**Ciclo de vida de las actividades y otros detalles.**

*Adaptado de: http://elpuig.xeill.net/Members/vcarceler/misc/android/ciclo-de-vida-de-las-actividades-y-otros-detalles-de-la-plataforma*

Android está enfocado hacia los dispositivos móviles que cuentan con unas características diferentes de los ordenadores personales. El tamaño de la pantalla, los recursos disponibles o la alimentación mediante batería son solo algunos de los detalles más evidentes que condicionan el desarrollo de las aplicaciones.

Además muchos de estos dispositivos móviles son equipos de comunicación, teléfonos, en los que es posible recibir o realizar llamadas. Por ejemplo nadie espera tener que cerrar una aplicación para poder atender a una llamada. Cualquier usuario desea que la aplicación se interrumpa de manera automática y se presente la interfaz que permite atender a la llamada entrante. Por esto el ciclo de vida de las actividades estará condicionado por la interacción del usuarios y también por otros eventos que pueden ocurrir.

A diferencia de lo que sucede en los ordenadores personales aquí el usuario sólo pide que se lancen aplicaciones. Estas no tienen opción terminar. Es la propia plataforma la que mata las aplicaciones cuando necesita recursos, pero cuando el usuario vuelve a la aplicación que esperaba tener abierta esta se lanza de manera automática recuperando su estado previo. Así las aplicaciones tienen un ciclo de vida que está controlado por el usuario y la plataforma.

**Activities stack, la interfaz de usuario de una aplicación**

Las interfaces de usuario basadas en ventanas que se solapan han tenido mucha aceptación en los ordenadores personales, pero no son adecuadas para dispositivos con la pantalla reducida y que se deben utilizar con mayor sencillez. En Android se utiliza una interfaz que en general ocupa toda la pantalla del dispositivo (aunque es posible lanzar notificaciones o diálogos que permitan ver parcialmente la pantalla que se encuentra detrás).

Así la interfaz de una aplicación estará formada por un conjunto de pantallas que permiten la interacción con el usuario. Cada una de estas pantallas será una instancia de [Activity](http://developer.android.com/reference/android/app/Activity.html).

Aunque, por motivos de seguridad, cada aplicación se ejecuta en su propio proceso (que además corresponde a un usuario diferente de Linux) es posible compartir pantallas entre diferentes aplicaciones. La plataforma permite que desde una aplicación se invoque cualquiera de las pantallas de otra aplicación, así si ya tenemos una *Activity* que muestra un mapa con determinada información no es necesario que la volvamos a implementar en cada aplicación que la necesita.

En todo momento el usuario tiene la posibilidad de pulsar el botón físico atrás que le permite volver a la pantalla anterior. Así, desde el punto de vista del usuario una aplicación está formada por la pila de pantallas abiertas.

**Ciclo de vida de las *Activities***

Las solicitudes para lanzar nuevas actividades, o recuperar las que no son visibles, se llaman [Intents](http://developer.android.com/reference/android/content/Intent.html) (intentos, solicitudes). Un *Intent* es el responsable de que le lance la primera actividad de una aplicación, y nuevos intentos serán responsables de ir lanzando las nuevas pantallas que sean necesarias.

Cada *Activity* puede encontrarse en diferentes estados:

* Activa o en ejecución cuando ocupa el primer plano. La pantalla está visible y tiene el foco de la interacción del usuario.
* Pausada, cuando ha perdido el foco pero todavía es parcialmente visible. Típicamente ocurre cuando se abre un diálogo encima de la pantalla.
* Parada o detenida, cuando nada de ella es visible.

El sistema puede terminar una Activity en cualquier momento, bien invocando su método [finish()](http://developer.android.com/reference/android/app/Activity.html" \l "finish()) o matando su proceso. Pero cuando escaseen los recursos y haya que terminar alguna Activity, acabará primero con aquellas que están paradas, luego con las pausadas y, ya en una situación crítica, con aquellas activas.

En cada caso cuando una Activity termina debe guardar su estado para que cuando sea lanzada lo pueda recuperar. Cuando se realizan cambios de estado la plataforma invoca determinados métodos de la actividad para que esta pueda realizar las operaciones oportunas.

Los métodos protegidos son:

void onCreate(Bundle *savedInstanceState*)   
void onStart()   
void onRestart()   
void onResume()   
void onPause()   
void onStop()   
void onDestroy()

Es posible monitorizar:

* El ciclo completo de vida de la actividad desde la primera llamada a [onCreate()](http://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) hasta la llamada final a [onDestroy()](http://developer.android.com/reference/android/app/Activity.html" \l "onDestroy()).
* El ciclo durante el que la actividad es visible, aunque sea parcialmente, desde [onStart()](http://developer.android.com/reference/android/app/Activity.html" \l "onStart()) hasta [onStop()](http://developer.android.com/reference/android/app/Activity.html" \l "onStop()).
* El ciclo durante el que la actividad está en primer plano y tiene el foco de atención, desde [onResume()](http://developer.android.com/reference/android/app/Activity.html" \l "onResume()) hasta [onPause()](http://developer.android.com/reference/android/app/Activity.html" \l "onPause()).

En cada uno de los métodos es necesario invocar el método de la superclase:

protected void onPause() {  
 super.onPause();  
 . . .  
}

El diagrama de las posibles transiciones de estado es:

