

ingenio

informática

interacción

internet

ideas

inversión

imaginación

impresionante

inspiración



ideal

inteligencia

impulso

imagen

innovación

integridad

instituto

increíble

información

Motivación



6 claves para el desarrollo de apps

Optimizar la interfaz grafica y hacerla usable:

- Además debe ser atractiva, no olvidar lo importante de la primera impresión
- Evitar la sobrecarga de elementos y apuntar a que sea intuitiva



Interacción sencilla:

- Simplificar la vida del usuario hacer que interactúe lo menos posible, por ejemplo con el uso de periféricos y sensores por ejemplo

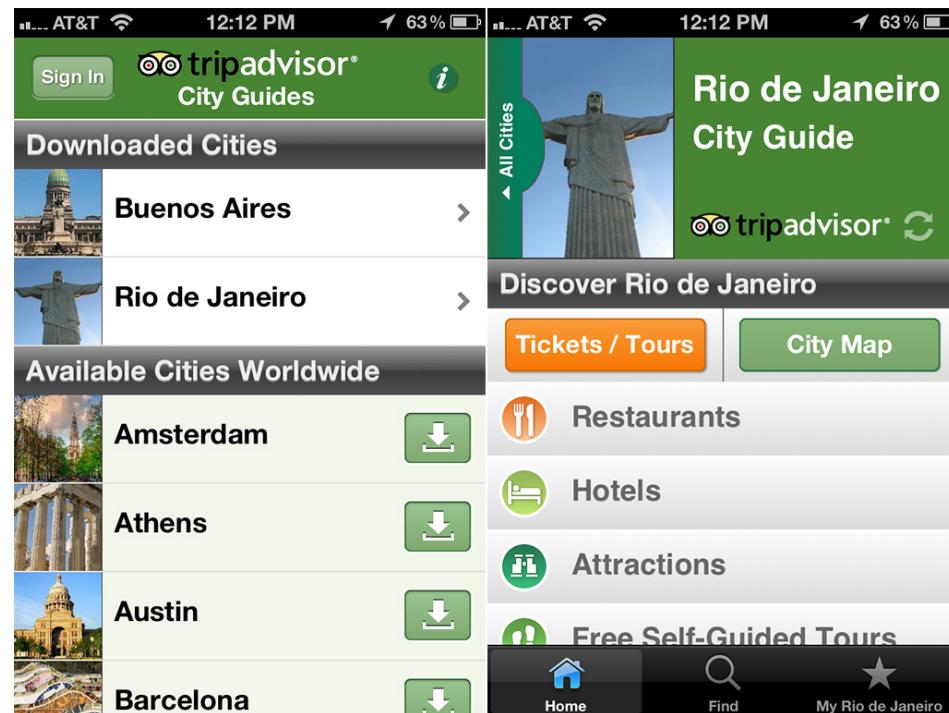
Ejemplo:

(juego con acelerómetros en vez de flechas)



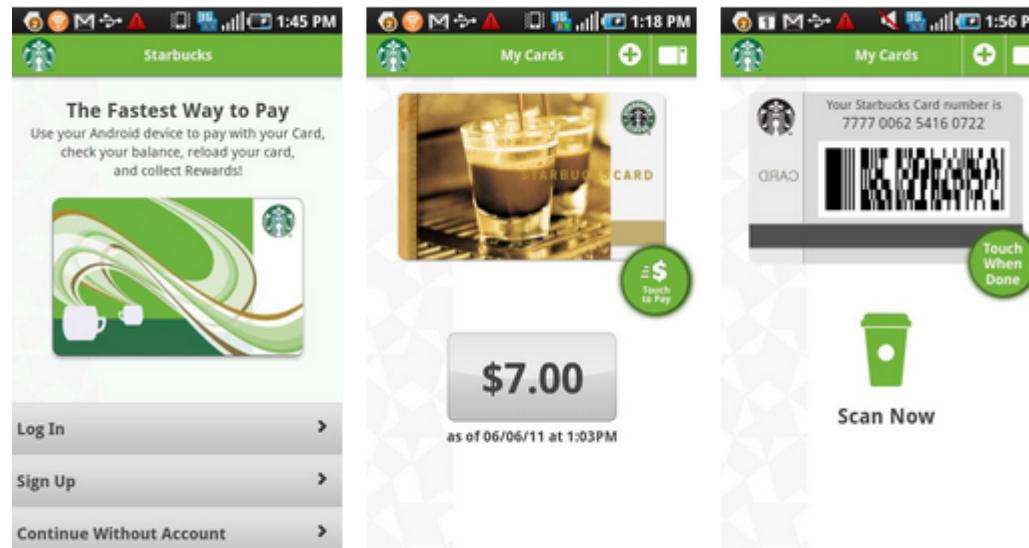
Aprovechar el contexto del usuario:

- El contexto de usuario hace referencia a los datos del mismo.
- Esto puede ayudar a darle un servicio mas personalizado



Aportar Valor:

- Es decir porque los usuarios van a usar nuestra app.
- Que les ayuda ha hacer?
- Facilita alguna tarea en el día a día?
- Los hace mas productivo?
- Que hace la app que no haga otra o en que mejora eso?
- Cubrir una necesidad



Modelo de negocio alrededor de la aplicación:

Como haremos dinero con la aplicación:

- Por encargo de una empresa
- Subir a google play y obtener dinero en base a publicidad en la aplicación, ofreciendo la aplicación gratis
- Vender la aplicación en el google play
- Freemium
- Gamification

8 Claves para analizar apps

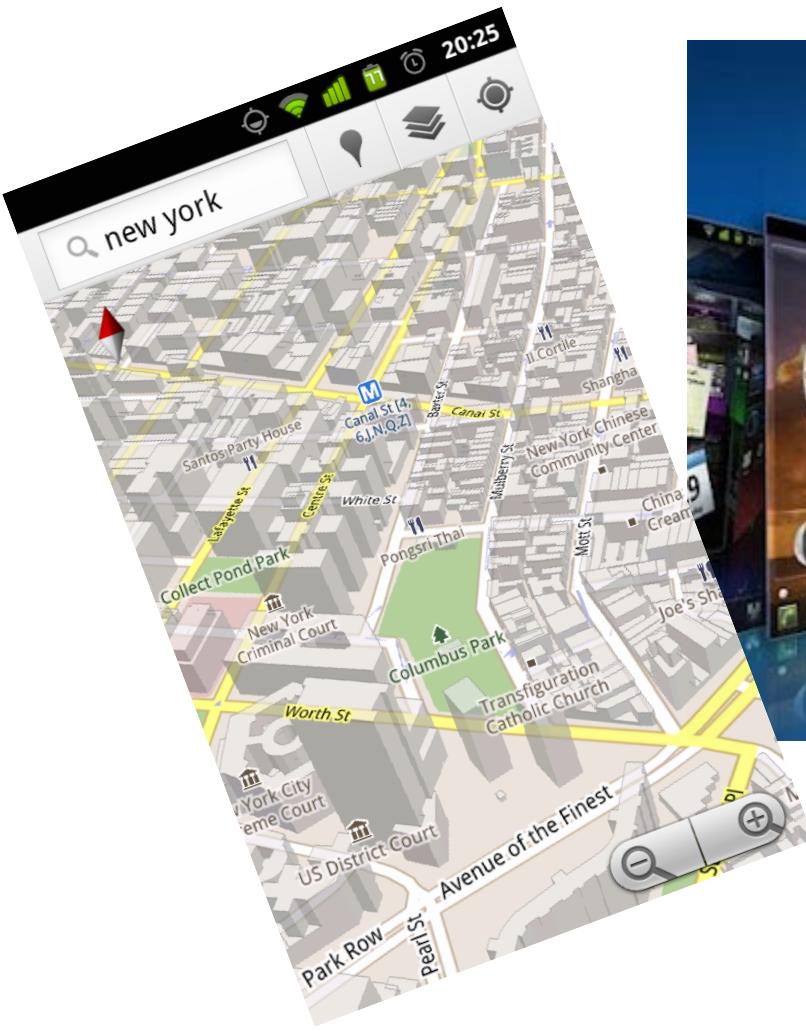
- **1. Tipo:** Será un juego, aplicación de negocios, turismo, etc



- **2. Audiencia:** a que tipo de usuario estará orientada la aplicación.



- **Fortalezas:** acelerometro?, geolocalizacion?, el diseño?



- **4. Dispositivo:** esta diseñada para un smartphone o para un tablet?.



- **5. Momento de uso / manos:**

- Donde van a utilizar la app, en la casa, caminando, en el tren, durante un viaje?. Tener en cuenta como diseñar la interfaz de acuerdo al momento de uso, por ejemplo: si la usara caminando esta debe poder ser manejada con una sola mano o incluso con un asistente de voz.

- **6. Tiempo de utilización:**

- Periodos cortos o en periodos largos.
- Es decir cuanto tiempo estará el usuario en la aplicación.

- **7. Internet:**

- Necesitara la aplicación de conexión a internet todo el tiempo?
- Ver una política para evitar el consumo constante de la red por ejemplo la utilización de bases de datos en memoria del lado del cliente.

- **8. Mejorar:**

- Que podemos mejorar o agregar a la app, sincronizacion con redes sociales, utilizacion de la camara, gps, etc.

