

<http://www.androidcurso.com/index.php/tutoriales-android/37-unidad-6-multimedia-y-ciclo-de-vida/158-ciclo-de-vida-de-una-actividad#:~:text=Una%20aplicaci%C3%B3n%20Android%20corre%20dentro,para%20asign%C3%A1rsela%20a%20otra%20aplicaci%C3%B3n.>

## Ciclo de vida de una actividad

El ciclo de vida de una aplicación Android es bastante diferente al ciclo de vida de una aplicación en otros S.O., como Windows. La mayor diferencia es que, en Android el ciclo de vida es controlado principalmente por el sistema, en lugar de ser controlado directamente por el usuario.



**video[Tutorial]** Ciclo de vida de una aplicación en Android :

(<https://www.youtube.com/watch?v=E1vAobJ8Fxo>)

Una aplicación en Android está formada por un conjunto de elementos básicos de interacción con el usuario, conocidos como actividades. Además de varias actividades, una aplicación también puede contener servicios. El ciclo de vida de los servicios se estudiará en el capítulo 8. Son las actividades las que realmente controlan el ciclo de vida de las aplicaciones, dado que el usuario no cambia de aplicación, sino de actividad. El sistema mantiene una pila con las actividades previamente visualizadas, de forma que el usuario puede regresar a la actividad anterior pulsando la tecla “retorno”.

Una aplicación Android corre dentro de su propio proceso Linux. Este proceso se crea con la aplicación y continuará vivo hasta que ya no sea requerido y el sistema reclame su memoria para asignársela a otra aplicación.

Una característica importante, y poco usual, de Android es que la destrucción de un proceso no es controlada directamente por la aplicación, sino que es el sistema el que determina cuándo destruir el proceso. Lo hace basándose en el conocimiento que tiene de las partes de la aplicación que están corriendo (actividades y servicios), en la importancia de dichas partes para el usuario y en cuánta memoria disponible hay en un determinado momento.

Si tras eliminar el proceso de una aplicación, el usuario vuelve a ella, se crea de nuevo el proceso, pero se habrá perdido el estado que tenía esa aplicación. En estos casos, será responsabilidad del programador almacenar el estado de las actividades, si queremos que cuando sean reiniciadas conserven su estado.

Como vemos, Android es sensible al ciclo de vida de una actividad; por lo tanto, necesitas comprender y manejar los eventos relacionados con el ciclo de vida si quieres crear aplicaciones estables.

Una actividad en Android puede estar en uno de estos cuatro estados:

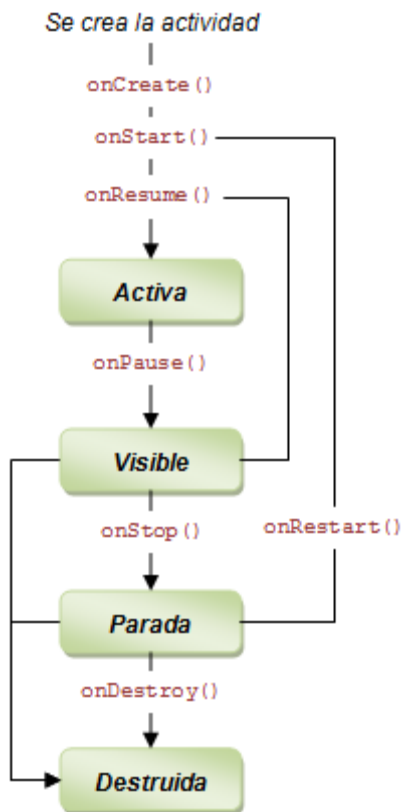
**Activa (*Running*):** La actividad está encima de la pila, lo que quiere decir que es visible y tiene el foco.

**Visible (*Paused*):** La actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra actividad con alguna parte transparente o que no ocupa toda la pantalla. Cuando una actividad está tapada por completo, pasa a estar parada.

**Parada (*Stopped*):** Cuando la actividad no es visible. El programador debe guardar el estado de la interfaz de usuario, preferencias, etc.

**Destruída (*Destroyed*):** Cuando la actividad termina al invocarse el método *finish()*, o es matada por el sistema.

Cada vez que una actividad cambia de estado se van a generar eventos que podrán ser capturados por ciertos métodos de la actividad. A continuación, se muestra un esquema que ilustra los métodos que capturan estos eventos.



#### Ciclo de vida de una actividad.

**onCreate(Bundle):** Se llama en la creación de la actividad. Se utiliza para realizar todo tipo de inicializaciones, como la creación de la interfaz de usuario o la inicialización de estructuras de datos. Puede recibir información de estado de la actividad (en una instancia de la clase Bundle), por si se reanuda desde una actividad que ha sido destruida y vuelta a crear.

**onStart():** Nos indica que la actividad está a punto de ser mostrada al usuario.

**onResume():** Se llama cuando la actividad va a comenzar a interactuar con el usuario. Es un buen lugar para lanzar las animaciones y la música.

**onPause():** Indica que la actividad está a punto de ser lanzada a segundo plano, normalmente porque otra actividad es lanzada. Es el lugar adecuado para detener animaciones, música o almacenar los datos que estaban en edición.

**onStop():** La actividad ya no va a ser visible para el usuario. ¡Ojo si hay muy poca memoria! es posible que la actividad se destruya sin llamar a este método.

**onRestart():** Indica que la actividad va a volver a ser representada después de haber pasado por *onStop()*.

**onDestroy():** Se llama antes de que la actividad sea totalmente destruida. Por ejemplo, cuando el usuario pulsa el botón de volver o cuando se llama al método *finish()*. Ojo si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.