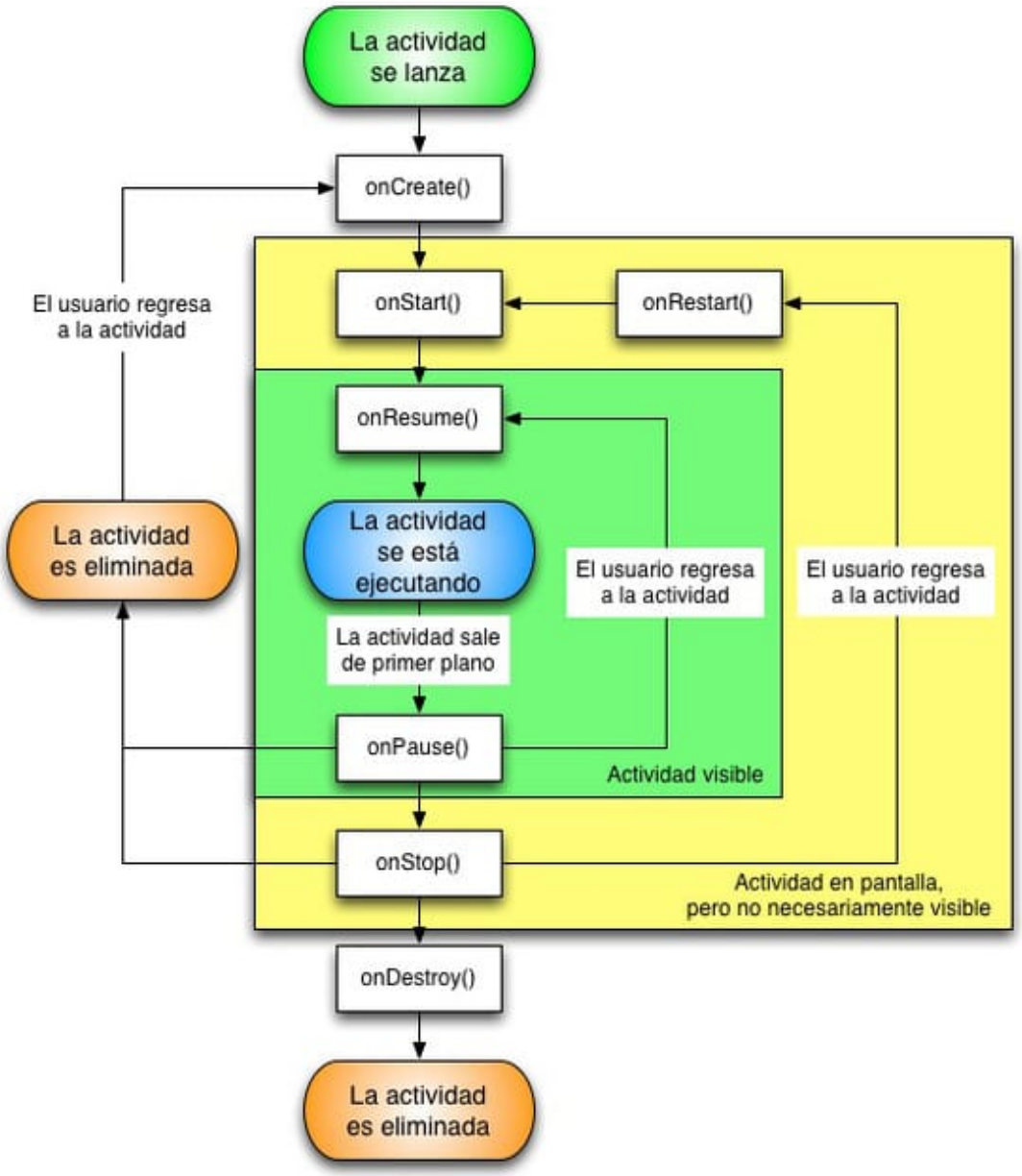
Media	Segundo plano, perdió el foco	Paused
Alta	Segundo plano, no visible	Stoped Destroyed

Vemos que el sistema no cerrará nunca la aplicación que tengan el foco, es decir, aquella aplicación que se está mostrando en la pantalla. Si la aplicación pierde el foco debido a que el usuario cambia de aplicación o está leyendo un mensaje o atendiendo una llamada, pasa a segundo plano y puede ser cerrada por el sistema si este necesita los recursos.

Es vital entender el funcionamiento del ciclo de vida de las aplicaciones para desarrollar aplicaciones robustas, ya que tenemos que manejar el ciclo de vida para determinar en qué momento recuperamos los datos de la aplicación.

En la tabla hemos incluido el estado de la actividad, puesto que el proceso de la aplicación está fuertemente vinculado al estado de la actividad, de modo que, para manejar el ciclo de vida desde nuestra aplicación, tenemos disponibles un conjunto de hasta seis métodos que el sistema invoca y podemos gestionar desde la actividad de la aplicación.

En el siguiente esquema podemos ver el ciclo vida representado de forma gráfica y el momento en el que se ejecuta cada método.



Los eventos del ciclo de vida

- **onCreate**(Bundle)Representa el momento en el que la actividad se crea. Este método normalmente lo generará el asistente al crear una nueva actividad en Android, y es donde crearemos todo lo que vaya a necesitar la actividad. Si antes hemos salvado los datos de la actividad en un objeto Bundle, podremos utilizarlo para regenerarla. Normalmente no lo usaremos.
- **onStart**()La actividad va a pasar a estar en pantalla, aunque no necesariamente visible. Si venimos de una parada, pasaremos antes por onRestart().
- **onRestart**()Anterior a onStart() cuando procedemos de una llamada a onStop().
- **onResume**()La actividad va a empezar a responder a la interacción del usuario.
- **onPause**()La actividad va a dejar de responder a la interacción del usuario.

- onStop()La actividad ha pasado completamente a segundo plano.
- onDestroy()La actividad va a ser destruida y sus recursos liberados.

Cuando necesitemos implementar uno de estos métodos, lo haremos añadiendo a nuestra actividad con estos perfiles:

```
public class MiActividad extends Activity {
     protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
...
}
     protected void onStart() {
super.onStart();
...
}
     protected void onRestart() {
super.onRestart();
...
}
     protected void onResume() {
super.onResume();
...
}
     protected void onPause() {
...
super.onPause();
}
     protected void onStop() {
...
onStop();
}
     protected void onDestroy() {
...
super.onDestroy();
}
}
```

Es importante mantener la llamada al método de la superclase para no encontrarnos con sorpresas. Las tareas de cada evento que están por encima de nuestra actividad deben mantenerse. Esta llamada irá al principio de los eventos de entrada, y al final de los de salida. De esa forma nos evitaremos sorpresas, ya que los elementos de la actividad que necesitemos pero que no estarán bajo nuestro control estarán creados antes de usarlos, y se destruirán después.

Ejercicio práctico

Para comprender mejor el ciclo de vida, vamos a ponerlo en práctica en el proyecto que hemos creado previamente. Vamos al archivo **MainActivity.kt**; si no puedes localizarlo, repasa el punto 5 de la Unidad anterior, en el que habla sobre la estructura del IDE.

Una vez localizado, sobrescribe los métodos **onCreate()**, **onRestart()**, **onStart()**, **onResume()**, **onPause()**, **onStop()** y **onDestroy()**.

En cada uno de los métodos utiliza la función **Log.d()** con un texto descriptivo del método.

La clase **android.util.Log** dispone de diferentes funciones para escribir en un log, es decir, un archivo en el que aparecen mensajes relevantes sobre la ejecución de la aplicación.

Log.d() sirve para mensajes de depuración, que tienen menos prioridad que los mensajes **Log.e()**, que son errores y se muestran en rojo en las salida de mensajes Logcat. Entonces, veamos un ejemplo cómo sobrescribimos la función **onCreate()** para mostrar un mensaje:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    Log.d(TAG, "Ciclo de vida - onCreate")
}
```

Observamos cómo utilizamos **override** para informar al compilador de que no estamos creando un método nuevo, sino sobrescribiendo el de la clase padre, que tiene igual nombre y parámetros.

Al sobrescribir funciones de las clases del sistema Android, recuerda que en la mayoría de los casos será conveniente llamar primero al método original. Lo haremos al principio del método, mediante el objeto **super**, que es la palabra clave para identificar al objeto padre.

Luego hemos añadido la llamada **Log**. La constante **TAG** sirve ver en el log desde qué clase imprimimos el mensaje. Podríamos definirla de la siguiente manera:

```
companion object {
    private const val TAG = "MainActivity"
}
```

Ejecuta la aplicación en el emulador y comprueba qué resultados obtienes en el Logcat.

Luego cambia a otra aplicación y comprueba qué métodos del ciclo de vida se han ejecutado. Cierra la aplicación. ¿Qué métodos se ejecutan?

Obra publicada con **Licencia Creative Commons Reconocimiento Compartir igual 4.0** <http://creativecommons.org/licenses/by-sa/4.0/>