Curso Android V3.0



ANDROID

Bienvenida e Introducción

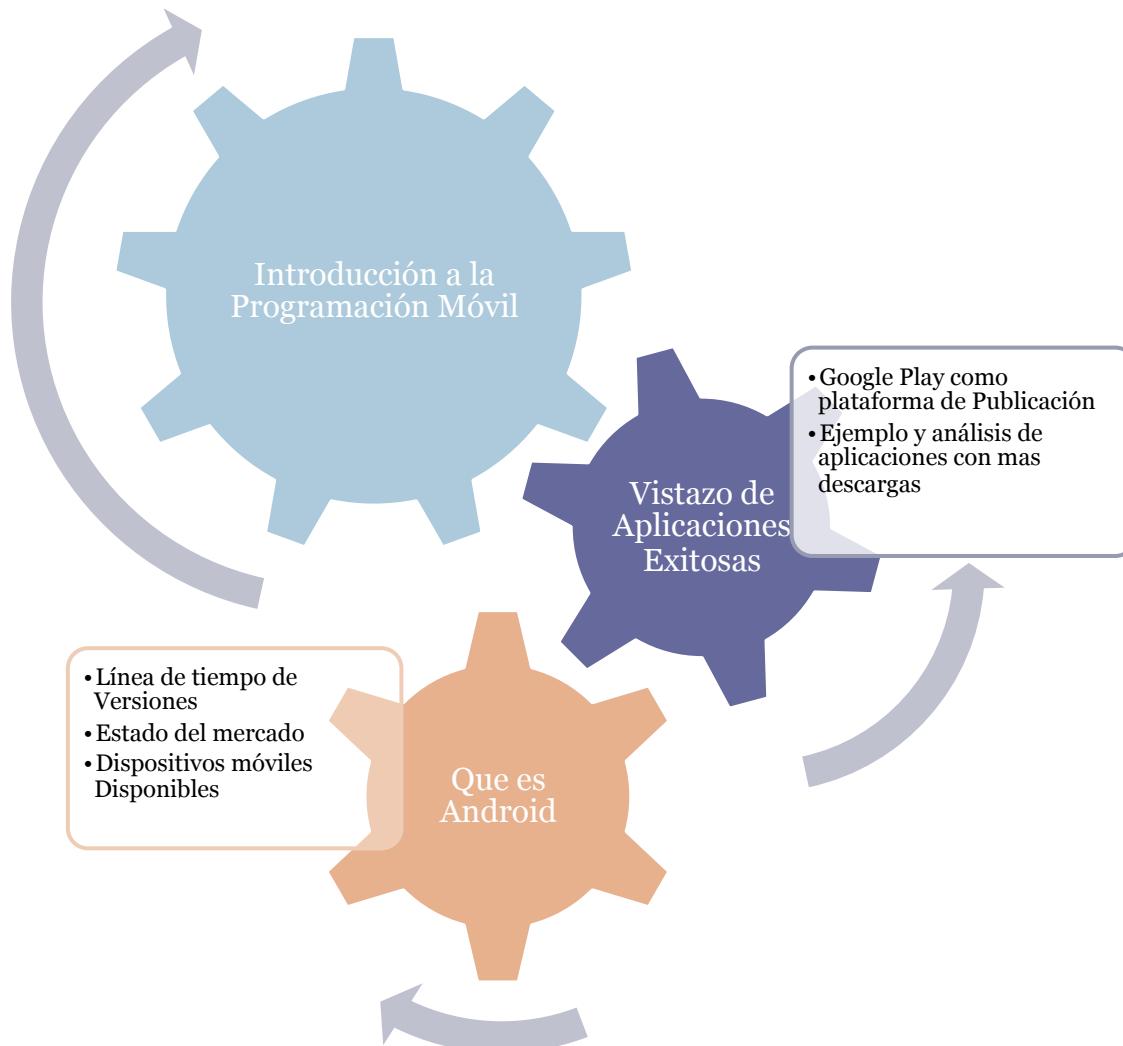
Emiliano Gonzalez

Pedro Coronel

2013



Contenido



Introducción

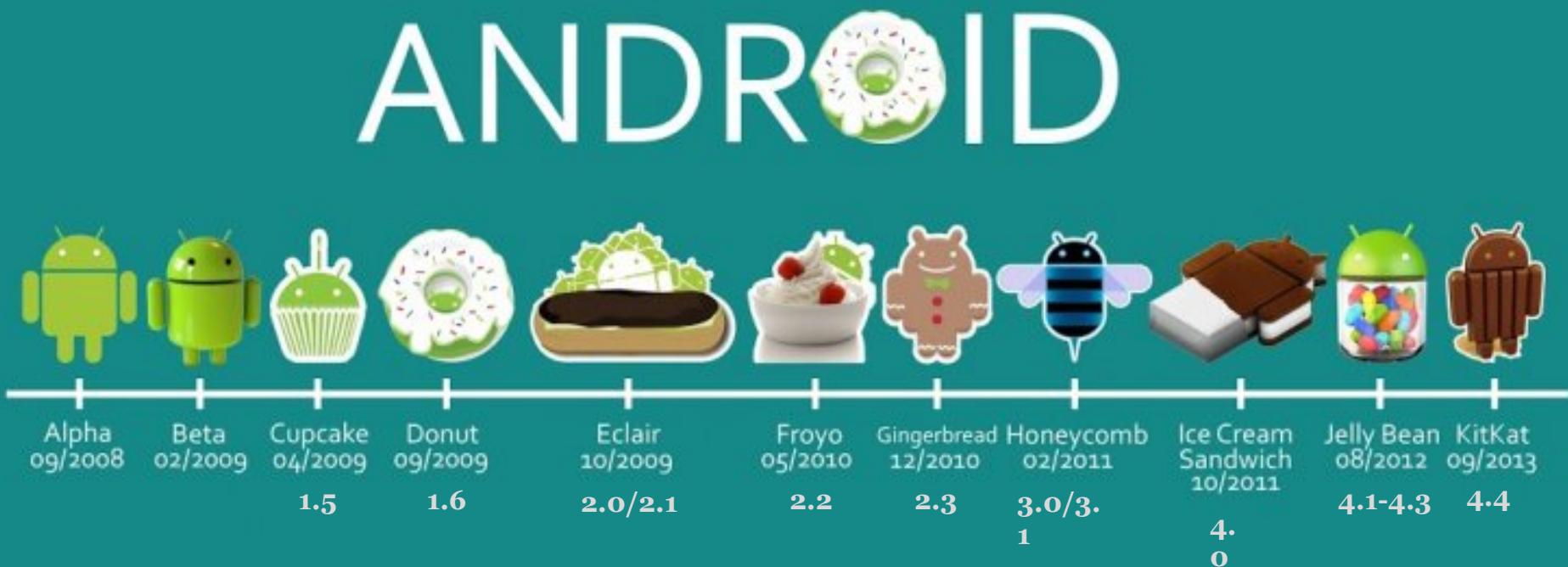
- A principio de año siempre nos **proponemos** nuevas metas y nuevos retos que obtener. Si uno de tus retos ha sido aprender un lenguaje nuevo, **Android** es hoy en día una de las **plataformas** más atractivas para las personas que desean aprender una tecnología en **vanguardia**.
- Con el Curso de Programación en Android aprenderás a crear desde cero tus propias aplicaciones para dispositivos móviles con sistema operativo Android. Desde los temas más básicos, como descargar e instalar las herramientas necesarias o crear tu primer proyecto paso a paso, hasta temas avanzados como la localización GPS o la utilización de base de datos SQLite



Que es Android?

- Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS de Apple, Symbian de Nokia y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. Esto permite que muchas empresas ofrezcan sus smartphones con este sistema operativo, así como Samsung, Sony, HTC, entre otros gigantes de los celulares.

Línea de tiempo



Idea? Que vamos a desarrollar?



Para que versión y dispositivo?



Como lo vamos a distribuir?



Desarrollo para dispositivos smart

¿Porqué tener una app para Android?

- Porque el sistema operativo líder del mercado, existen más de 100 equipos disponibles con Android, entre smartphones y tablets, por tanto usted debe elegir si desarrolla una aplicación para el sistema que actualmente es usado por 37% (posiblemente llegue al 50%) a fin de año o se queda con el 25% que usa iOS de Apple en sólo 2 equipos Iphone y Ipad.
- Originalmente el desarrollo de apps en Android ha estado muy orientado a aplicaciones gratis, pero actualmente podríamos decir que el Android Market está en 60% – 40% de aplicaciones de pago vs aplicaciones gratuitas y entre más gente lo use posiblemente se llegué a un porcentaje de 50-50% con el cual no podemos decir que existen oportunidades de negocio en esta plataforma también.



Como esta el mercado hoy?

Sistemas Operativos para Smartphone y mercado actual

MOBILE



OPERATING SYSTEM MARKET SHARE

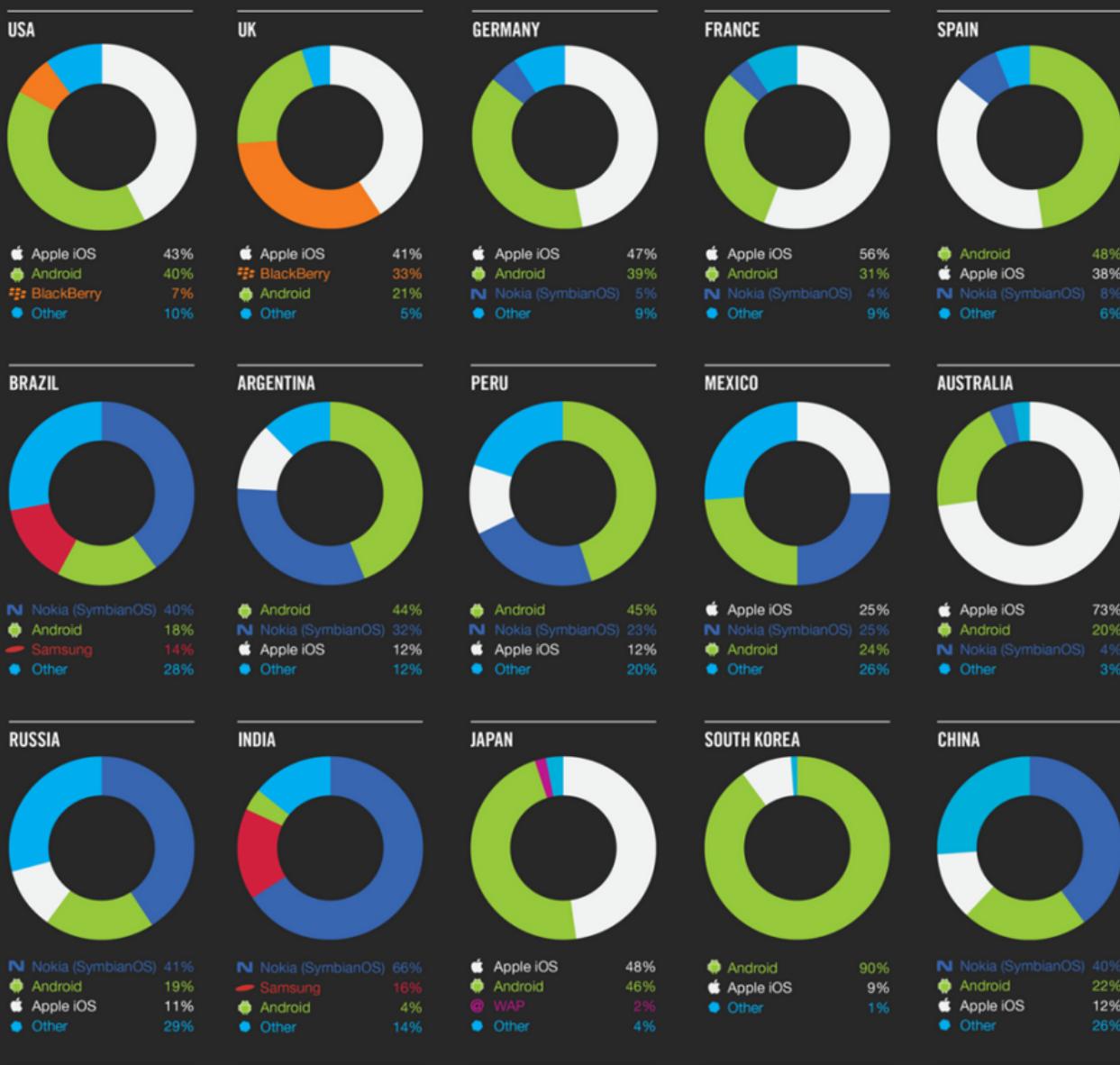
Version 3.0 February 2012



GLOBAL MARKET SHARE AT A GLANCE

The Top 3 ...

