Android Básico

*El objetivo de las siguientes entregas es ofrecer una **introducción a Android desde cero**. El manual va a estar compuesto por capítulos muy específicos con secciones que realizan tareas concretas. Queda de tu parte el acoplamiento del código de las secciones que necesites para tus aplicaciones.*

Todo el código que sea utilizado estará colocado en mi Github.

https://github.com/josedlujan

Autores del manual

Este manual ha sido realizado por los siguientes colaboradores de DesarrolloWeb.com:

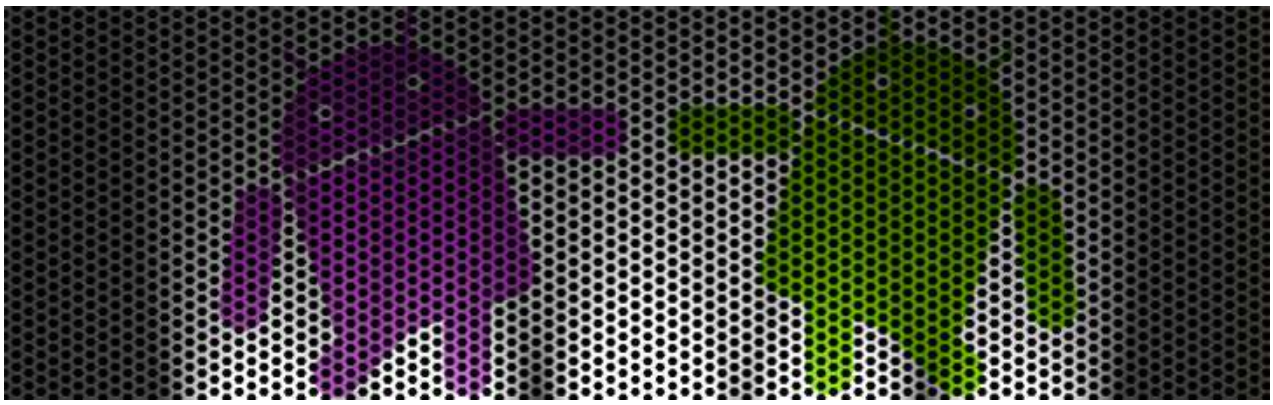
José Dimas Luján
José Dimas es fundador de Ockham Ti, empresa de desarrollo de software, app móviles, videojuegos y cursos.
(10 capítulos)

Introducción a Android

En este primer artículo nos adentraremos en algunos aspectos básicos sobre Android: presentación, historia y estructura principal.

¿Qué es Android?

La evolución de la tecnología va a paso veloz, Android es de las tecnologías que esta alcanzado a todos por el simple motivo de que se encuentra en los móviles. Android es un sistema operativo basado en Linux. La diferencia principal es que tiene módulos que responden a la pantalla táctil, eventos nativos del móvil. Se desarrolló por una compañía llamada Android, Inc. En 2005 Google adquiere la empresa para seguir trabajando en el mismo proyecto que después conociera la luz como un S.O. para móviles denominado finalmente como Android.



A finales de 2008 Septiembre-Octubre, sale a la venta el primer dispositivo móvil con Android.

Historia de Android

Android tiene una característica peculiar: las versiones tienen nombre de postres en inglés y cada versión que cambia, continúa de forma incremental en el alfabeto, es decir que si el primer nombre inicio con A, el siguiente con B, el siguiente C y así sucesivamente; ya veremos que sucede cuando lleguen a la Z.

Hasta el día de hoy, que comienzo a escribir el manual Android para Desarrolloweb.com, tenemos la versión 4.4 KitKat.

Demos un repaso a las Versiones.

Versión 1.0 Apple Pie - Salió en septiembre del 2008.
Versión 1.1 Banana Bread - Salió en febrero 2009.
Versión 1.5 Cup Cake - Salió en abril 2009
Versión 1.6 Donut - Salió en septiembre 2009
Versión 2.0 Eclair - Salio en octubre 2009
Versión 2.2 Froyo - Salió en mayo 2010
Versión 2.3 Gingerbread - Salió en diciembre 2010
Versión 3.0 Honeycomb - Salió en febrero 2011
Versión 4 .0 Ice Cream Sandwich - Salió en octubre 2011
Versión 4.1 Jelly Bean - Salió en julio 2012
Versión 4.4 KitKat - Salió en octubre 2013

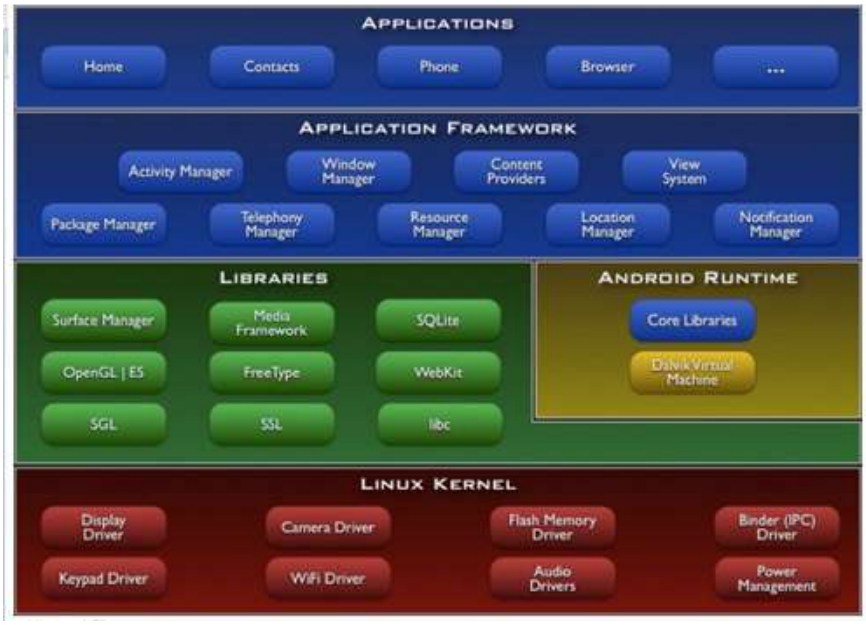
Para más información, puedes consultar este enlace: http://www.android.com/versions/kit-kat-4-4/

Estructura

Ya mencionamos que Android está basado en Linux. Para ser más específicos, hablamos del kernel. Android utiliza

como base el kernel de Linux. Esto no significa que por estar basado en el algo que se desarrolló en Linux funcione para Android, por ejemplo Android no tiene soporte glibc.

Ahora vamos a darle un vistazo a la estructura:



Tenemos esta estructura:

- Capa Roja, Kernel.
- Capa Verde, Librerias.
- Capa Amarilla, Android runtime.
- Capa Azul, application Framework
- Capa Azul Ultima, Application.

Vamos a conocer cada una de ellas:

Capa del Kernel (Roja)

Aquí tenemos el corazón de Android: el manejo de memoria, procesos, *drivers*, etc. Aquí es donde se da la comunicación con el *hardware*. Esto nos sirve para no estar peleando con los fabricantes de cada móvil, nos ayuda a solo usar la "cámara" y no tener que saber cómo funciona la cámara del fabricante X, fabricante Y; solamente hacemos lo que nos interesa, que sería usar la cámara y listo. Además de eso, aquí se administran los recursos del celular, memoria, energía...

Capa Librerías (Verde)

Esta capa tiene las librerías nativas de Android, están escritas en C o C++ y tienen tareas específicas.

- **Surface manager**: Gestión del acceso a la pantalla.
- **Media Framework**: Reproducción de imágenes, audio y vídeo.
- **SQLite**: BD
- **Webkit**, Navegador.
- **SGL**: Gráficos 2D.
- **OpenGL**: Gráficos 3D.
- **Freetype**: Renderizar vectores o imágenes.

Android Runtime (Capa Amarilla)

Esta capa amarilla no se considera al 100% una capa. Lo que es muy importante comentar es que aquí se encuentra Dalvik, la máquina virtual de Android, que no es lo misma que la Java Virtual Machine. Esto quiere decir que cuando compilamos en Java lo que se genera solamente va a funcionar en la JVM, porque Dalvik es una máquina virtual, pero de Android, así que el ByteCode que genera Java es inservible para Dalvik.

Algunas de las características de Dalvik son:

- Trabaja en entorno con restricción de memoria y procesador.
- Ejecuta el formato .dex.
- Convierte .class en .dx.

Application Framework (Capa azul)

Esta capa es la es más visible para el desarrollador, ya que la mayoría de los componentes que forman parte del desarrollo los vamos a encontrar aquí.

- **Activity Manager**- Administra las actividades de nuestra aplicación y el ciclo de vida.
- **Windows Manager**- Administra lo que se muestra en la pantalla.
- **Content Provider**-. Administra dependiendo de cómo le indiquemos algunos contenidos, puede ser información que necesitamos la encapsule para enviar o compartir.
- **View**- Las vistas de elementos que son parte de la interfaz gráfica, como los mapas, cuadros de texto, etc.
- **Notification Manager**- Administra las notificaciones.
- **Package Manger**- Administra los paquetes y nos permite el uso de archivos en otros paquetes.
- **Telephony Manager**- Administra lo que tiene que ver con la telefonía, llamadas, mensajes.
- **Resource Manager**- Administra recursos de la aplicación, como los xml, imágenes, sonido.
- **Location Manager**- Gestiona la posición geográfica.
- **Sensor Manager**- Gestiona los sensores que tenga el dispositivo.
- **Cámara**- Administra la cámara.
- **Multimedia**- Administra lo referente a audio, video y fotos.

