Activity

Si creamos una pantalla para una aplicación, podemos decir que estamos creando una actividad, aunque hay otras formas de crear pantallas, pero lo que siempre va a ser igual es que una actividad es la forma básica.

En Java casi todo viene dentro de una clase o es una clase, por lo que cuando vamos a crear una activity hacemos que esta herede de "AppCompatActivity". Hoy en día la clase que se utiliza para crear una activity es esta, debido a que las actividades han evolucionado e implementan nuevas funcionalidades, animación o efectos. También se puede utilizar la clase "Activity" sacrificando los beneficios de la clase anterior.

Ciclo de Vida de una Activity

Este ciclo tiene estados y al pasar de un estado a otro tenemos métodos predefinidos a los que nosotros les podemos colocar instrucciones para ejecutarse en ese momento.

- Created <- onCreate()
- Started <- onStart()
- Paused <- onPause()
- Resumed <- onResume()
- Stopped <- onStop()
- Destroyed <- onDestroy()

OnCrate(): Se ejecuta una sola vez al inicio de una actividad. En este se definen las variables, un archivo XML como la parte gráfica de la actividad o también la configuración de la interfaz, etc. Cuando el método onCreate() termina de ejecutarse llama al método onStart() seguido de onResume().

onStart(): Aquí es donde la actividad se comienza a mostrar al usuario.

onResume(): Aquí la actividad entra en primer plano y el usuario interactúa con la actividad.

onPause(): Aquí se encuentra parcialmente oscurecida por una actividad que se halla en primer plano. En este estado no se reciben datos de entrada del usuario y no puede ejecutarse código.

onStop(): En este método se encuentra completamente invisible u oculto para el usuario. Las variables e información se mantienen pero no podemos ejecutar el código.

onRestart(): Este método se llama después del onStop(). Después, de este continúa el onStart(), luego el onResume() y finalmente ya está de nuevo mostrándose la actividad al usuario.

onDestroy(): Cuando el sistema destruye la actividad, se manda a llamar al método onDestroy() para la actividad. Este método es la última oportunidad que tenemos para limpiar los recursos

Layouts

Todas las interfaces gráficas de una aplicación android son desarrolladas haciendo uso de XML. Todos los archivos que tienen una interfaz definida están con la extensión .XML y los vamos a encontrar almacenados en el directorio: res/layout.

Este directorio es el asignado por android para gestionar los archivos que van a ser interfaces de nuestra aplicación. Además de definirnos la interfaz gráfica de una aplicación también colocamos una estructura y orden a nuestros elementos.

Un **layout** es un contenedor que nos permite asignar ciertas propiedades o características a los elementos que se colocaran dentro de los layouts. Tenemos diferentes tipos de layouts que nos van a permitir distintos acomodos a los elementos interiores.

Tipos de Layouts

Debemos de visualizar cualquier layout como un contenedor que contiene reglas para colocar elementos de una interfaz gráfica de android.

Los layouts elementales que tenemos en android son:

- LinearLayout
- RelativeLayout
- FrameLayout

→ Son los más utilizados

- AbsolutLayout → DEPRECATED, ya no se utiliza.
- FrameLayout

LinearLayout: Su propiedad principal es muy sencilla. El acomodo dentro del LinearLayout puede ser de dos formas: vertical u horizontal.

RelativeLayout: Coloca a los elementos en una posición relativa a otro elemento.

Absolute Layout: Coloca a los elementos que contiene en una posición absoluta, manteniendolos en dicha posición sin importar lo que suceda.

TableLayout: Se usa bastante en el momento de tener un acomodo similar al de las tablas de una hoja de cálculo.

FrameLayout: Este elemento siempre acomodará los elementos internos en la parte superior izquierda. Un error común es colocar varios elementos en este ya que uno tapará a otro. Este elemento se ocupa comunmente para colocar Fragments.