Tema 3: Ciclo de vida de una Activity

La pila de Actividades de Android

En este tema veremos …

- Ciclo de vida de una Activity
- Estados de una Activity
- Eventos programables
- Guardando el estado de una Activity

Ciclo de vida

Activity stack
Un esquema general
Los posibles estados

Activity stack

Las interfaces de usuario basadas en ventanas que se solapan han tenido mucha aceptación en los ordenadores personales, pero no son adecuadas para dispositivos móviles.

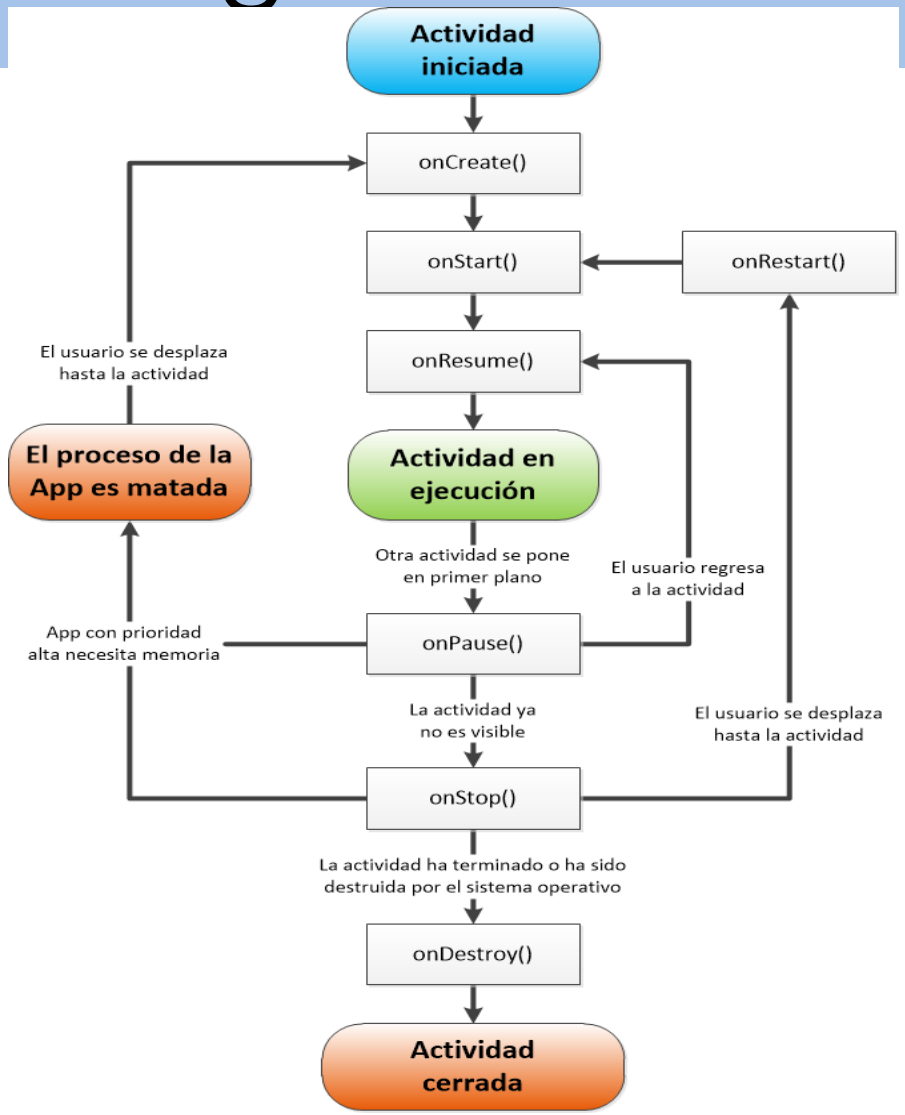
Así la interfaz de una aplicación estará formada por un conjunto de pantallas que permiten la interacción con el usuario. Cada una de estas pantallas será una instancia de Activity.

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.**

Un esquema general

En realidad una actividad de una app en Android sólo tiene 4 posibles estados:

- **Iniciada** (Started)
- **En ejecución** (Running)
- **Cerrada** (Shut down)
- **Destruida** (Killed)



Los posibles estados

En realidad una actividad de una app en Android sólo tiene 4 posibles estados:

- **Iniciada** (Started)
- **En ejecución** (Running)
- **Cerrada** (Shut down)
- **Destruida** (Killed)

Cada transición de estado dispara uno o mas eventos que podemos capturar y programar

Estados de una Acivity

Iniciada
En ejecución
Cerrada
Destruida

Iniciada

Iniciada: Se lanza la Activity y se le empiezan a asociar recursos del sistema. Además se coloca en la parte más alta de la pila de actividades.

En ejecución

En ejecución: se trata de la actividad que se está ejecutando en ese momento: es visible, tiene el foco de la aplicación y es capaz de recibir datos por parte del usuario. Android tratará por todos los medios de mantener esta actividad en ejecución, deteniendo cualquier otra actividad en la pila siempre que sea necesario.