Aplications (Capa Azul última)

Aquí tenemos las aplicaciones que vienen en el dispositivo, por ejemplo: el gestor de correos, los mensajes, el *market*, etc.

Artículo por José Dimas Luján

Entorno de desarrollo en Android

Analizamos el entorno más conveniente para desarrollar en Android.

Siempre que se comienza una lectura como tutorial, manual, etc. hay que considerar la fecha de su creación: el día que estoy escribiendo este manual para Desarrolloweb.com y en esta fecha, el entorno más conveniente para el desarrollo en Android es Eclipse, probablemente más adelante haga un capítulo con referencia al otro entorno Android studio.

Los elementos que necesitamos para el desarrollo de aplicaciones en Android son los siguientes:

1.Java
2.Eclipse
3.Android SDK
4.ADT

La recomendación es ir en ese orden para no perdernos si nos estamos iniciando.
Comenzaremos con la explicación :

Paso 1- Java

Para obtener Java tendremos que irnos a la página de Oracle en la sección de descargas, hay que buscar el Java JDK (Java Development Kit).

Ya en la sección de descargas del Java JDK debemos poner atención en la versión que descargamos y seleccionar la de tu sistema operativo y procesador. El JDK Esta para Linux, Mac, Solaris y Windows, además para procesadores ARM, x86 y x64, SPARC. Selecciona la que corresponda a tu máquina.

Ya que tengas el instalador, lo único que tienes que hacer es ejecutarlo y el clásico "Siguiente, siguiente" y listo. No se debe tener ningún problema.

Java SE Development Kit 7u51

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ◉ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	67.7 MB	jdk-7u51-linux-arm-vfp-hflt.tar.gz
Linux ARM v6/v7 Soft Float ABI	67.68 MB	jdk-7u51-linux-arm-vfp-sflt.tar.gz
Linux x86	115.65 MB	jdk-7u51-linux-i586.rpm
Linux x86	132.98 MB	jdk-7u51-linux-i586.tar.gz
Linux x64	116.96 MB	jdk-7u51-linux-x64.rpm
Linux x64	131.8 MB	jdk-7u51-linux-x64.tar.gz
Mac OS X x64	179.49 MB	jdk-7u51-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.02 MB	jdk-7u51-solaris-i586.tar.Z
Solaris x86	95.13 MB	jdk-7u51-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.53 MB	jdk-7u51-solaris-x64.tar.Z
Solaris x64	16.28 MB	jdk-7u51-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	139.39 MB	jdk-7u51-solaris-sparc.tar.Z
Solaris SPARC	98.19 MB	jdk-7u51-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.94 MB	jdk-7u51-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.33 MB	jdk-7u51-solaris-sparcv9.tar.gz
Windows x86	123.64 MB	jdk-7u51-windows-i586.exe
Windows x64	125.46 MB	jdk-7u51-windows-x64.exe

Paso 2- Eclipse

Ahora vamos por Eclipse, Eclipse es un IDE, no explicaremos a detalle sus características, pero sí diremos que es

el entorno por excelencia en Android y Java. Eclipse es un proyecto de código abierto y tiene su web: www.eclipse.org. Ahí vais a encontrar toda la información, diferentes versiones, documentación, ayuda, artículos, etc.

En la sección de descargas vamos a encontrar varias versiones del mismo, esas versiones van cambiando con el paso del tiempo y Eclipse siempre te va a poner la más recomendable por lenguaje o tecnología. En todo caso la que usamos la mayoría para Android es la version de Eclipse que dice: "Eclipse IDE for Java Developers", al igual que en el paso 1, tienes que ver qué versión de sistema operativo tienes y el procesador.

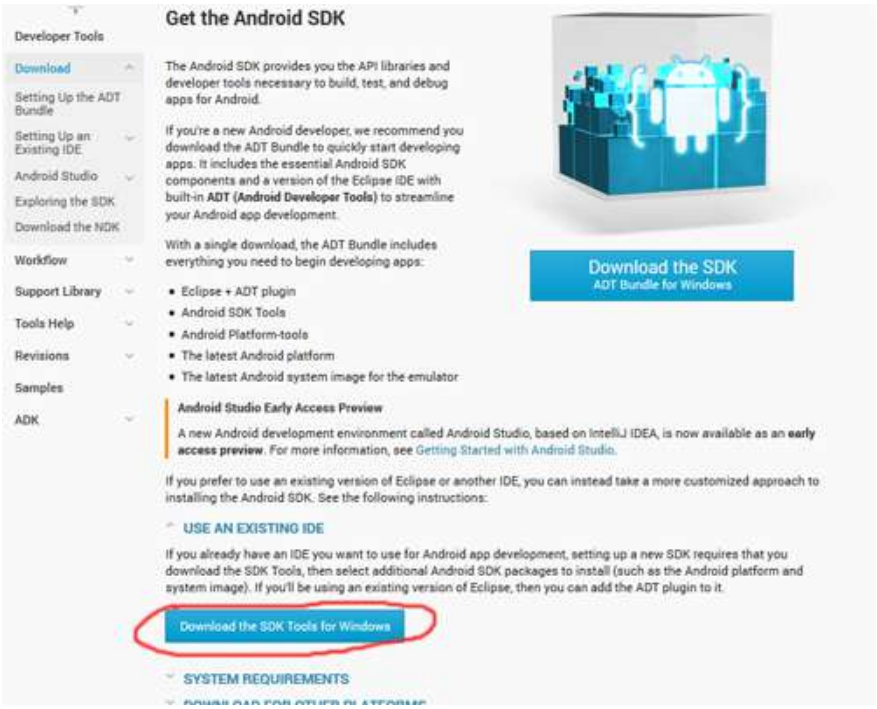
Después de descargado, descomprimimos la carpeta y tendremos un directorio Eclipse. Algo que tenemos que resaltar es que Eclipse NO se instala; para explicarlo de forma sencilla, digamos que funciona como un portable, es decir, que cada vez que se abre nos pregunta en que directorio puede trabajar o dónde colocar los proyectos, esto también nos da una ventaja ya que podremos migrar la carpeta con Eclipse a otro PC mientras coloquemos las rutas y tengamos Java en el otro.

Ya que tenemos Eclipse, vamos a abrirlo. Dentro del directorio que se descomprimió debe existir un archivo llamado Eclipse que es el ejecutable. Le damos doble clic y en lo que se carga, nos va preguntar el Workspace. Esta pregunta es para que le indiques el área de trabajo, en dónde va a colocar los proyectos que se van creando en Eclipse. Normalmente lo mejor es colocarlo en la misma carpeta de Eclipse, creando una carpeta llamada "proyectos", "workspace", "programas", etc. Con esto termina de cargar Eclipse y ya lo tenemos.

Paso 3 - Descargar el SDK de Android.

El SDK nos lo proporciona Google, en realidad TODO lo que necesitamos para desarrollar: documentación, ejemplos, iconos, lo proporciona Google, más adelante en el manual haremos referencias a páginas específicas. Ya tiene muchos años que el enlace del SDK es la mismo: http://developer.Android.com/sdk/index.html

Ahora con cuidado aquí, que tenemos varios enlaces de descargas y muchos se pierden en esta parte por no leer con cuidado, estamos buscando el SDK y se encuentra en la parte inferior que dice "USE AN EXISTING IDE" como está en la imagen marcado.



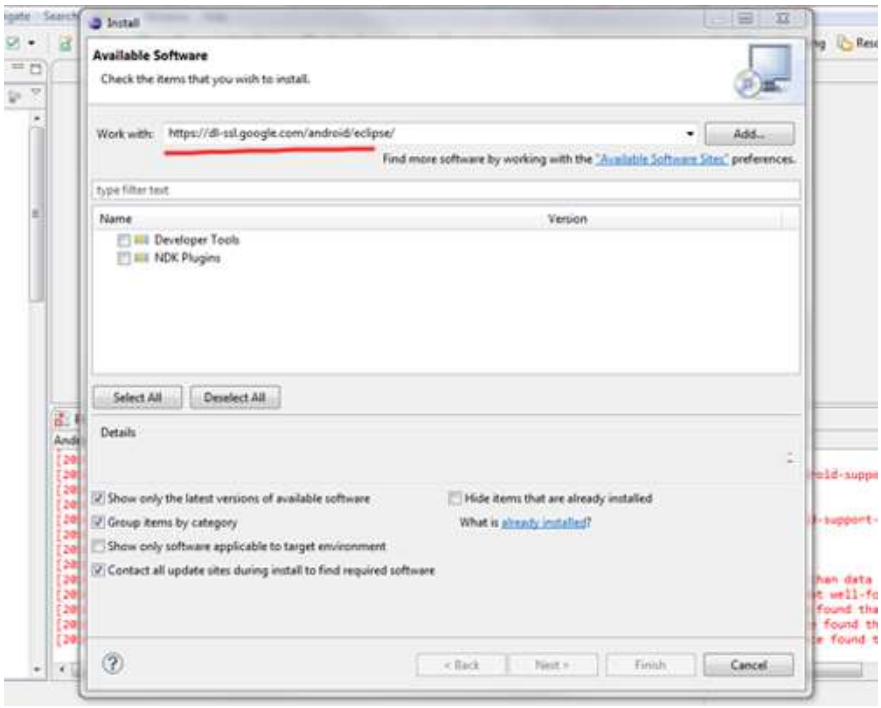
Ya que lo tenemos en nuestra máquina, lo ejecutamos y listo ya tenemos el SDK.