- Apple iOS
- Nokia (SymbianOS)
- Android

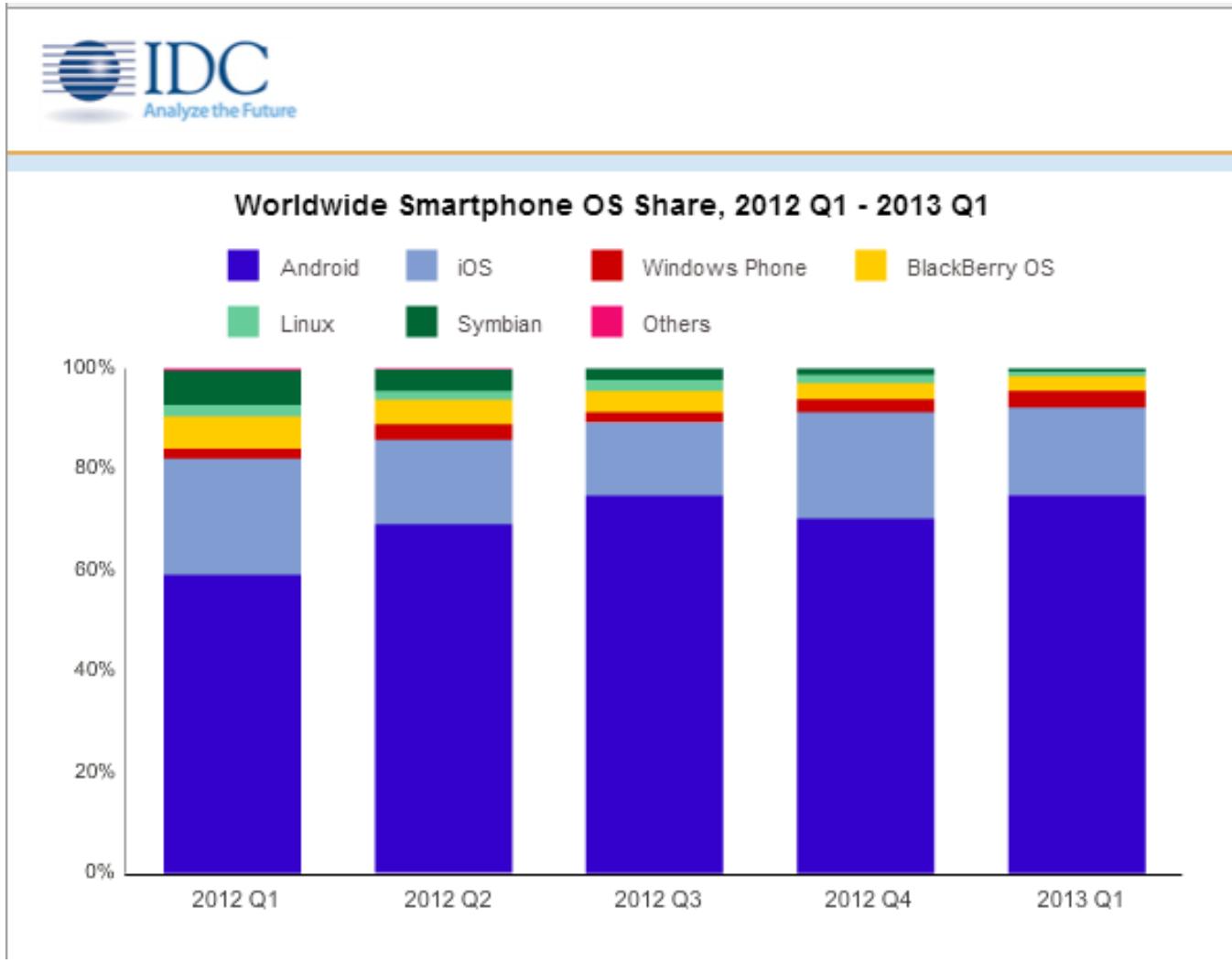


Data Source: <http://gs.statcounter.com/>
 Published Under a Creative Commons Attribution 3.0 Unported License
 You are free to copy, distribute and transmit the work and to adapt the work providing it is
 attributed to www.statcounter.com

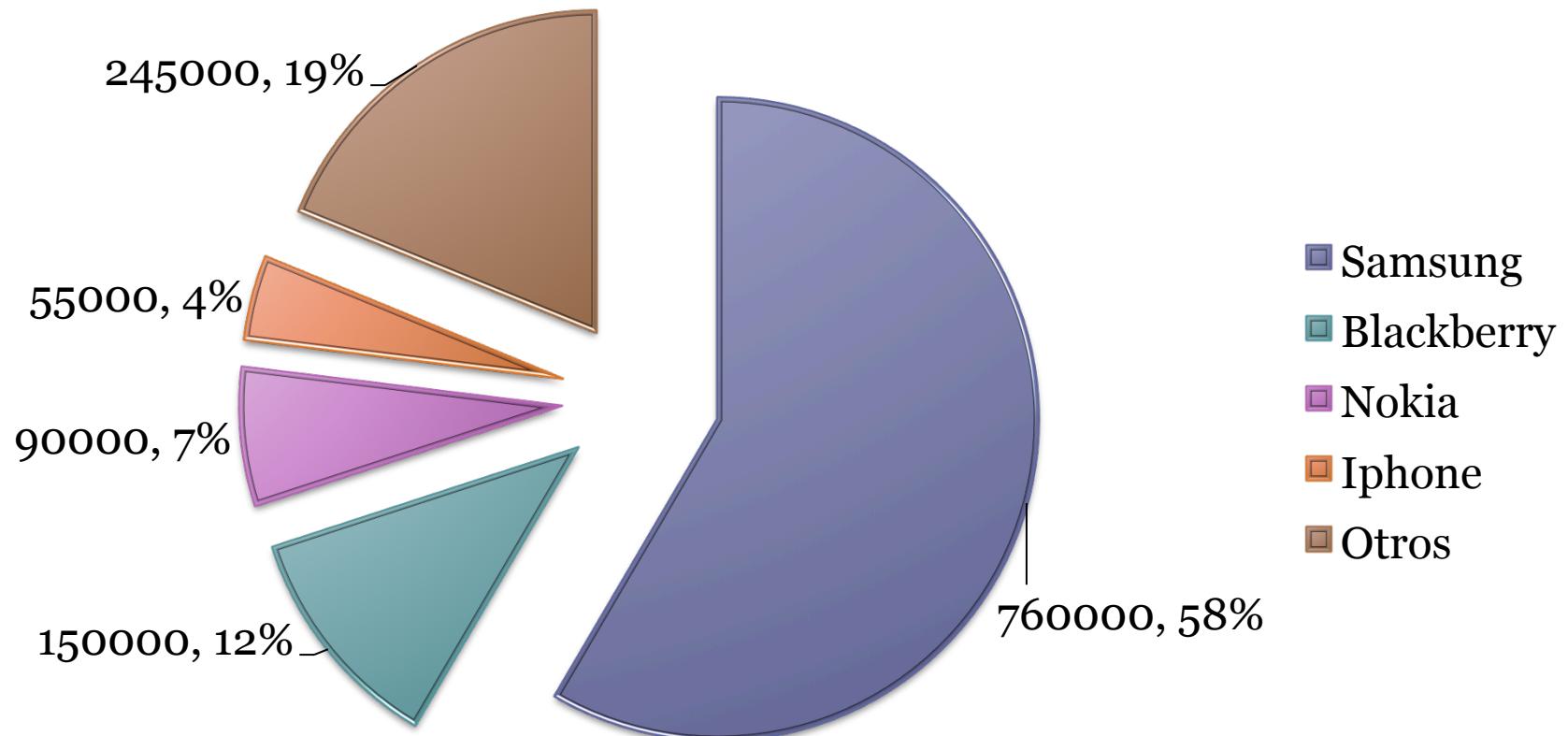
<http://connect.iconnected.co.uk/>

icrossing /:::/

Actualmente como estamos?



Y en Paraguay?



¿Que dispositivos tienen Android?

- El primer error en el que muchos caen es pensar que Android es solo un sistema operativo para celulares. Hoy en día en el mundo de las tablets, Android es el sistema operativo con mayor variedad de modelos y marcas. Y si nos ponemos a observar a las empresas más atrevidas como Samsung, nos damos cuenta que ya están incluyendo Android en dispositivos de SmartTv e incluso en cámaras fotográficas digitales (video)

Google play

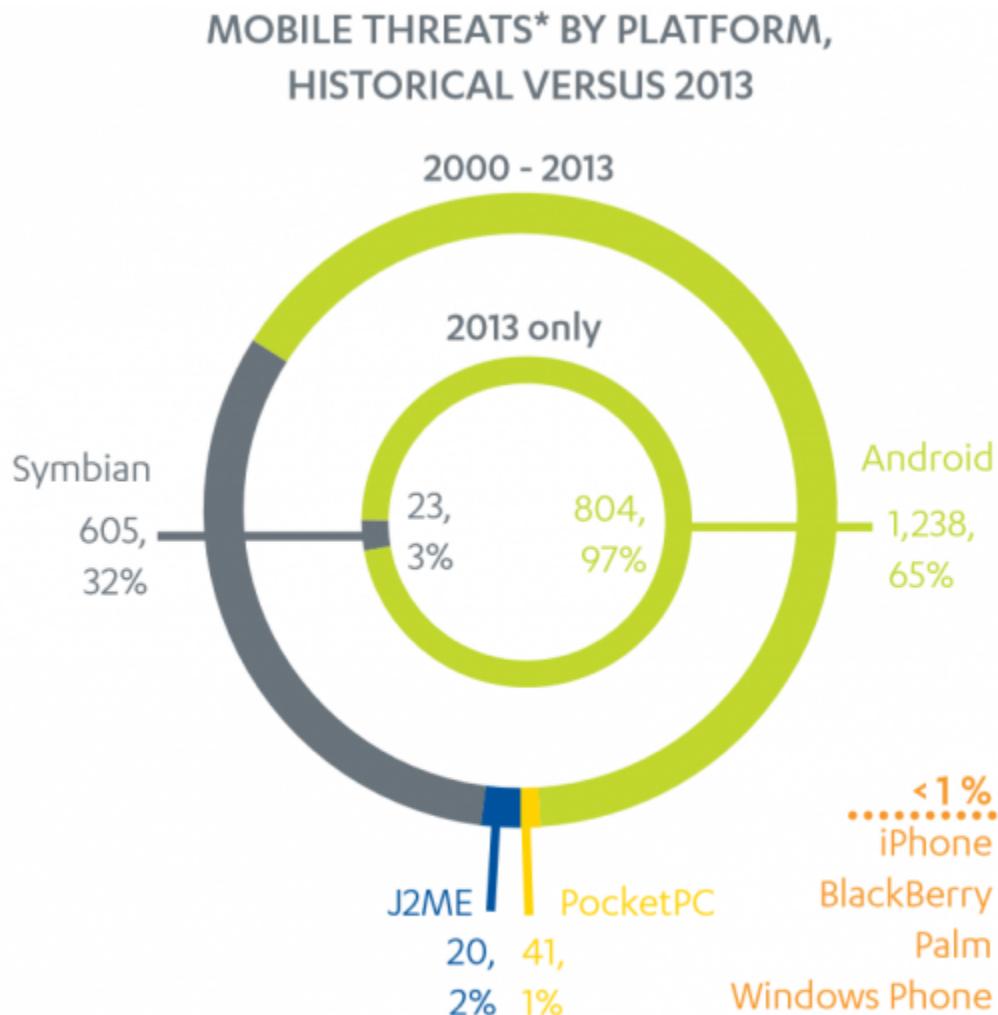




Que necesitamos para publicar?

- **Registro de desarrolladores**
 - Para poder distribuir tus productos en Google Play, tienes que registrarte. Debes pagar una cuota de registro única de 25 dólares. El objetivo de esta cuota es incentivar la entrada de productos de alta calidad en Google Play (y reducir, por ejemplo, los productos no deseados).
- **Tamaño de archivo APK:** el tamaño máximo admitido es 50 MB.
- **Borrador de archivo .apk de aplicación:** al subir un archivo .apk, se puede guardar como un borrador mientras editas el resto de aspectos de la lista.
- **Capturas de pantalla:** son necesarias dos capturas de pantalla. Las seis restantes son opcionales.
- **Icono de aplicación de alta resolución** (obligatorio)
- **Gráfico promocional** (opcional)
- **Gráfico de funciones** (opcional)
- **Vídeo promocional** (opcional)

Un detalle no menor... Los Malwares

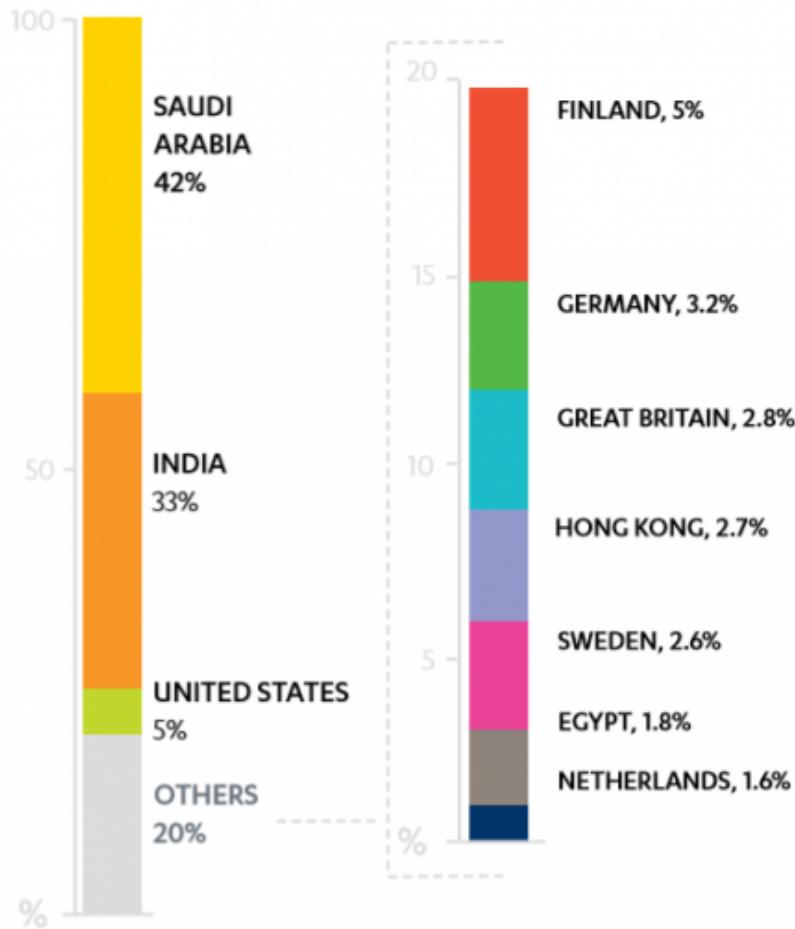


- **Android es el responsable del 79% de todo el malware de dispositivos móviles en 2013**, un aumento en un 18% respecto al año 2012. Eso sí, antes de comenzar a espolvorear Lysoform en tu celular, te reconfortará saber que de esta enorme cifra, **solo el 0,1% correspondía a contenido alojado en Google Play**.