Cerrada

Cerrada: Ha pasado a segundo plano y está completamente tapada por una nueva actividad, en ese caso el sistema también puede optar por cerrarla si necesita liberar recursos.

Destruida

Destruida: El sistema decide que necesita los recursos y decide destruir a la Activity. Esto puede ser provocado porque el usuario hace tiempo que no accede a la misma o porque está en la parte mas baja de la pila de actividades.

Eventos programables

OnCreate
OnStart
OnResume
OnPause
OnRestart
OnStop
OnDestroy

OnCreate

onCreate(): Se dispara cuando la *Activity* es llamada por primera vez. Aquí es donde debemos crear la inicialización normal de la aplicación, crear vistas, hacer los bind de los datos, etc. Este método te da acceso al estado de la aplicación cuando se cerró. Después de esta llamada siempre se llama al onStart().

OnStart

onStart(): Se ejecuta cuando la *Activity* se está mostrando apenas en la pantalla del dispositivo del usuario

OnResume

onResume(): Se ejecuta una vez que la Activity ha terminado de cargarse en el dispositivo y el usuario empieza a interactuar con la aplicación. Cuando el usuario ha terminado de utilizarla es cuando se llama al método onPause().

OnPause

onPause(): Se ejecuta cuando el sistema arranca una nueva *Activity* que necesitará los recursos del sistema centrados en ella. Hay que procurar que la llamada a este método sea rápida ya que hasta que no se termine su ejecución no se podrá arrancar la nueva *Activity*. Después de esta llamada puede venir un onResume() si la *Activity* que haya ejecutado el onPause() vuelve a aparecer en primer plano o un onStop() si se hace invisible para el usuario.

OnRestart

onRestart(): Se ejecuta cuando tu *Activity* ha sido parada, y quieres volver a utilizarla. Si ves el diagrama podrás ver que después de un onStop() se ejecuta el onRestart() e inmediatamente llama a un onStart().

OnStop

onStop(): Se ejecuta cuando la *Activity* ya no es visible para el usuario porque otra *Activity* ha pasado a primer plano. Si vemos el diagrama, después de que se ha ejecutado este método nos quedan tres opciones: ejecutar el onRestart() para que la *Activity* vuelva a aparecer en primer plano, que el sistema elimine este proceso porque otros procesos requieran memoria o ejecutar el onDestroy() para apagar la aplicación.

OnDestroy

onDestroy(): Esta es la llamada final de la *Activity*, después de ésta, es totalmente destruida. Esto pasa por los requerimientos de memoria que tenga el sistema o porque de manera explícita el usuario manda a llamar este método. Si quisiéramos volver a ejecutar la *Activity* se arrancaría un nuevo ciclo de vida.

Guardando el estado de una Activity

¿Cuándo se pierde un estado?
¿Cómo actuar?
Un ejemplo

¿Cuando se pierde un estado?

En ocasiones al regresar a la primera Activity de una app es posible que el sistema haya eliminado el proceso que ejecutaba la actividad.

En este caso, el proceso será creado de nuevo, pero se habrá perdido su estado, es decir, se habrá perdido el valor de sus variables y el puntero de programa.

En este apartado estudiaremos un mecanismo sencillo que nos proporciona Android para resolver este problema.

Cuando se ejecuta una actividad sensible a la inclinación del teléfono, es decir, puede verse en horizontal o en vertical, se presenta un problema similar al anterior.

La actividad es destruida y vuelta a construir con las nuevas dimensiones de pantalla y por lo tanto se llama de nuevo al método onCreate(). ***Antes de que la actividad sea destruida también resulta fundamental guardar su estado.***

Cómo actuar?

Dependiendo de lo sensible que sea la perdida de la información de estado se podrá actuar de diferentes maneras:

1. No hacer nada (perdiendo toda la información)
2. Si es muy sensible, se recomienda usar el método onPause() para guardar el estado y onResume() para recuperarlo.

Existe una tercera posibilidad:

- **onSaveInstanceState(Bundle)**: Se invoca para permitir a la actividad guardar su estado. Ojo, si hay muy poca memoria, es posible que la actividad se destruya sin llamar a este método.

- **onRestoreInstanceState(Bundle)**: Se invoca para recuperar el estado guardado por onSaveInstanceState().

No es tan fiable como onPause() pero si es mas sencilla

¿Un ejemplo?

Guardando un par de variables durante la ejecución de una Activity.

```
String var;
int pos;

	@Override
	protected void onSaveInstanceState(Bundle guardarEstado) {
		super.onSaveInstanceState(guardarEstado);
		guardarEstado.putString("variable", var);
		guardarEstado.putInt("posicion", pos);
	}

	@Override
	protected void onRestoreInstanceState(Bundle recEstado) {
		super.onRestoreInstanceState(recEstado);
		var = recEstado.getString("variable");
		pos = recEstado.getInt("posicion");
	}
```