Paso 4. - ADT (Android Development Tools)

A diferencia de todo lo que hemos descargado e instalado, este paso a va a ser el más largo, porque tenemos que instalar el ADT dentro de Eclipse y configurar. Así que como primer paso, abrimos Eclipse, vamos a la sección "Help -> Install New Software".

Después copiamos la liga: https://dl-ssl.google.com/Android/eclipse/

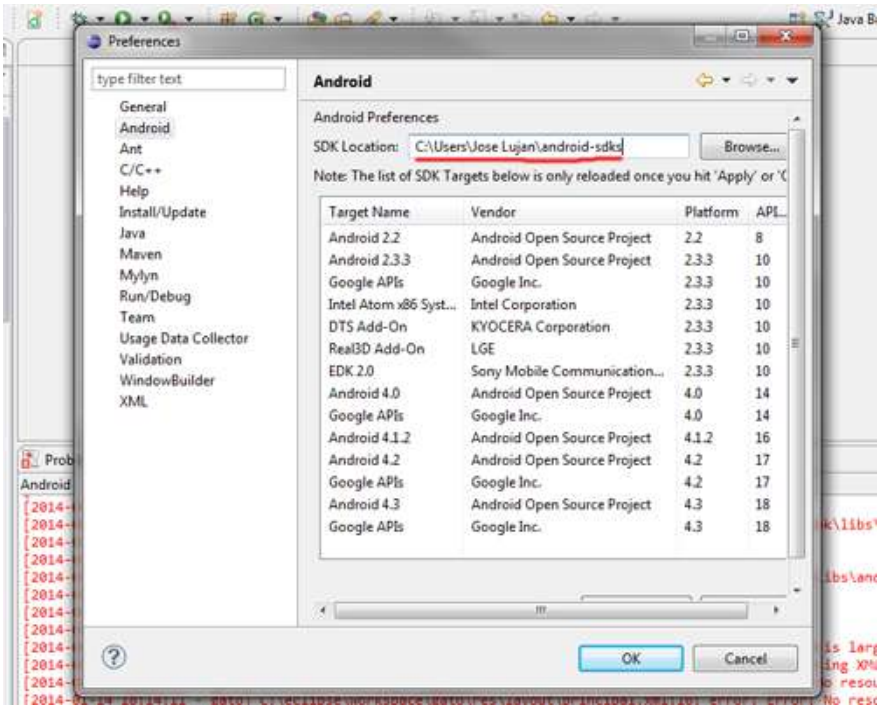
y la colocamos en donde dice “Work with” como en la imagen. Y seleccionamos “Developer Tools” y “NDK Plugins” después damos clic en "Next" para comenzar la descarga. Nos va a decir que aceptemos la licencia, le damos aceptar y listo.



Con esto ya tenemos instalado el ADT, pero nos falta configurarlo.

Configuración ADT

Para configurar el ADT tenemos que irnos a la opción del menú Windows -> Preference y le damos clic a la sección Android como en la imagen, en la opción que dice “SDK Location” tenemos que colocar la ruta en donde se instaló el SDK que instalamos en el paso 3. Por defecto se instala en una ruta similar a la que estás viendo en la imagen.



C:UsersNOMBRE USUARIOAndroid-sdks

Colocamos la ruta y le damos aceptar.

¡Listo! Con esto tenemos nuestro entorno preparado y listo para desarrollar en Android, aunque nos faltan otras cosas que no son el entorno en específico, pero las veremos más adelante.

Artículo por José Dimas Luján

Instalación de versiones de Android y Herramientas para desarrollar

Pasos a seguir para la instalación de las distintas versiones de Android y las herramientas necesarias para el desarrollo.

Para arrancar nuestras aplicaciones tenemos que tener bien claro que las versiones son algo que debemos considerar con cuidado en Android.

Vamos a explicar esto con un ejemplo: si desarrollas una aplicación para la versión 4.0 de Android, existe la posibilidad de que a los dispositivos con una versión anterior no les funcione, es decir, que estas dejando fuera a los móviles de versiones anteriores. Tenemos que resaltar que esto significa que si el 80% de los móviles en el mercado son versiones anteriores a la 4.0 y tu aplicación solamente tendrá el acceso al 20% del mercado.

Como te darás cuenta, desarrollar siempre para la última versión no va a ser conveniente y en muchos casos tenemos que buscar un equilibrio, a diferencia de lo que sucede en IOs, que tras unas semanas de liberar una actualización, el 70%-80% ya tiene la última versión. Esto es porque en Android los fabricantes son los que se encargan de la actualización.

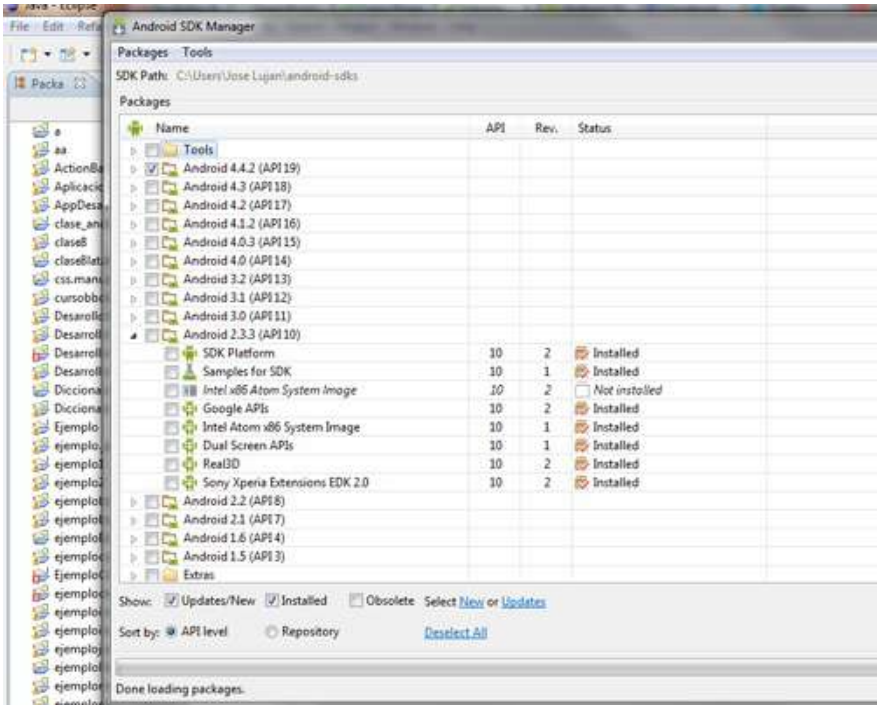
Pero también sucede al revés: si uno desarrolla una aplicación para la versión 2.3, puede que no podamos usar algunas mejoras o avances en las versiones superiores, como por ejemplo una barra de menú, algún sensor nuevo, etc. Como mencionamos anteriormente, esto es un reto para el desarrollador, que debe buscar el equilibrio.

Por ahora, las versiones que cuentan con más presencia en el mercado son las versiones 2.3, 3 y 4. Con esto quiero decir que debemos considerar desde la versión 2.3 para el desarrollo, versiones anteriores de Android ya tienen un número muy bajo en el mercado y ya casi nadie las considera para el desarrollo. Como otro dato a resaltar, parece que este 2014 es el año en el que 2.3 deja de ser la versión de Android con más móviles en el mercado para pasar la estafeta la versión 4, ya que la mayoría de los fabricantes están apresurando el paso por la gran inconformidad que se mostró durante un par de años atrás.

Entonces por ahora vamos a descargar para nuestros ejercicios las versiones 2.3, 3 y 4, además de agregar la sección extra el paquete “Android Support Library” que siempre es bueno tenerla como apoyo para soporte de versiones antiguas.

Para descargar estas versiones en Eclipse y poder usar lo que viene con ellas, tenemos que hacer lo siguiente:

Abrimos Eclipse y nos vamos a Window -> Android SDK Manager, que nos mostrará una ventana con herramientas, extras y las versiones de Android listas para descargar y poderlas incluir en nuestros proyectos.



Como vemos, son todas las versiones de Android, lo que sucede es que están en un servidor, así que tendremos que seleccionar las que necesitamos y descargarlas.

Esto que estamos viendo lo tendremos que hacer a veces para actualizar algunas versiones, conseguir acceso a algunas API, además de instalar algunas herramientas u opciones nuevas que aparezcan para el desarrollo en Android. Así que es bueno que le des una repasada a esta ventana y mirar las diferentes descargas que nos proporciona. Después de seleccionar todos los paquetes de descarga, se le puede dar a instalar. Recomiendo dejar la computadora un rato descargando esto, ya que no se baja a una velocidad habitual. Al ser servidores con acceso para tantas personas no son de lo más rápido independientemente de tu velocidad, así que con calma, lo dejas descargando, le das unas vueltas para ver que todo esté funcionando y al final ya tenemos las versiones que necesitemos para trabajar.

En realidad podríamos tener todas las versiones de Android sin ningún problema conviviendo en nuestra máquina, solamente que será un poco retardado, así que si las quieres tener, lo puedes hacer.

Con esto finalizamos las instalaciones de versiones y herramientas de Android.

Artículo por José Dimas Luján

Emulador de Android

En este capítulo vamos a conocer al emulador de Android que nos va acompañar durante el desarrollo de nuestras aplicaciones.