* Count of new families, or new variants of existing families, for all mobile platforms.

Otro dato interesante

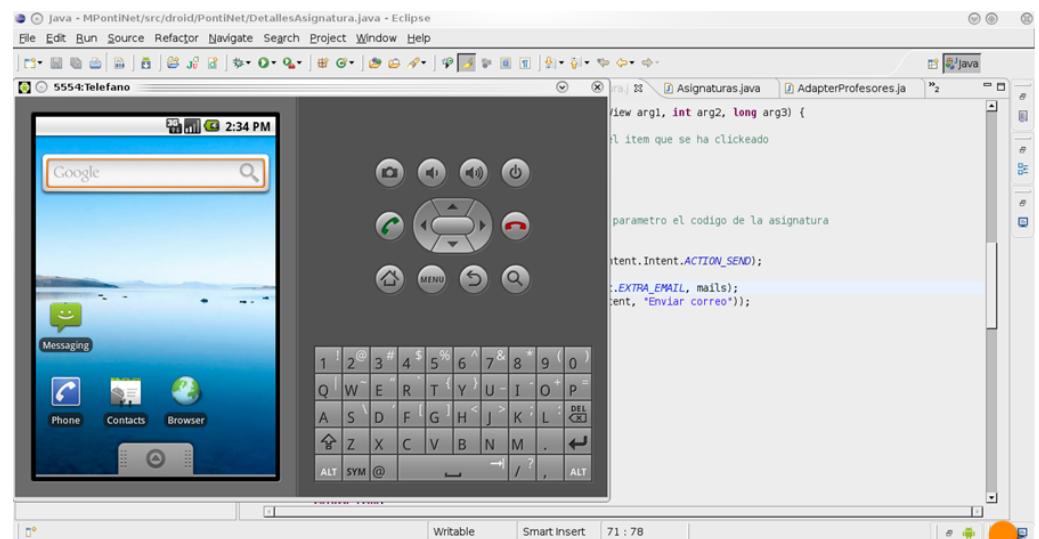
TOP 10 COUNTRIES REPORTING ANDROID
MALWARE DETECTIONS IN H2 2013,
BY PERCENTAGE



- De los diez países que más informaron de detecciones de malware en Android, tres cuartos correspondían específicamente a Arabia Saudita e India. Algo curioso al pensar que en estos países también existe soporte de Google Play.

Framework - Que necesito?

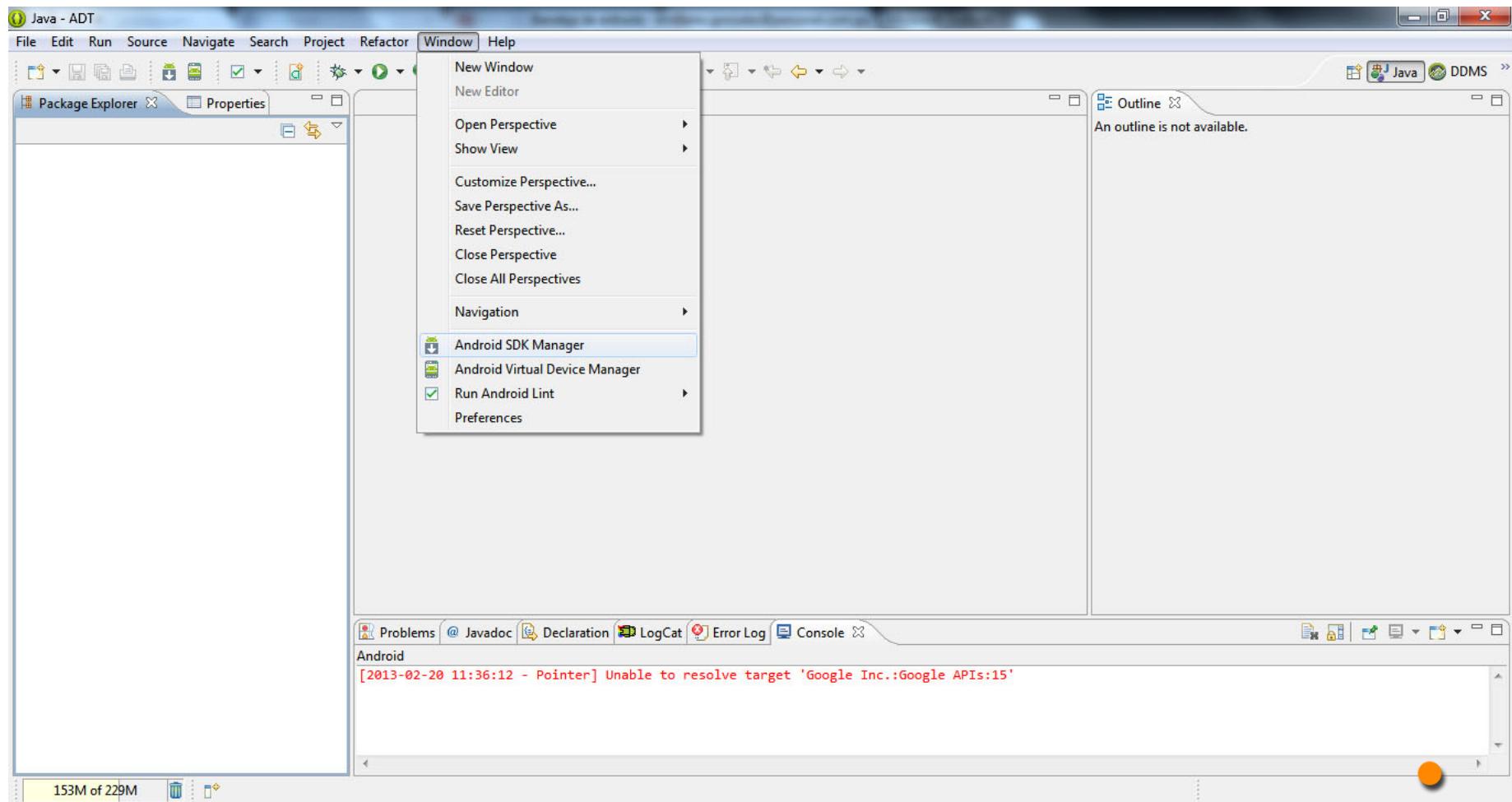
- Eclipse IDE
- Android SDK
- Configurar el Android ADT
- Configurar el Android Virtual Device (Emulador)



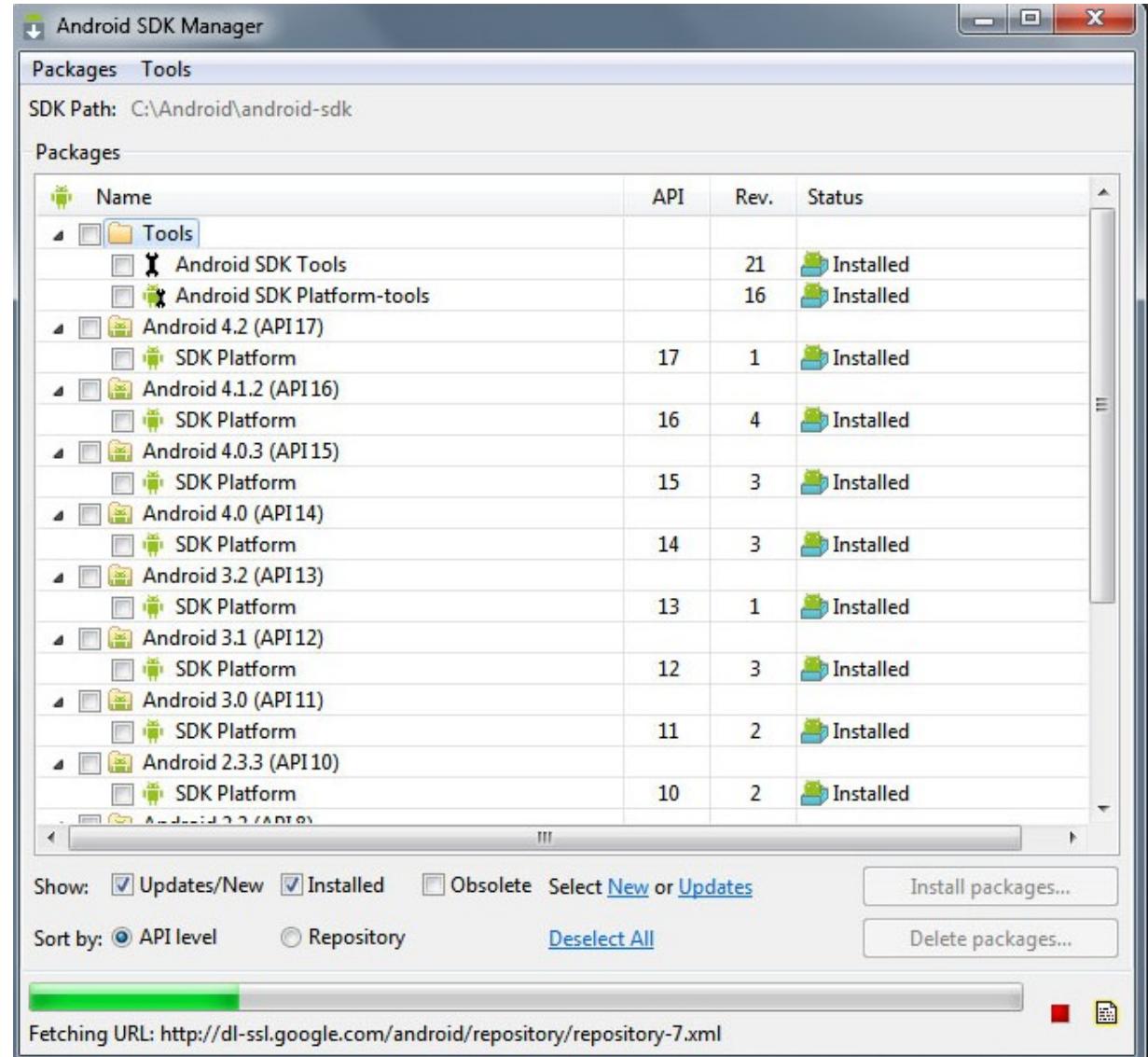
Pasos para instalación

- Paso 1. Instalación de Java (jdk)
- Paso 2. Instalación de Eclipse.
- Paso 3. Intalar el SDK de Android.
- Paso 4. Instalar el plugin Android para Eclipse.
- Paso 5. Configurar el plugin ADT.
- Paso 6. Descargar los targets necesarios.
- Paso 7. Configurar un AVD.