Comencemos explicando el concepto de emulador: un emulador es un *software* que imita al *hardware* o a un sistema operativo con el objetivo final de ejecutar un programa, aplicación, software, etc. sobre éste. En el caso de Android, el emulador lo que hace es recrear el hardware de un móvil para hacer funcionar el SO Android sobre él.

Esto nos ayuda a no tener que comprar todos los móviles con Android, con diferentes tamaños de pantalla y con diferentes versiones. Es una gran herramienta para la fase pruebas y el desarrollo de una aplicación.

También podemos ejecutar nuestra aplicación desde Eclipse en el celular, pero esto lo vamos a ver en otro apartado.

Para usar el emulador, lo que tenemos que hacer es crear un dispositivo, después configurarlo y finalmente ejecutarlo.

Vamos a **Window -> AVD Manager o Android Virtual Device Manager**, en la pestaña de **Android Virtual Devices** seleccionamos el botón **New**.

Vamos a configurar en orden sus campos:

AVD Name: Colocamos el nombre del dispositivo. Si quieres colocar “Juanito” o “Pepito” no pasa nada, esto es irrelevante; lo recomendable es colocarle qué versión de Android soporta para saber por lo menos qué configuración tiene de manera rápida, por ejemplo: “Juanitov4”

Device: Seleccionamos el dispositivo que deseamos emular. Tendremos desde un Nexus hasta tamaños de pantallas pequeños como los primeros móviles; esto depende de las dimensiones y características que necesites. Cuando seleccionamos alguno de los dispositivos, muchos de los campos se llenan con características por defecto. La recomendación siempre es dejarlo así, porque es lo ideal para su funcionamiento, aunque también puede ser que para probar alguna característica concreta de una aplicación, necesitemos más adelante especificar alguno de diferente forma.

Target: Ponemos la versión de Android que le vamos a colocar a ese dispositivo (muchas veces conviene dejar la que está por defecto.)

CPU/ABI: Éste es el procesador, igual conviene dejar el que viene de serie.

Keyboard: Lo seleccionamos si queremos usar el teclado del PC.

Skin: Si lo dejamos seleccionado, nos permite ver los botones físicos de un móvil, como por ejemplo el "Home".

Front Camera: la puedes emular, pero la recomendación es que si necesitamos la cámara, probemos la aplicación en algún móvil, ya que siempre es mejor probar esto en una cámara real.

Back Camera: Lo mismo que en el campo anterior.

Memory Option: En esta opción tenemos dos apartados la RAM del móvil. Recuerda que esto puede alentar a tu PC dependiendo de la memoria que tenga, así que no te emociones queriendo hacer que vuele el emulador, la mayoría puede funcionar con 560mb a 1Gb y esto puede volver muy lenta a tu máquina. La otra opción VM Heap es la memoria dinámica asignada a la máquina virtual. En las dos opciones conviene dejar la opción por defecto.

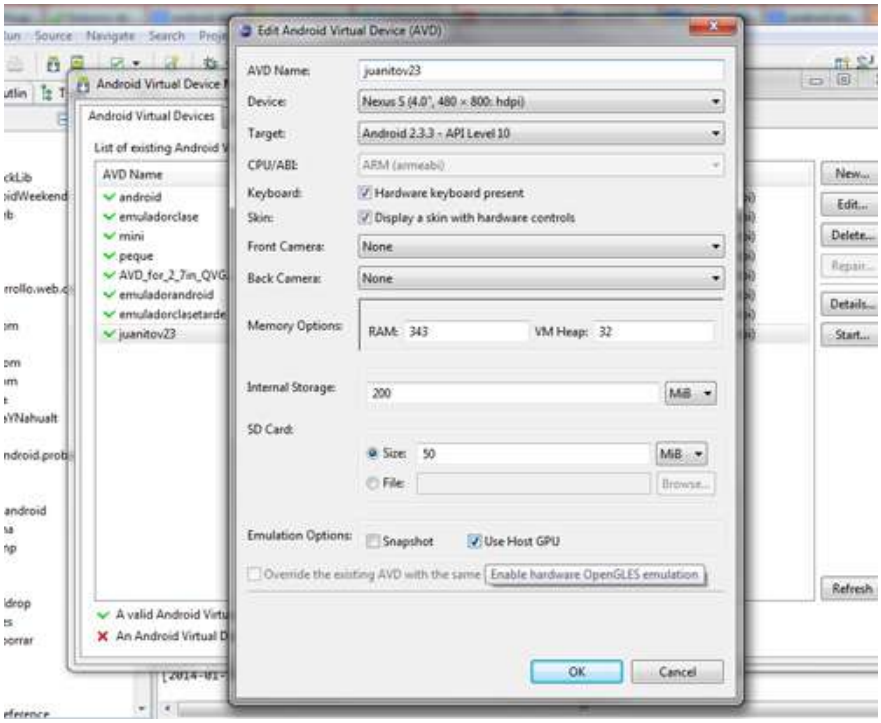
Internal Storage: Es la memoria interna del móvil y al igual que en el punto anterior, no te emociones en esta parte y coloques mucho, ya que el espacio lo toma de tu disco duro y la verdad que los móviles, tienen muy poco. A veces con 200 Mb puede ser más que suficiente.

SD Card: Esta es la tarjeta de memoria. Aquí lo que sucede es que el emulador crea un archivo y éste se comporta como la tarjeta de memoria. Si no necesitas mucho, puedes colocar 50 Mb. Tenemos por último dos opciones que no se pueden usar juntas, es decir, o usas la primera o la segunda, pero no puedes usar las dos al mismo tiempo.

Snapshot: Sirve para que vaya más rápido el emulador, pero lo que sucede es que, aunque cerraste el emulador, se inicia más rápido porque se tiene el último estado antes de cerrar, por eso al hacerlo en el emulador, si seleccionaste esta opción, tardará un poco más porque se pone a crearlo en ese momento.

Host GPU: Si seleccionas ésta, el emulador utilizara el GPU del PC. Esto sirve para que cuando este moviendo dentro del emulador todo sea mucho más fluído, porque toda la parte de procesamiento de gráficos se está haciendo con el del PC.

Con esto terminamos de configurar nuestro emulador.



El emulador tarda un poco en iniciar y esto depende de la máquina; me tocó verlo en un *notebook* y tardó 21 minutos en iniciar, pero también lo uso en un Macpro y no tarda ni un minuto, así que no desesperéis. Si tiene la palomita verde, debe de iniciar algún día. Otra recomendación es no cerrarlo, aunque vayas a trabajar con otra aplicación, no es necesario abrir otro, el mismo emulador sabe que aplicación le ordenas ejecutar al darle ejecutar, todo depende del archivo que este en primer plano en Eclipse.

Artículo por José Dimas Luján

Hola Mundo en Android

Realizamos todos los pasos necesarios para crear el mítico Hola Mundo en Android.

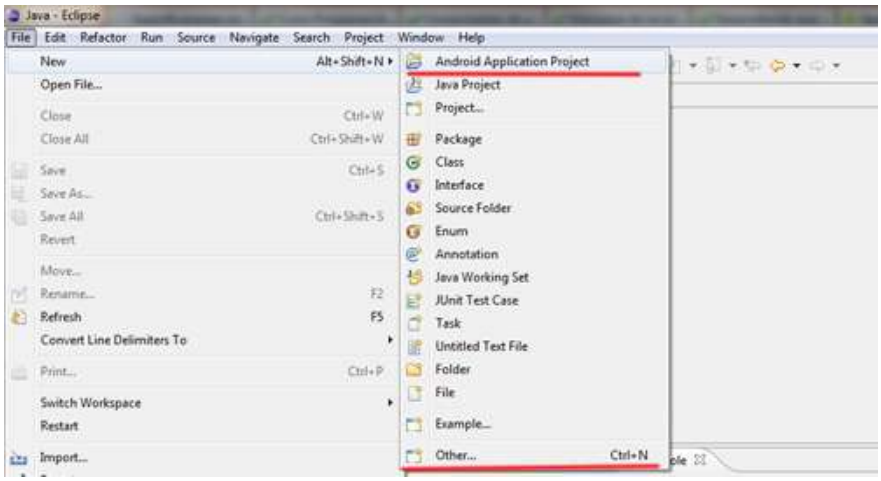
En este capítulo vamos a crear el legendario “Hola mundo”. Comenzamos abriendo Eclipse y en la parte del menú vamos por la opción:

File -> New ->> Android Aplication Project.

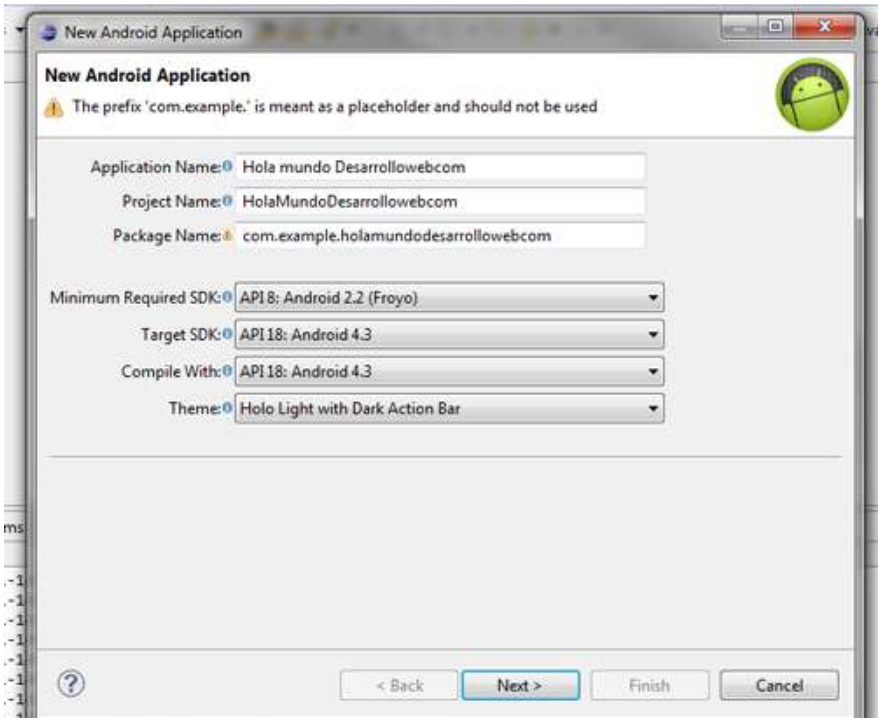
En el caso de que no te salga, puedes seguir esta otra ruta:

File ->> New -> Other -> Android -> Android Application Project

La segunda debería funcionar. En la imagen están marcados con rojo los dos caminos.



Ahora tenemos una ventana enfrente de nosotros con algunas opciones. Expliquemos qué son:



Application Name: Éste es el nombre de la aplicación con el que va aparecer en la Playstore en caso de publicarla; por ahora se llama "Hola mundo Desarrollowebcom". Al momento de colocar el nombre, los siguientes campos se llenan por defecto, pero vamos a explicarlos:

Project Name: Éste es el nombre del proyecto para Eclipse. Habitualmente se llama igual que en el campo anterior. Por recomendación no coloquéis espacios dentro del nombre ni tampoco caracteres raros, ya que se va a crear un directorio con ese nombre.

Package Name: Éste es el nombre del paquete. Debemos tratar que sea único, ya que es el nombre del paquete se sube a la Playstore y además el que se instala en los móviles. Por convención, casi siempre inicia con "com.loquetuquieres" haciendo referencia al dominio de la empresa, compañía o persona que lo esta creando. Puedes colocar los puntos "." Entre las palabras en lugar de los espacios, además de ser un nombre largo para que sea más difícil de repetir, si no quitas el prefijo que viene por defecto "com.example" la Playstore no lo aceptará, ya que piensa que es un ejemplo o un clásico "Hola mundo" pero puedes colocarle algo así: "**com.tuempresa.tunombre.nombreapp**"

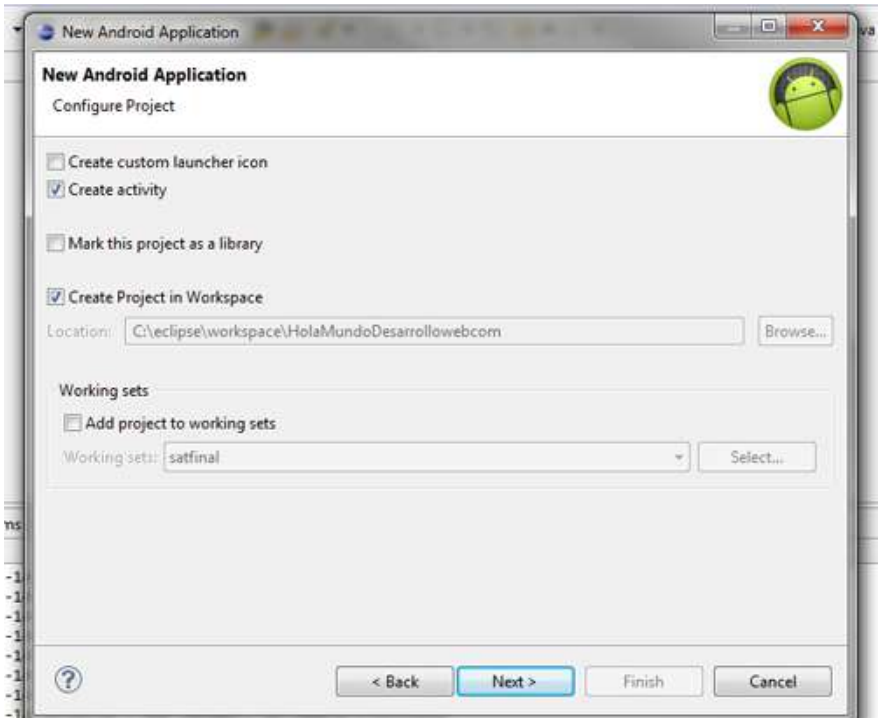
Esto es un ejemplo sencillo de un nombre correcto, en nuestro caso lo haremos así: **com.desarrolloweb.joselujan.holamundo Minium Requires SDK:** Debemos seleccionar la versión mínima del SDK que aceptará nuestro SDK, es decir, que si colocamos 2.3, el móvil que quiera instalar nuestra aplicación mínimo debe ser la versión 2.3.

Target SDK: Ahora seleccionamos el máximo nivel de API que vamos a soportar en nuestra aplicación.

Compile with: Seleccionamos la versión del SDK con la que compilaremos el proyecto. Muchas veces es costumbre colocar aquí y en el campo anterior la versión más nueva de Android.

Terminamos con esta ventana y le damos a "Siguiente".

La siguiente ventana es como la imagen. Expliquemos pues las opciones:



Create custom launcher icon: Nos pregunta si queremos crear desde un inicio el icono de la aplicación, por ahora no lo seleccionamos, ya que después lo vamos a ver por separado.

Create activity: Éste sí lo dejamos, importante, ya que crea la primera ventana de nuestra aplicación Android.

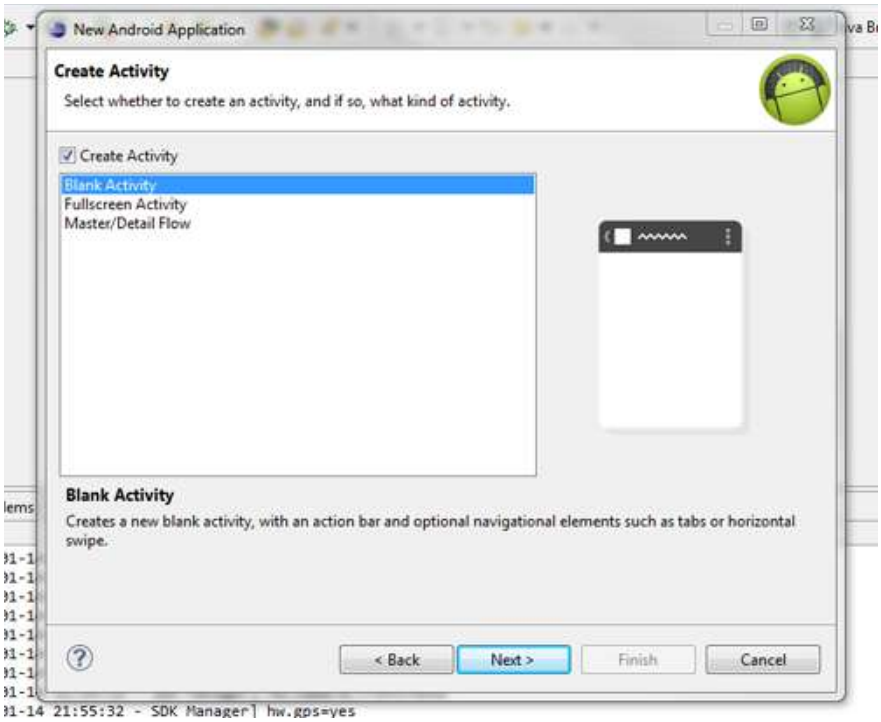
Mark this Project as a library: Éste no es necesario seleccionar en un principio casi nunca, es para que el proyecto que estamos creando lo marque como librería.

Create Prokect in Workspace: Aquí nos está indicando la ruta del proyecto que estamos creando, si la queremos cambiar la podemos colocar manualmente.

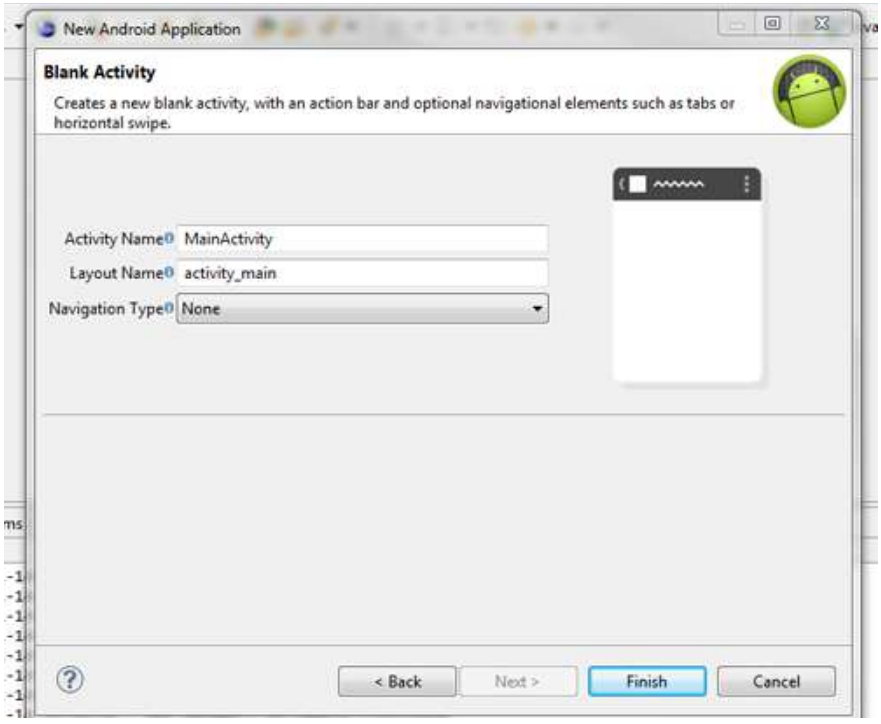
Ahora le damos "Next".