Interfaz del Eclipse



Android SDK Manager



Android Virtual Device Manager (AVD)

Android Virtual Device Manager

Android Virtual Devices Device Definitions

List of existing Android Virtual Devices located at C:\Users\gonzaemi\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
gingerbread	Android 3.2	3.2	13	ARM (armeabi)
Jellybean	Android 4.2	4.2	17	ARM (armeabi-v...)
GoogleL7	?	?	?	?

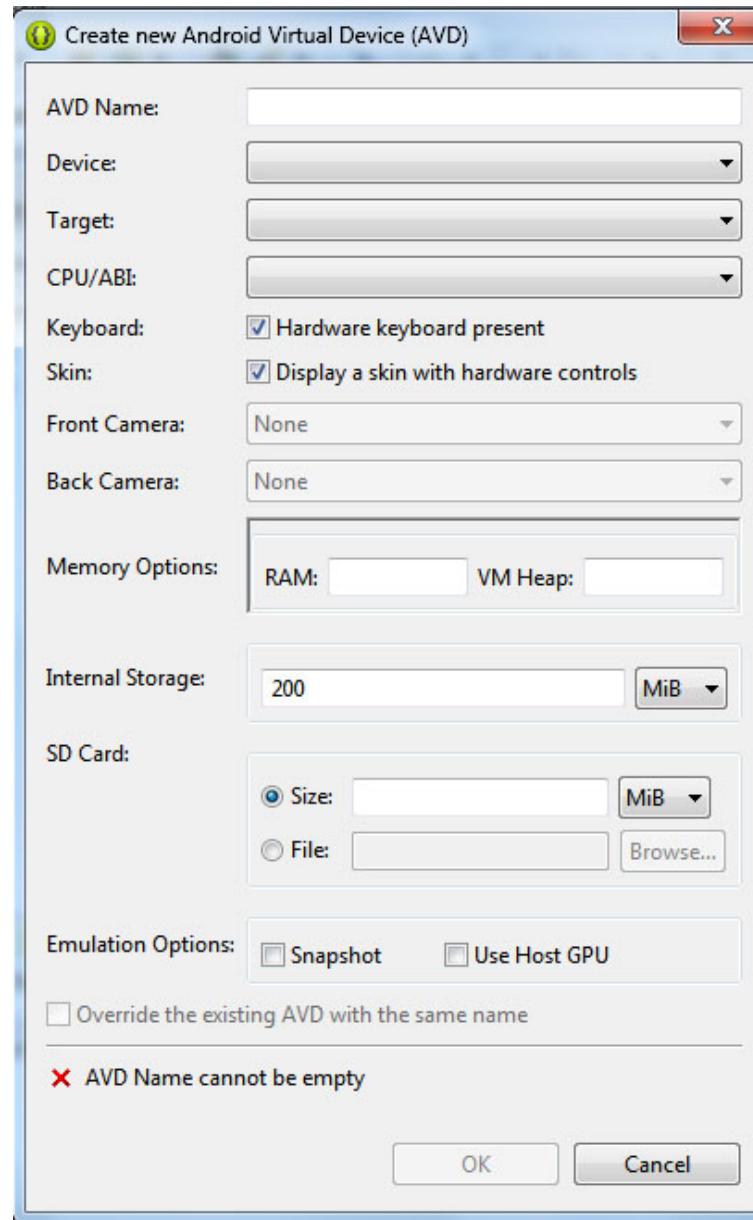
New... Edit... Delete... Repair... Details... Start... Refresh

✓ A valid Android Virtual Device. ⚡ A repairable Android Virtual Device.

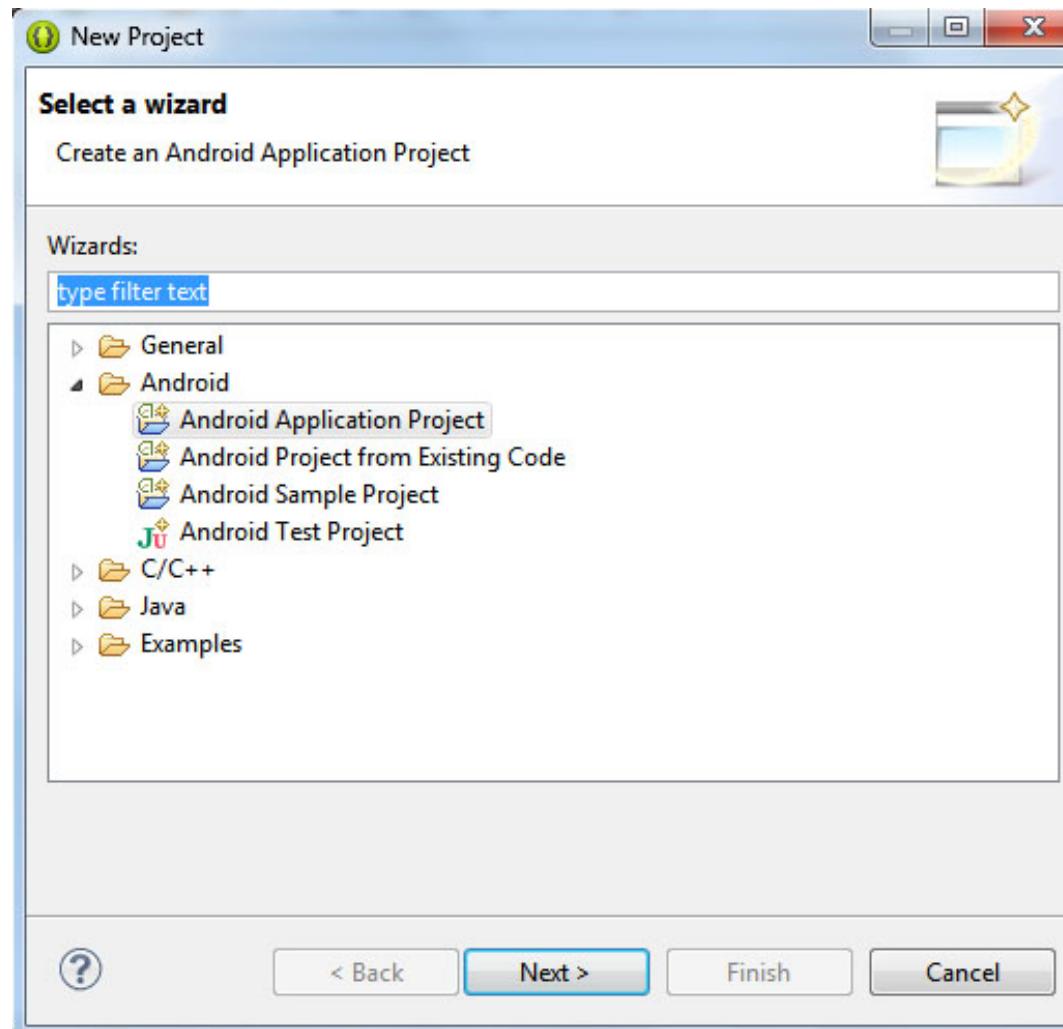
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