La siguiente ventana hace referencia a la "activity" que decidimos crear en la ventana pasada. Por ahora no explicaremos el termino Activity, ya que lo veremos en breve.

Seleccionamos "Create Activity" y después la opción "Black Activity".



En la siguiente ventana nos solicita tres campos:



Activity Name: El nombre de la actividad, es decir, el nombre de la clase que controla la primera pantalla. Podemos dejar el que tiene.

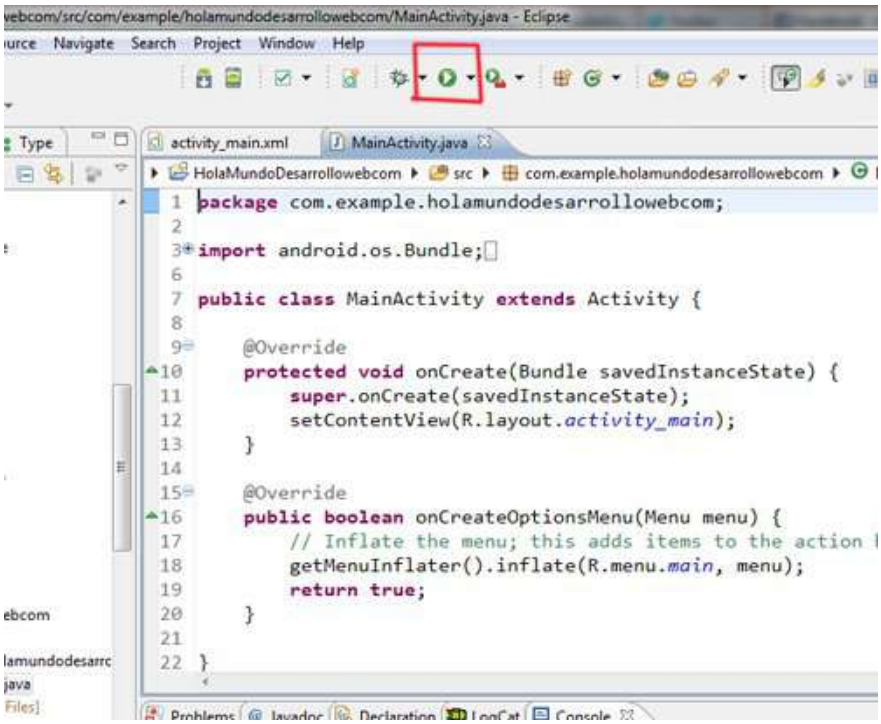
Layout Name: El nombre de la maquetación, es decir, el nombre de la parte gráfica de la primera pantalla. Podemos dejar el que tiene.

Navigation Type: El tipo de navegación que queremos usar, por ahora lo dejamos como "none", ya que lo

veremos más adelante en detalle.

Ahora le damos a "Finish", después de esto ya creamos nuestro primer proyecto y nos abre la ventana en donde vamos a encontrar nuestra clase principal del proyecto. Más adelante nos dedicaremos a entender todo claramente.

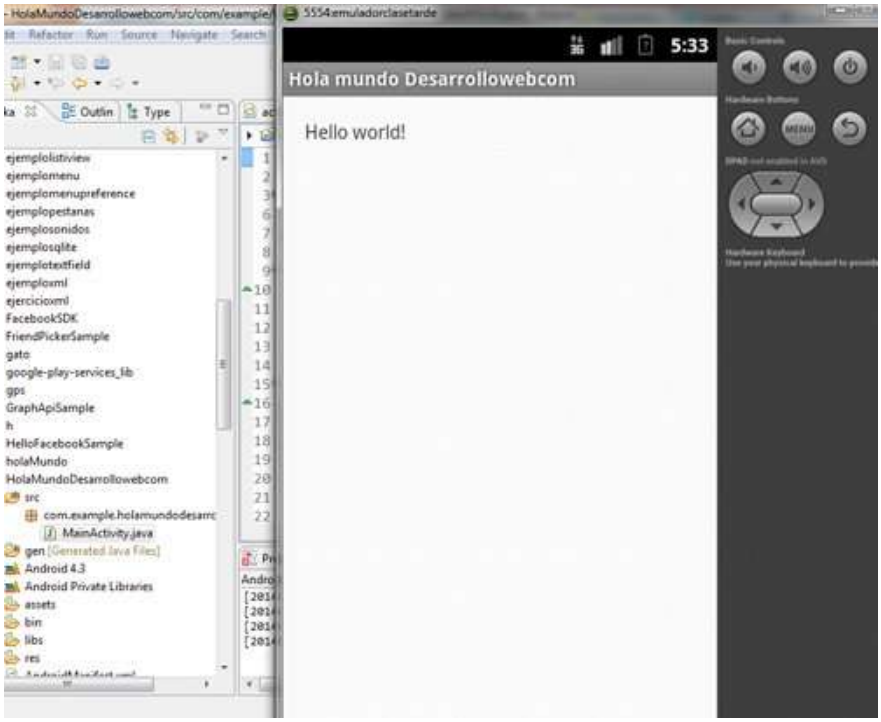
Para ver nuestro “Hola mundo” en nuestro entorno Eclipse vamos a la parte superior. Aparecerá un botón de color verde parecido al "play" de los aparatos electrónicos que sirve para ejecutar los proyectos.



Le damos clic y luego seleccionamos "Ejecutar como Aplicación Android" o “Android Application” en inglés y le damos a OK.

Si configuramos previamente nuestro emulador, este debe ejecutar solo el último emulador que usamos o el único que tenemos creado.

Listo, tenemos nuestro ¡Hola mundo!



Artículo por José Dimas Luján

¿Qué es una Activity?

El término Activity (actividad) es de los primeros que debemos entender a la perfección en Android, ya que es de lo más básico y se usa muchísimo en las aplicaciones.

Podemos decir que todas las pantallas de una aplicación son una "activity". Más adelante vamos a ver que existen algunas variaciones, pero por ahora digamos que todas lo son. Es decir, que si una aplicación tiene cinco pantallas, tiene 5 "Actividades" o *activities*.

Las activities están conformadas por dos partes: la parte lógica y la parte gráfica.

La parte lógica es una archivo .java que es la clase que se crea para poder manipular, interactuar y colocar el código de esa actividad.

La parte gráfica es un XML que tiene todos los elementos que estamos viendo de una pantalla declarados con etiquetas parecidas a las del HTML, es decir, que el diseño de una aplicación en Android se hace similar a una página web; XML es un primo de HTML.

Resumiendo, una actividad está conformada por la parte lógica (un archivo Java) y la parte gráfica (un archivo XML).

Adentrando más en el tema, ya sabemos que tenemos un archivo .java, esto quiere decir que tenemos una clase principal, al ser una actividad extiende de la clase Activity (por eso el nombre) que nos proporciona Android para crear actividades con sus métodos asignados.

Veamos una actividad básica:
Este sería nuestro archivo "ManActivity" como el del ejercicio del "HolaMundo".

```
package com.example.holamundodesarrollowebcom;

import android.os.Bundle;
import android.app.Activity;


public class MainActivity extends Activity {

          @Override
          protected void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.activity_main);
          }



}
```

Expliquemos por líneas:

```
package com.example.holamundodesarrollowebcom;
Esta línea únicamente indica el nombre del paquete en donde se encuentra nuestra clase.
import android.os.Bundle;
import android.app.Activity;
```

Los dos *imports*, son la forma de decir que necesitamos esos archivos para trabajar dentro de la clase, los que ya nos da Android para no tener que escribir las cosas desde cero.

```
public class MainActivity extends Activity {
          @Override
          protected void onCreate(Bundle savedInstanceState) {
                    super.onCreate(savedInstanceState);
                    setContentView(R.layout.activity_main);
          }

}
```

En esta última sección de código lo que estamos haciendo es crear una clase que se llama "MainActivity" y la estamos extendiendo de acvitity, en español esto es el concepto de herencia de la famosa programación orientada a objetos, estamos diciendo que "mainactivity" es una clase que hereda las cosas de la clase Activity que ya tiene Android definida.

Todas las *activities* deben llevar por lo menos un método, el método "oncreate", que es en donde se crea la actividad o podemos decir que es donde se le da vida.

Del método "onCreate" lo más importante es la línea de código: SetContentView(R.Layout.acivity_main).

Que es la que hace el trabajo de enlazar la parte lógica con la parte gráfica. El archivo XML que va a mostrarse cuando se mande a llamar la clase "MainActivity" es el archivo XML llamado "activity_main".

Para cerrar la explicación: si yo creo una actividad nueva y la llamo "VentanaPrinicipal", debo hacer que herede de activity si quiero que funcione como actividad y para decirle que el archivo XML que va a mostrar sea el "ventanaprincipal.xml" o "pepito.xml". La línea que dice "SetContentView" debe llevar dentro algo parecido a esto: "setContentView"(R.layout.ventanaprincipal).

Más adelante vamos a ver como agregar una actividad nueva, ya que además de esto necesitamos hacer otras cosas en el proyecto. Espero por ahora que quede claro el término de Activity que vamos a estar utilizando.

Artículo por José Dimas Luján

Estructura de una aplicación Android

Explicamos la estructura de una aplicación Android, con los archivos y directorios que debemos tener en cuenta al realizar un proyecto.

Tomando como base el “Hola Mundo” que hicimos, explicaremos la estructura del proyecto y todos los archivos que tiene.

src– Aquí van las clases de nuestra aplicación, es decir los archivos .java.

gen- Son archivos que genera Java y por ninguna razón los debemos tocar. Si lo hacemos, ya no van a servir y puede que ni el proyecto sirva para más adelante. Cada vez que compilamos, Java se encarga de actualizarlo y de generarlo de nuevo. Dentro de gen encontramos 2 archivos: el BuildConfig y R. El archivo R es el archivo que tiene los identificadores de todo lo que tiene la aplicación, por ejemplo imágenes, campos de texto, botones, etc. Java le asigna un identificador y nosotros no tenemos que preocuparnos por él, ya que le colocamos un nombre común que podamos recordar y Java sabe cómo se llama para nosotros.

assets- Este directorio contiene recursos de ayuda para la aplicación, audio, videos, bases de datos, la carpeta "assets" y la carpeta "res" sirven ambas para guardar recursos, pero la diferencia es que los que se encuentran en "assets" no generan un identificar en el archivo R que vimos se encuentra en el directorio "gen".

bin- Aquí tenemos archivos generados por el mismo Java, que en realidad no los utilizamos y tampoco debemos manipular, son archivos binarios como bien dice su nombre.

libs- Se encuentran librerías externas que necesita el proyecto.

res- El directorio "res" contiene todos los recursos de la aplicación.

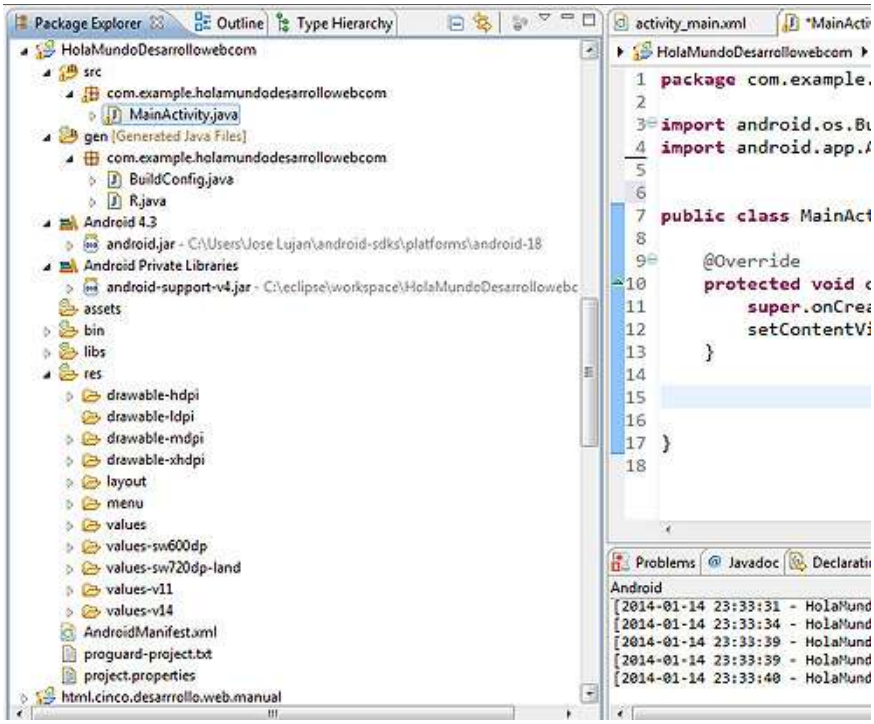
res/drawable- Contiene todas las imágenes y gráficos PNG que vamos a incluir en nuestra aplicación. Cada uno

representa una densidad, más adelante tendremos un capitulo para este tema en específico.

res/layout- En este directorio colocamos todos los XML que son la parte gráfica de nuestras "activities", es decir, todos los XML que son las pantallas de nuestra aplicación.

res/values- Se encuentran archivos con cadenas de texto que usamos en nuestra aplicación, algunos estilos de nuestra aplicación.

AndroidManifest.xml- El archivo Manifest es el más importante para nuestra aplicación, es la columna vertebral de nuestro proyecto, en él declaramos todas las actividades del proyecto, los permisos, versiones del SDK que usamos y un montón de cosas que vamos a ver más en detalle.



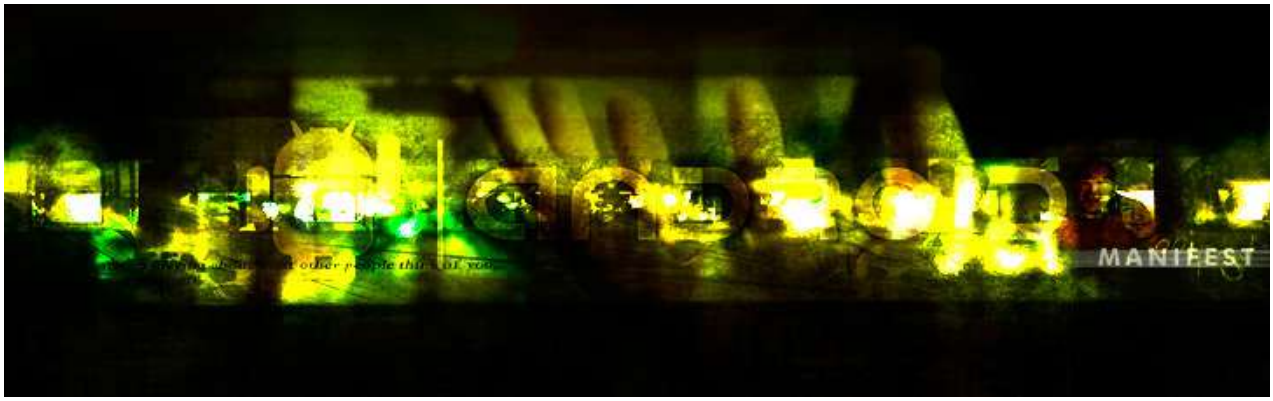
Artículo por José Dimas Luján

Archivo Manifest

El Archivo Manifest es la parte más importante en una aplicación Android. En este capítulo lo vemos en detalle.

Antes ya explicamos de forma general el Manifest, pero es bueno darle una pasada en detalle. Recuerda que es el archivo más importante de una aplicación Android: si éste no se encuentra, nuestra aplicación no va a funcionar.

Tomamos como ejemplo el Archivo Manifest de nuestra aplicación "Hola Mundo" que hicimos anteriormente.




```
1<?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="com.example.holamundodesarrollowebcom"
4    android:versionCode="1"
5    android:versionName="1.0" >
6
7    <uses-sdk
8        android:minSdkVersion="8"
9        android:targetSdkVersion="18" />
10
11    <application
12        android:allowBackup="true"
13        android:icon="@drawable/ic_launcher"
14        android:label="@string/app_name"
15        android:theme="@style/AppTheme" >
16        <activity
17            android:name="com.example.holamundodesarrollowebcom.MainActivity"
18            android:label="@string/app_name" >
19            <intent-filter>
20                <action android:name="android.intent.action.MAIN" />
21
22                <category android:name="android.intent.category.LAUNCHER" />
23            </intent-filter>
24        </activity>
```

```
25    </application>

26 </manifest>
```

La primera linea Eclipse la coloca por defecto por nosotros, solamente indica la versión y el formato del documento.

Etiqueta Manifest

Etiqueta <manifest> es la etiqueta principal. En ella tenemos unos atributos:

Atributo xmlns: siempre lo coloca Eclipse igual, así que no tendremos por qué moverlo.

Atributo Package: hace referencia al nombre del paquete de la aplicación, así que si cambiamos el nombre de la aplicación, este lo debe cambiar o cambiarlo manualmente nosotros.

Atributo VersionCode: esto es el número de versión del código de la aplicación y nos sirve para saber qué version estamos trabajando; la idea es manejarlo por número de publicación, es decir, que cada vez que vamos actualizar para publicar la *app*, deberíamos cambiarlo.

Atributo VersionName: se trata del número de versión también, pero a diferencia del anterior, este número de versión es el que se muestra en la Playstore. Podemos decir que el "versioncode" es para el desarrollador y el "versionname" es para el público en general.

Etiqueta SDK

Aquí vamos a colocar parámetros que sirven para saber la compatibilidad de las versiones en Android.

Atributo minSDKVersion: indicamos la versión mínima de SDK con la que va a funcionar la aplicación. Cuando creamos el proyecto lo indicamos al iniciarlo, pero se puede cambiar manualmente.

Atributo targetSdkVersion: esta es la versión de la API a la que se dirige principalmente nuestra *app*.

Etiqueta Application

Dentro de este etiqueta tenemos todos los elementos que forman parte de nuestra aplicación.

Atributo allowBackup: si colocamos en "true" este valor, estamos diciéndole al sistema que se permite hacer copia de seguridad de la aplicación y contenido.

Atributo icon: es el icono de la aplicación.

Atributo label: es el nombre de la aplicación.

Atributo theme: es el tema que estamos seleccionado para los estilos de nuestra aplicación.