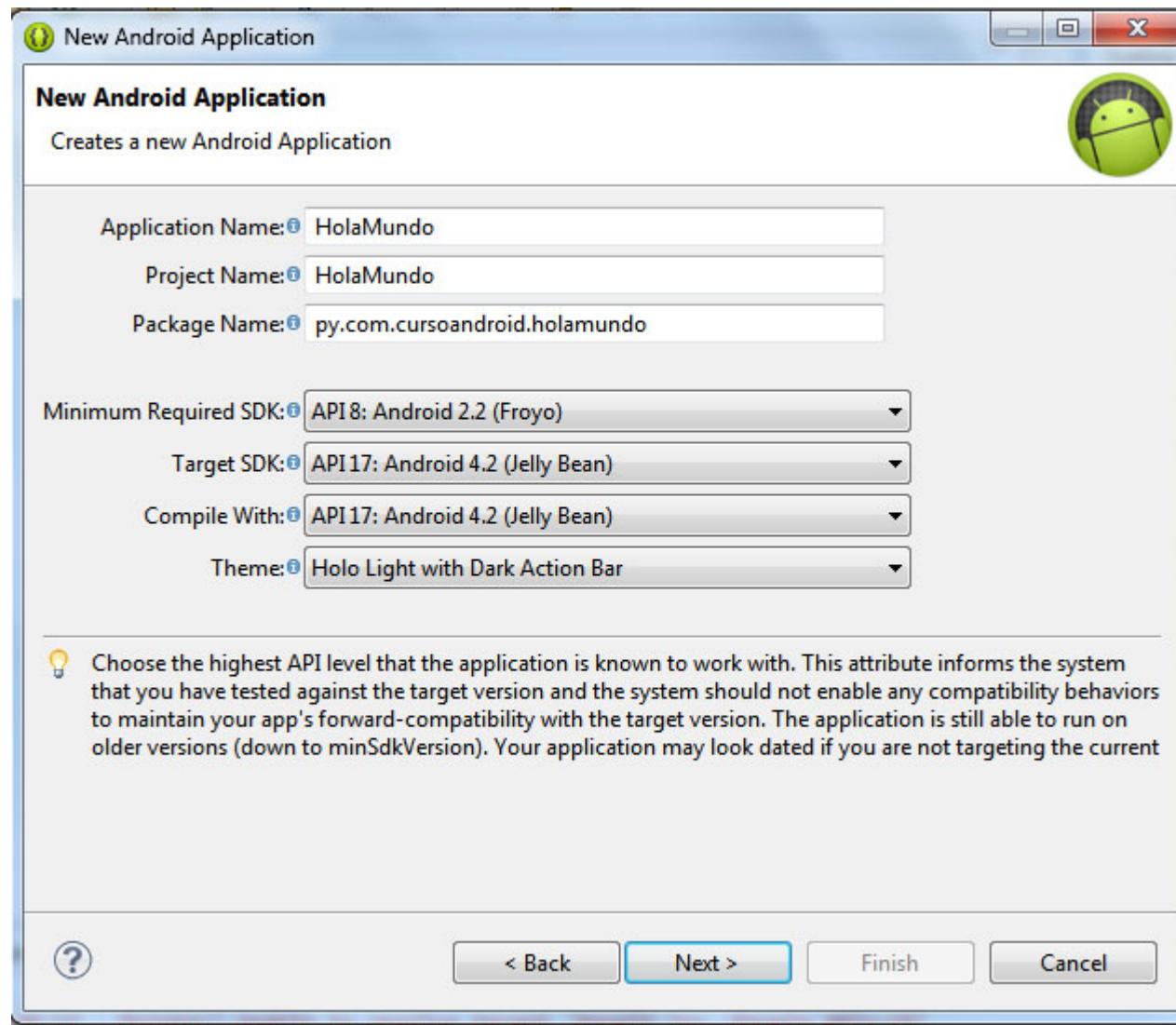


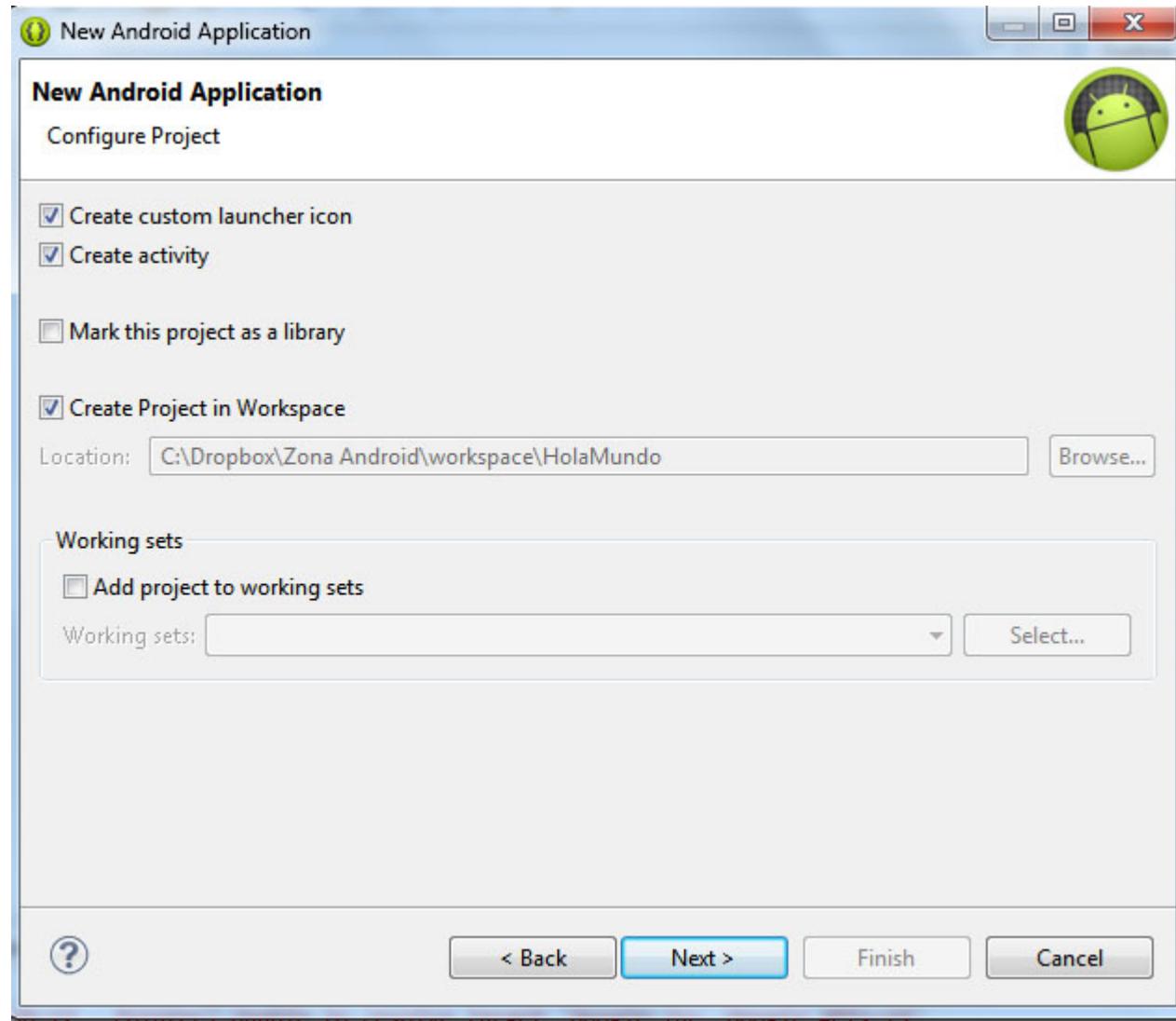
Nuevo AVD



Crear un Nuevo Proyecto (Hola Mundo)







 New Android Application

Configure Launcher Icon

Configure the attributes of the icon set

Foreground: Image Clipart Text

Image File: launcher_icon

Trim Surrounding Blank Space

Additional Padding: 0%

Foreground Scaling: Crop Center

Shape None Square Circle

Background Color:

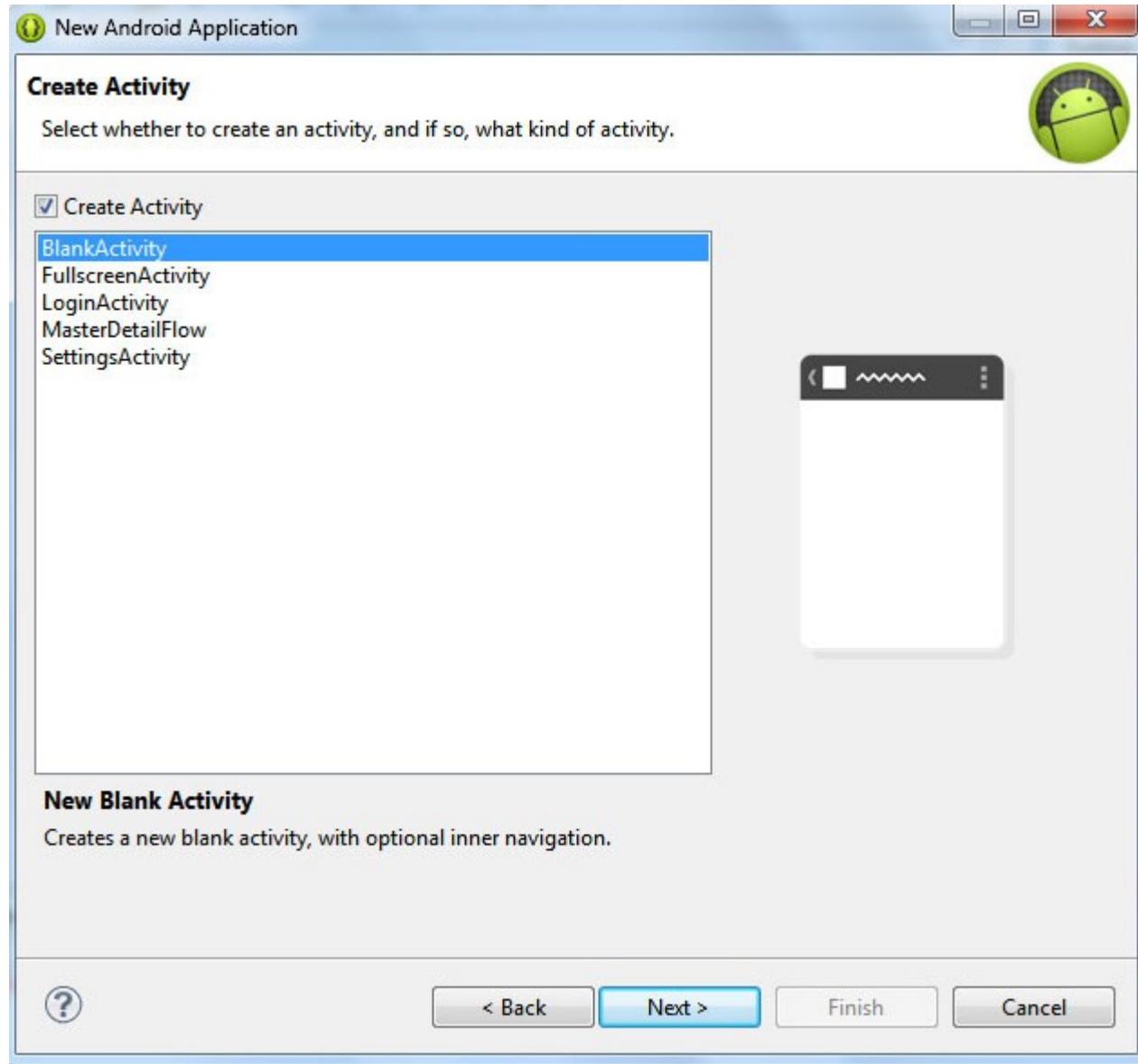
Preview:

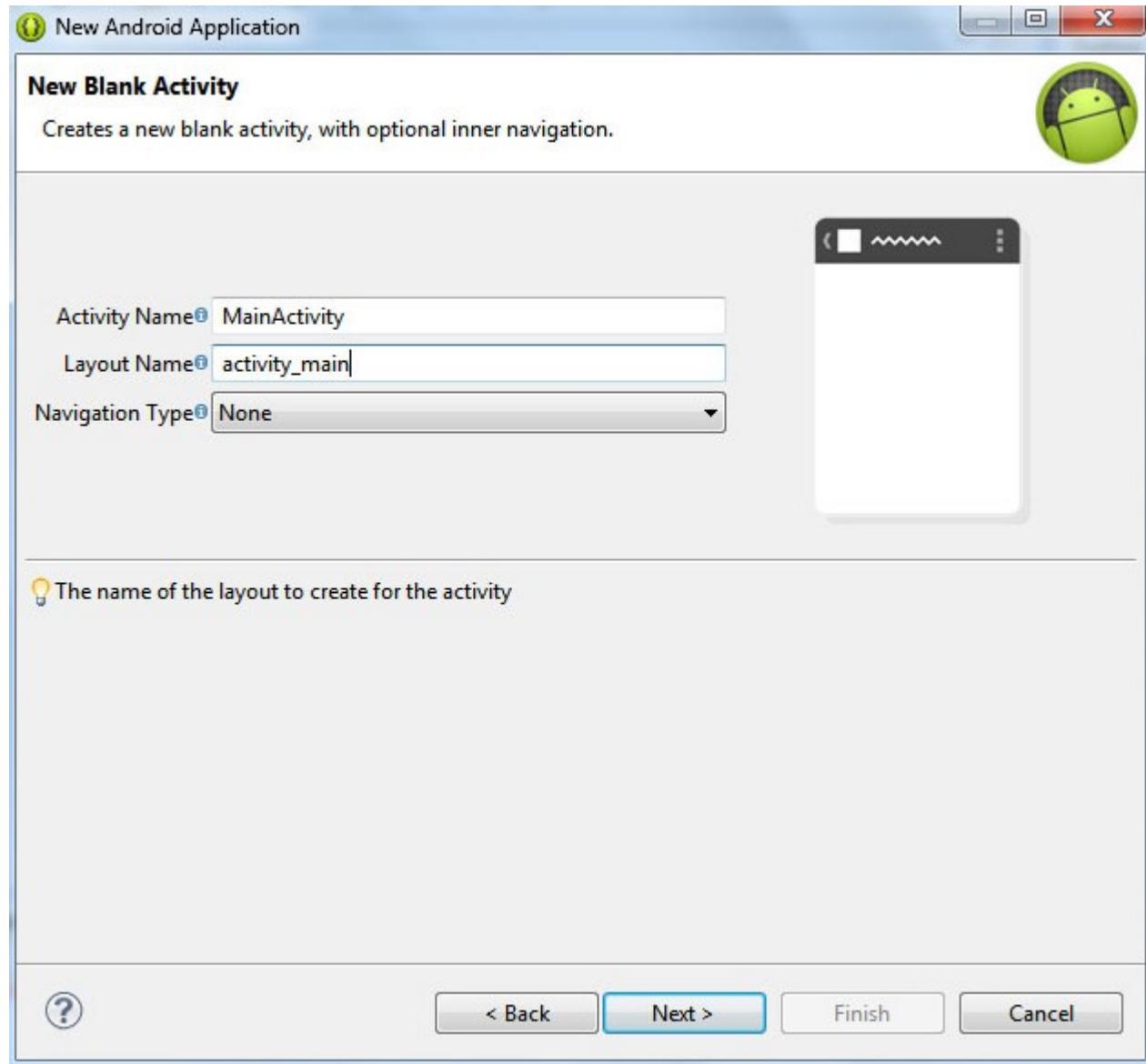
ldpi: 

mdpi:

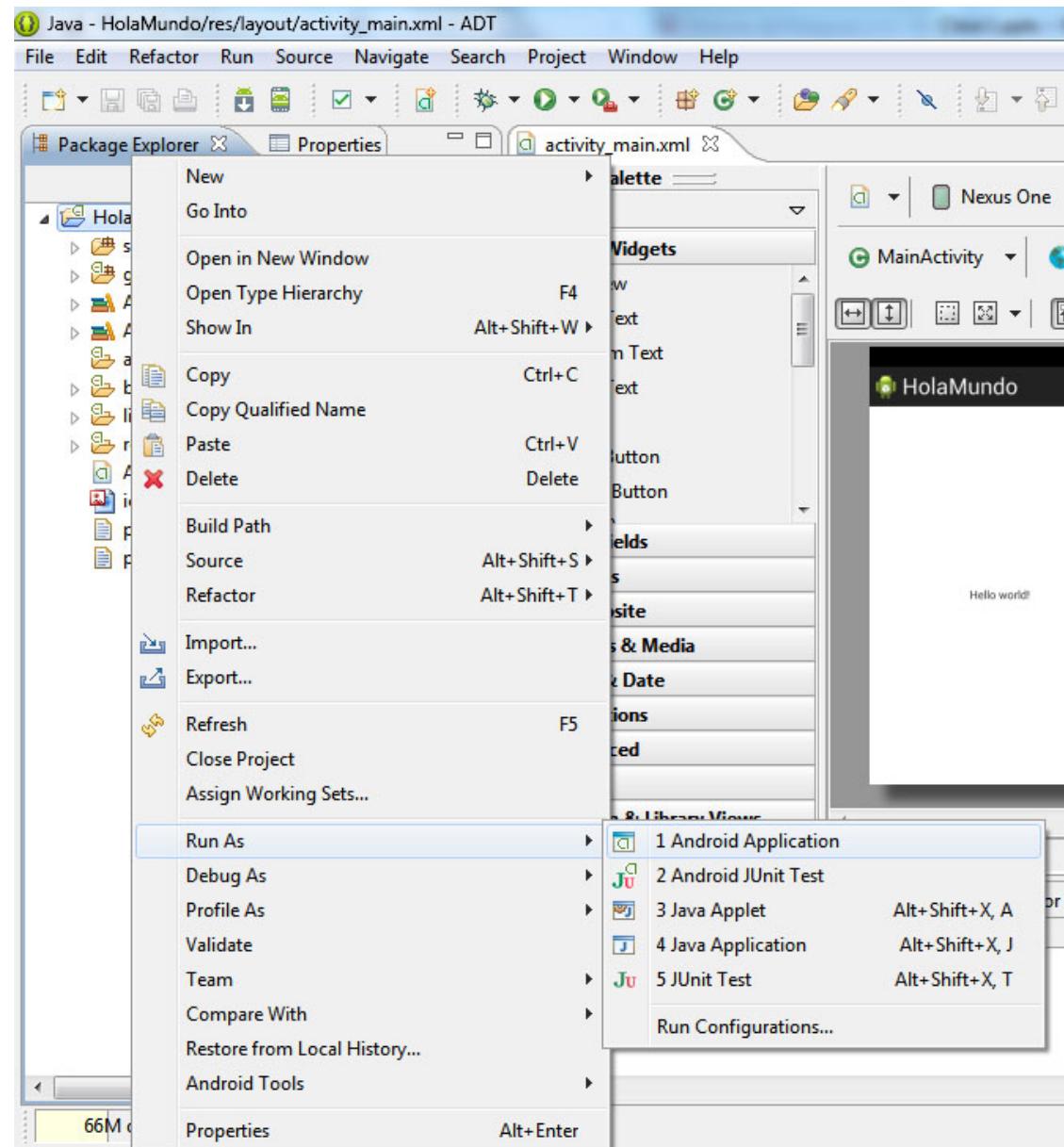
hdpi:

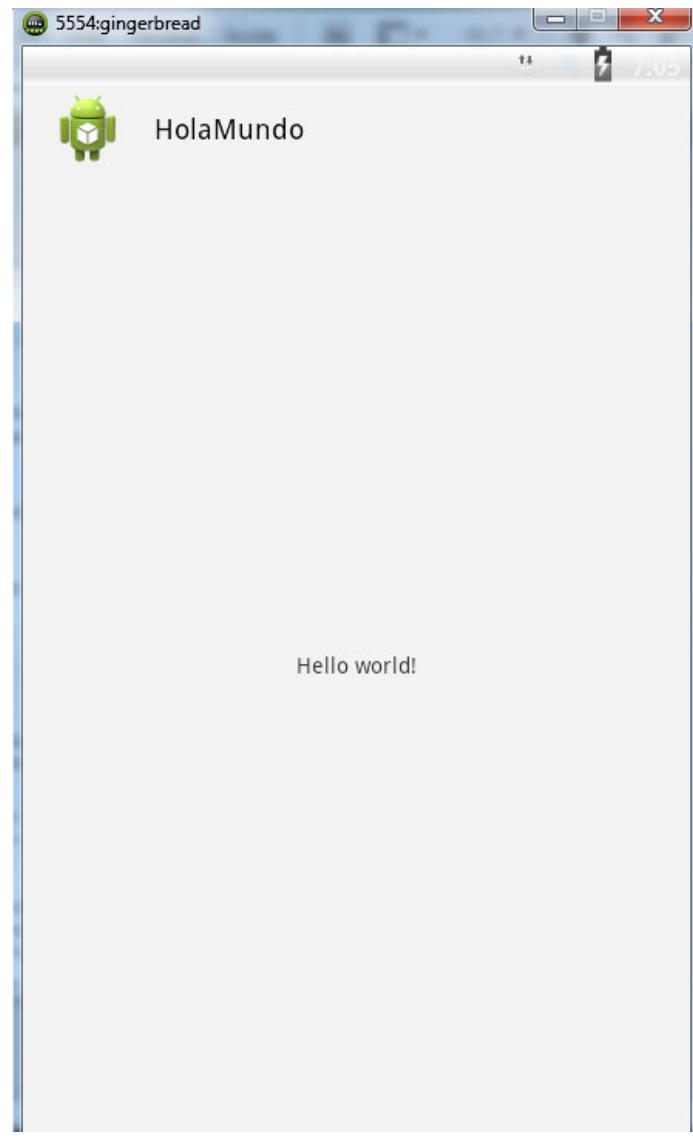
xhdpi: 