Etiqueta Activity

La etiqueta Activity sirve para dar de alta todas las actividades que creamos en nuestra aplicación; recuerda que si tenemos cinco actividades, debemos tener cinco etiquetas activity, pero cada una con sus parámetros.

Atributo name: este es el nombre de la clase Java que implementa el Activity.

Atributo label: este es el nombre de la Activity que el usuario va a ver.

Etiqueta Intent-filter

Esta etiqueta sirve para especificar qué es lo que tiene permitido hacer esta activity.

Etiqueta action

Esta etiqueta nos permite colocarle un nombre para más adelante, dentro del código, mandarla a llamar de ese modo.

Etiqueta Category

Esta etiqueta siempre tiene dos opciones:

```
<category android:name="android.intent.category.LAUNCHER" />
```

Usamos Launcher solo una vez dentro de todo el Manifest. Con esto le indicamos que es la actividad principal, la que se va a lanzar cuando la aplicación se inicie, digamos que es nuestra Activity principal.

```
<category android:name="android.intent.category.DEFAULT" />
```

Todas las demás actividades llevan el "DEFAULT" ya que solo le indica que se ejecuta, como solo tenemos una activity, tenemos un launcher. En caso de tener nueve activities, tendríamos ocho Default y un Launcher.

Con esto ya estudiamos nuestro Manifest en detalle. Esta es la estructura básica de nuestra aplicación, pero tenemos muchas más opciones para agregar a nuestra aplicación, que vamos a ir conociendo conforme vayamos avanzado.

Artículo por José Dimas Luján

Componentes en Android

En Android tenemos unos componentes básicos que debemos conocer, saber qué son y en qué nos pueden servir.

Así sabremos qué es lo que nos va a ser de utilidad para cada aplicación que afrontemos.
Tendremos seis componentes que podemos utilizar:

Activity

Las actividades son las pantallas que vemos de un móvil, que está conformada por dos archivos, un archivo XML (parte gráfica) y un archivo .Java (la parte lógica) de nuestra ventana.

Services

Los servicios son tareas que se realizan en segundo plano y no contienen una interfaz de vista para el usuario.

Content Provider

El Proveedor de contenido nos sirve para proporcionar datos a nuestra aplicación, esto nos ayuda a compartir datos sin preocuparnos por el mecanismo de almacenamiento, lo que nos importa es la información que nos puede proporcionar para una o varias tareas que necesitan los datos en la aplicación.

Intent

Los *intents* son objetos que sirven para mandar mensajes. Además, se puede pedir ejecutar una acción a otro componente con el mismo intent. Dentro de los intents existen diferentes tipos dependiendo de lo que necesitemos.

Widget

Los widgets son "pequeñas" aplicaciones que reciben actualizaciones periódicas, por ejemplo en algunos móviles tenemos *widgets* del clima en nuestro móvil, sin necesidad de entrar a la *app* tenemos la opción de colocar el widget en una de nuestras ventanas principales en Android y lo mismo sucede con algunos reproductores de música, etc. La idea es que el usuario pueda hacer cosas sin entrar directamente en la aplicación.

Processes and Threads

Cuando una aplicación se inicia, lo que sucede de manera muy técnica, es que se crea un proceso. Entonces todo lo que se ejecuta dentro de esa aplicación es parte del mismo proceso de ejecución. Un hilo es otra forma de dividir el trabajo, un proceso puede tener varios hilos de ejecución, el hilo lo podemos tomar como una secuencia de control dentro de cualquier proceso y este ejecuta lo que indiquemos de manera independiente.

Estos son, a grandes rasgos, los componentes que podemos encontrarnos dentro de una aplicación Android.

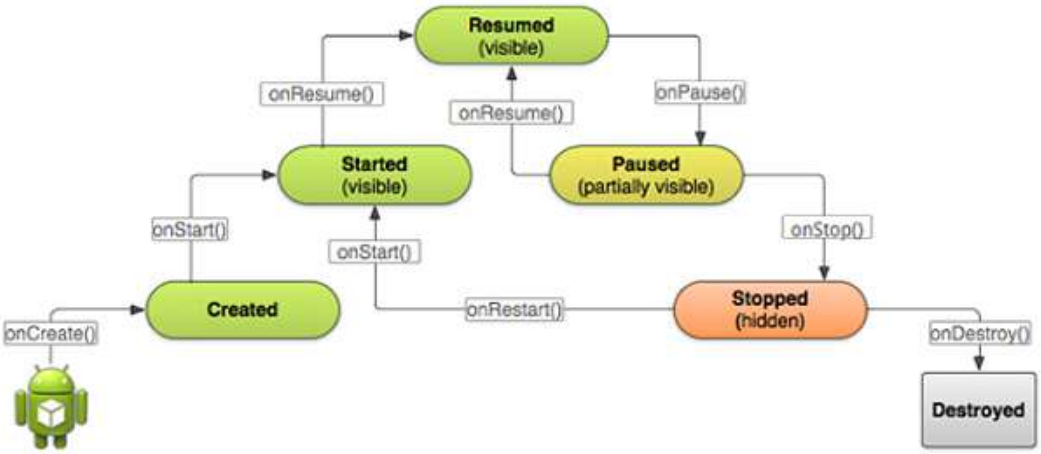
Artículo por José Dimas Luján

Ciclo de vida de una actividad

El ciclo de vida de una "activity" o actividad en Android es de lo primero que debemos conocer si queremos hacer el uso correcto de la misma en nuestra aplicación.

Continuamos con el Manual de desarrollo para Android con un artículo que nos debe aclarar algunas de las bases que debemos conocer perfectamente para dominar el desarrollo de Apps para este sistema operativo. No es otra que el ciclo de vida de una activity.

A diferencia de muchos sistemas o aplicaciones que se inician con un método "main()", que si vienes del mundo Java o C++ me imagino que te sonará, aquí la secuencia es diferente y es como se muestra en la imagen:



Tratare de explicar la imagen: lo que está encerrado en óvalos son lo que llamaremos “estados” y para llegar a un estado tenemos que pasar por un método o más, en la imagen se muestran los métodos y son los que tiene el prefijo “on”. Entonces, por ejemplo:

Ejemplo 1: Para llegar al estado "Created" de una actividad, pasamos por el método "onCreate()".

Ejemplo 2: Para pasar al estado "Started" desde el estado "Stopped" pasamos por los métodos "onRestart()" y "onStart()".

Espero que con esto quede claro cómo interpretar la imagen.

En Android se inicia con la invocación de métodos específicos que corresponden a etapas de vida. Tenemos una secuencia de métodos que devuelven una llamada e inician la actividad, también tenemos una secuencia de métodos que terminan la actividad.

Como ven en la imagen, tenemos una pirámide, esta primera inicia por la izquierda con el "onCreate()" y termina del lado derecho con el "onDestroy"; vamos del principio al final.

Que quede claro que no siempre utilizamos los métodos del ciclo vida, depende de lo que vamos a necesitar y cuándo lo vamos a necesitar, hay aplicaciones donde se aplican todos, ninguno o solo un par.

Tenemos los métodos:

- **onCreate()**
- **onStart()**
- **onResume()**
- **onPause()**
- **onStop()**
- **onRestart()**
- **onDestroy()**

onCreate()

Es el que debemos ejecutar en un inicio para definir, por ejemplo, la interfaz del usuario y también crear algunas variables de ámbito de la clase. Este método por lógica solo se debería ejecutar solo una vez, al momento de invocar la actividad. En este método casi siempre vamos a encontrar cómo se define un archivo XML, como la parte gráfica de la actividad o la configuración de la interfaz.

Cuando el método "onCreate()" termina de ejecutarse llama al método "onStart()" y "onResume()", esto sucede de manera muy rápida.

Técnicamente, la actividad se vuelve visible para nuestro usuario cuando llamamos en "onStart()", pero como

sigue muy rápido al "onResume()" y se mantiene en ese estado hasta que suceda otra cosa. Por ejemplo, cuando se apaga la pantalla o cuando vamos a otra actividad, también en caso de recibir una llamada telefónica.

onStart()

Es donde la actividad se muestra ya al usuario como comentamos anteriormente.

onResume()

Es el estado en donde se encuentra en primer plano y el usuario interactúa con la actividad, podemos decir en español que es el estado “corriendo” o “ejecutando”.

onPaused()

Es cuando esta se encuentra parcialmente oscurecida por una actividad que se encuentra en el primer plano, por ejemplo está medio transparente o no cubre toda la pantalla, en este estado no se reciben datos de entrada del usuario y no puede ejecutarse código.

onStop()

En este estado se encuentra completamente invisible u oculto para el usuario, podemos decir que se encuentra en el “fondo”, en este estado podemos decir que todo se congela, por ejemplo las variables e información se mantiene pero no podemos ejecutar el código.

onRestart()

Este método se llama después del "onStop()" cuando la actividad actual se está volviendo a mostrar al usuario, es decir, cuando se regresa a la actividad. Después de este continua el "onStart()" y luego en "onResume()" y finalmente ya está de nuevo mostrándose la actividad al usuario.

onDestroy()

Cuando el sistema destruye su actividad se manda a llamar al método "onDestroy()" para la actividad. Este método es la última oportunidad que tenemos de limpiar los recursos y que si no eliminamos podrían no tener un buen rendimiento para el usuario en caso de olvidarlo. Es buena práctica asegurarse de que los hilos que creamos son destruidos y las acciones de larga duración también estén ya detenidas.

Con esto finalizamos este capítulo y toda la información que se plasma en este manual de Android sale de la documentación oficial: developer.android.com

Artículo por José Dimas Luján