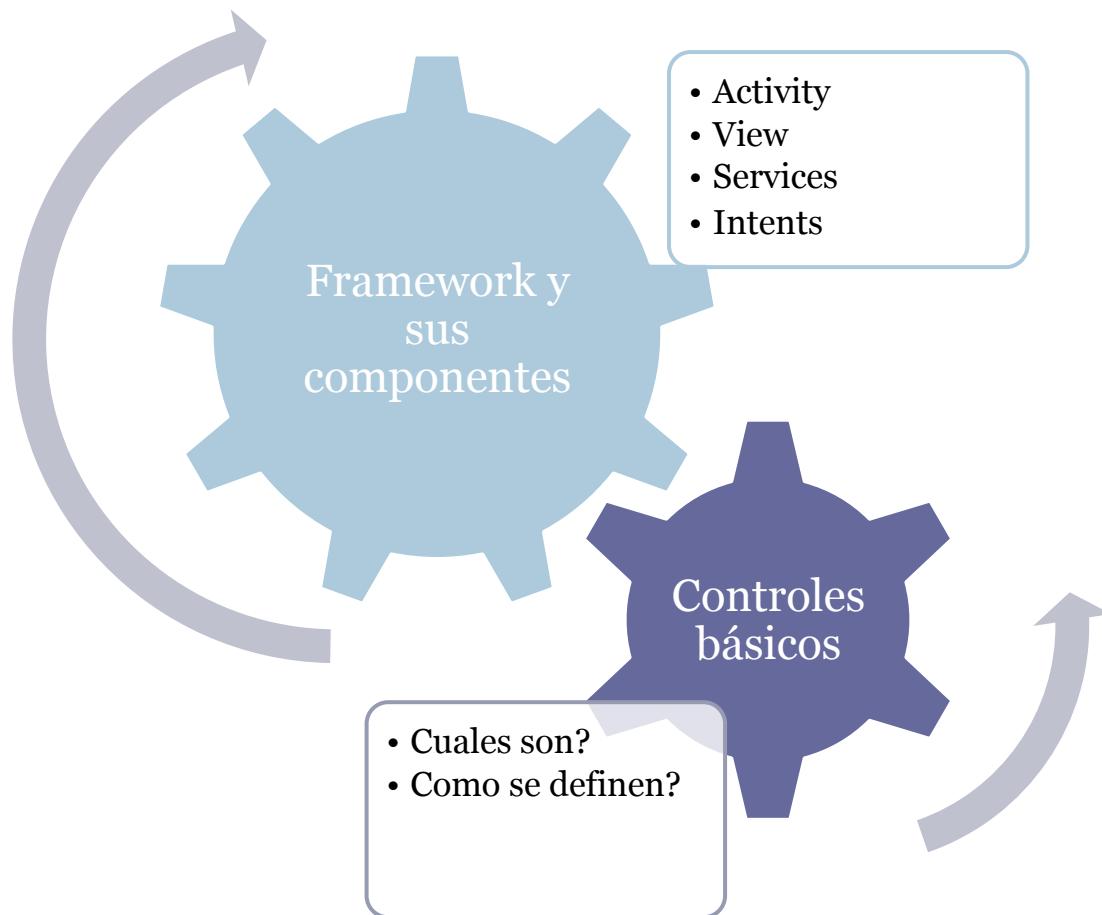




Ejecutar Prueba



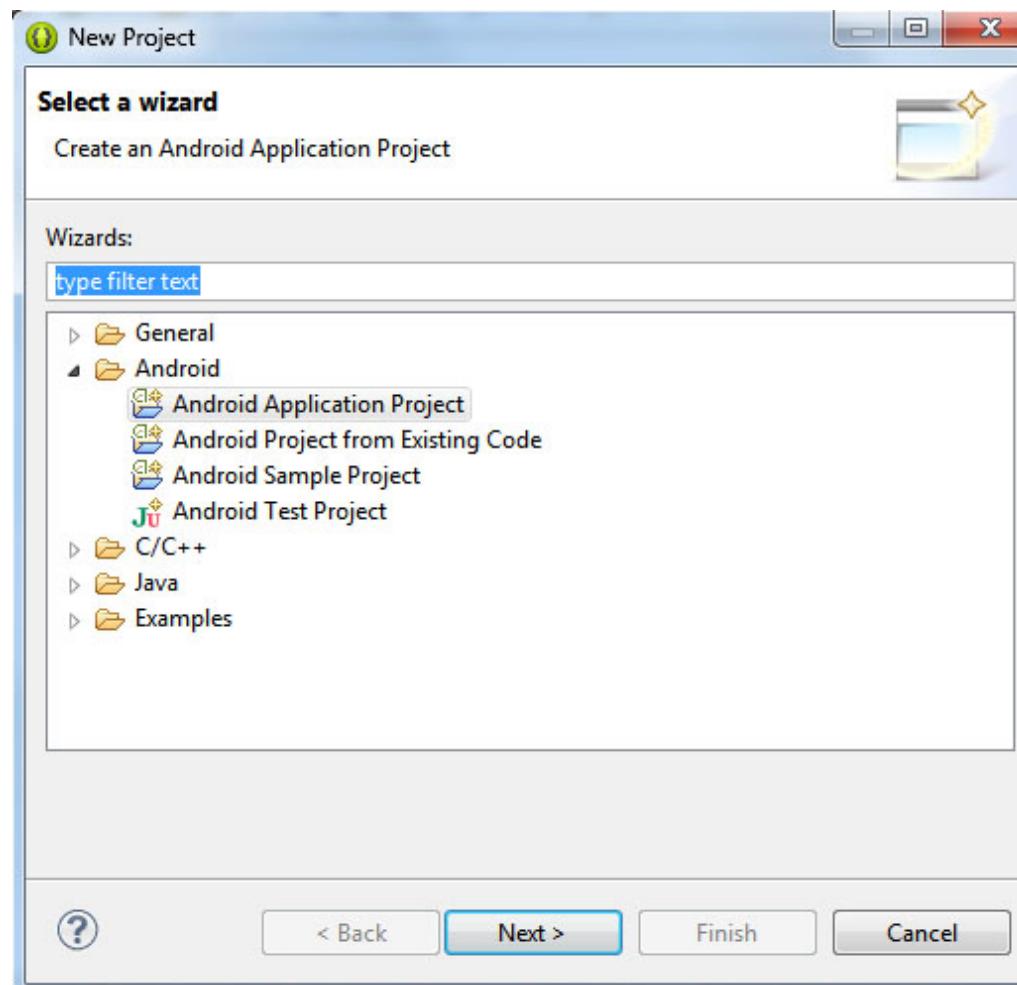




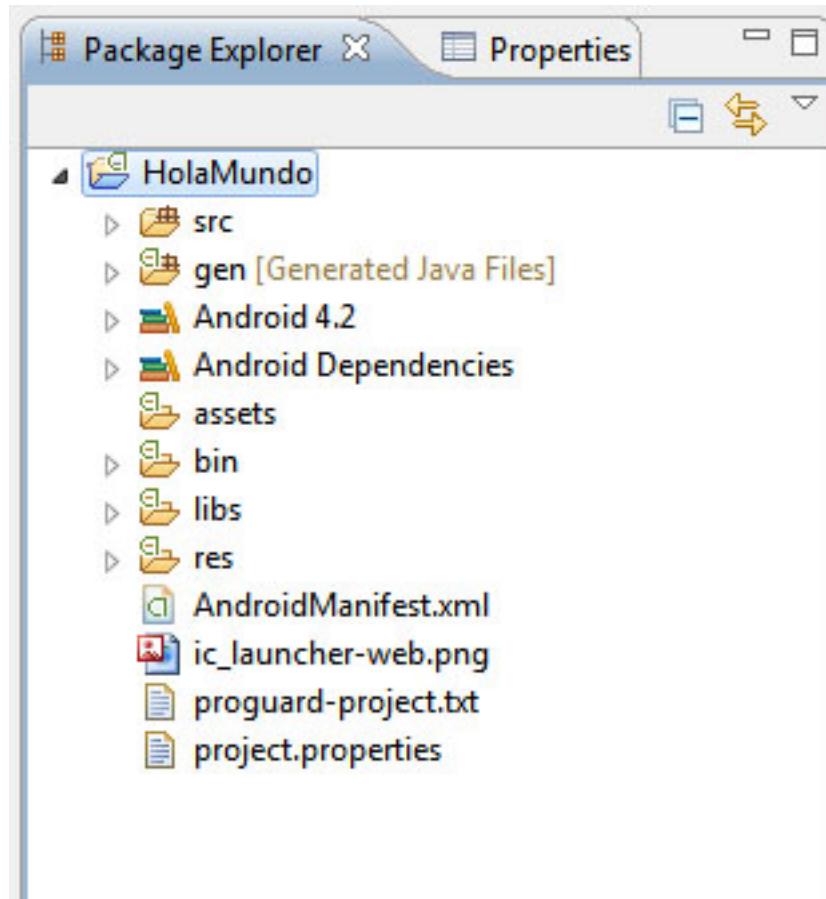
Anatomía de una App típica de Android



Análisis de la estructura del proyecto (Hola Mundo)

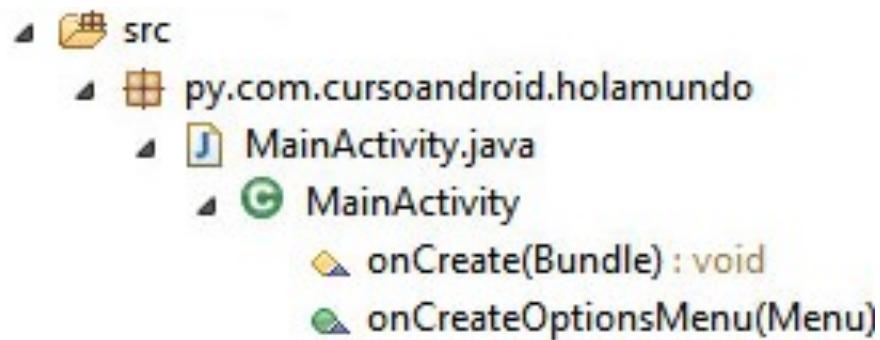


Estructura típica de un proyecto android



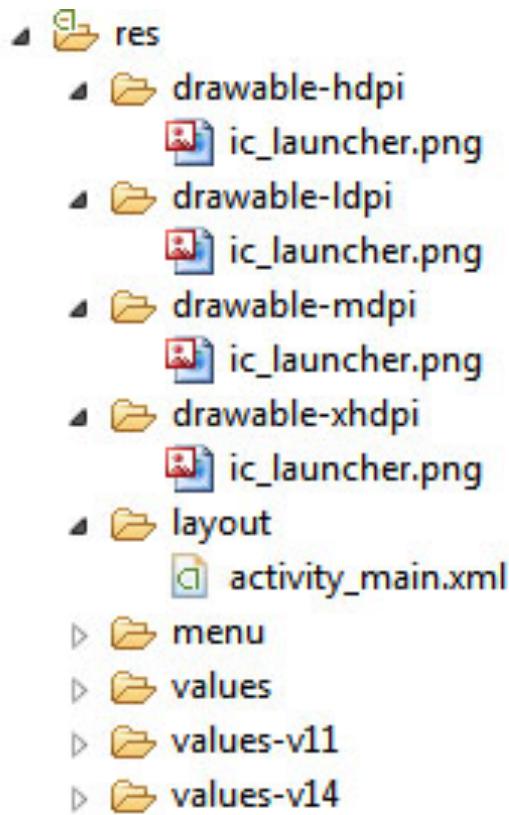
- **Carpeta /src/**

- Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Inicialmente, Eclipse creará por nosotros el código básico de la pantalla (Activity)
- principal de la aplicación, siempre bajo la estructura del paquete java definido.

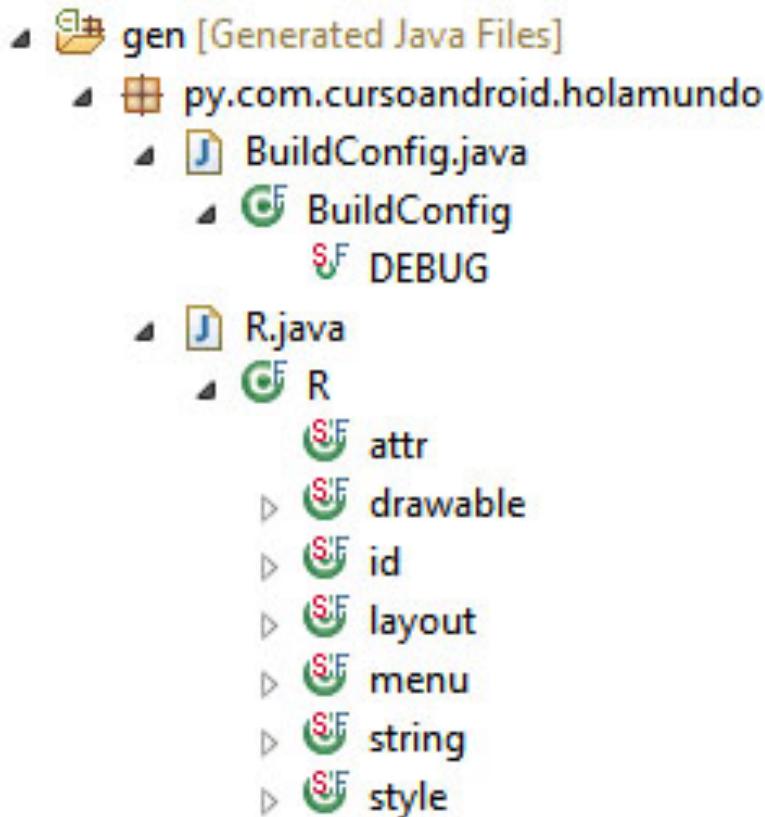


- **Carpetas /src/**

- Contiene todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares,
- etc. Inicialmente, Eclipse creará por nosotros el código básico de la pantalla (Activity)
- principal de la aplicación, siempre bajo la estructura del paquete java definido.



- ***Carpetas /res/***
 - Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc.



- ***Carpetas /gen/***

- Contiene una serie de elementos de código generados automáticamente al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente en java dirigidos al control de los recursos de la aplicación.

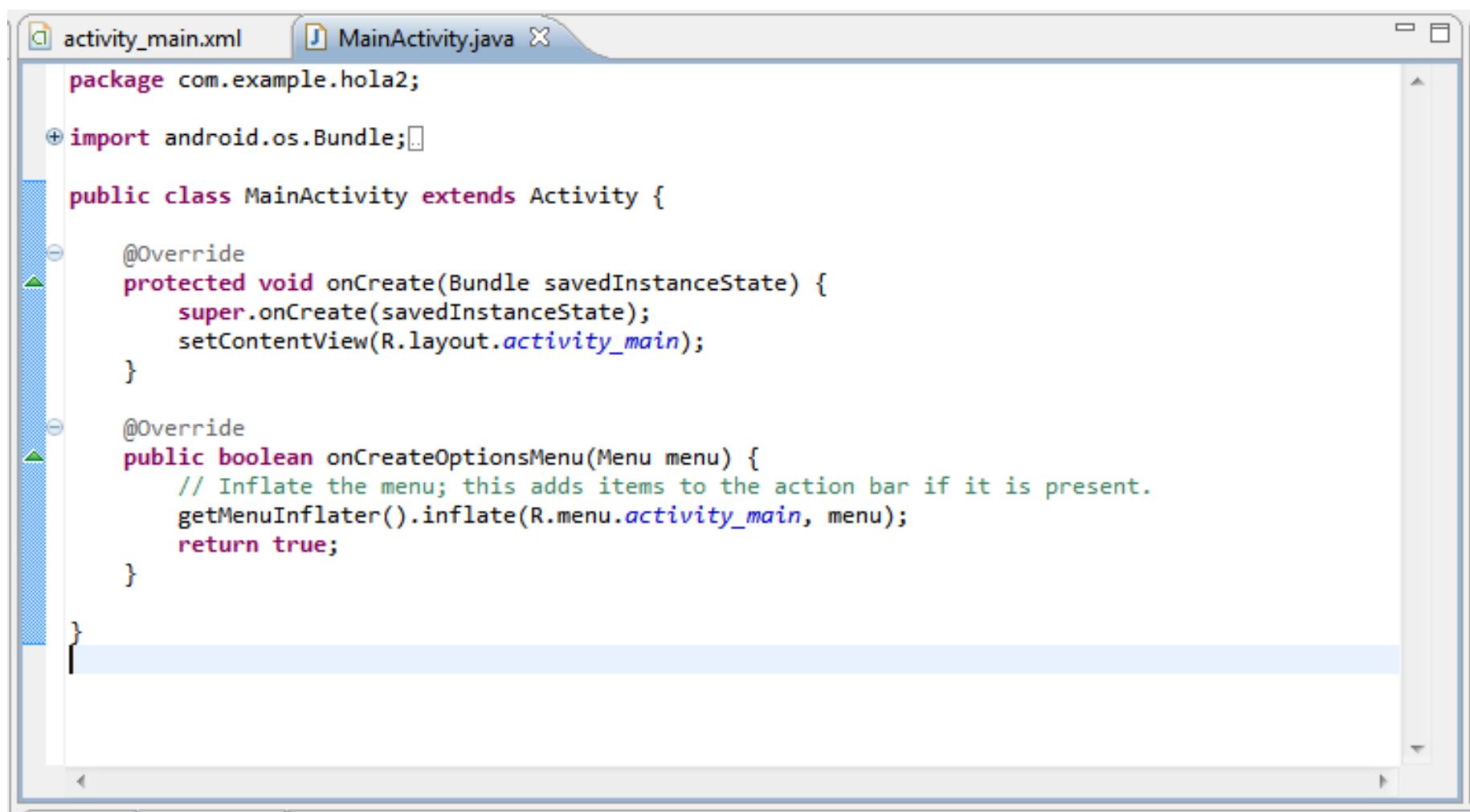
Componentes

- Activity
- View
- Service
- Content Provider
- Broadcast Receiver
- Widget
- Intent

Activity

- Las actividades (*activities*) representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana en cualquier otro lenguaje visual.

Activity



The screenshot shows a Java code editor window with the following details:

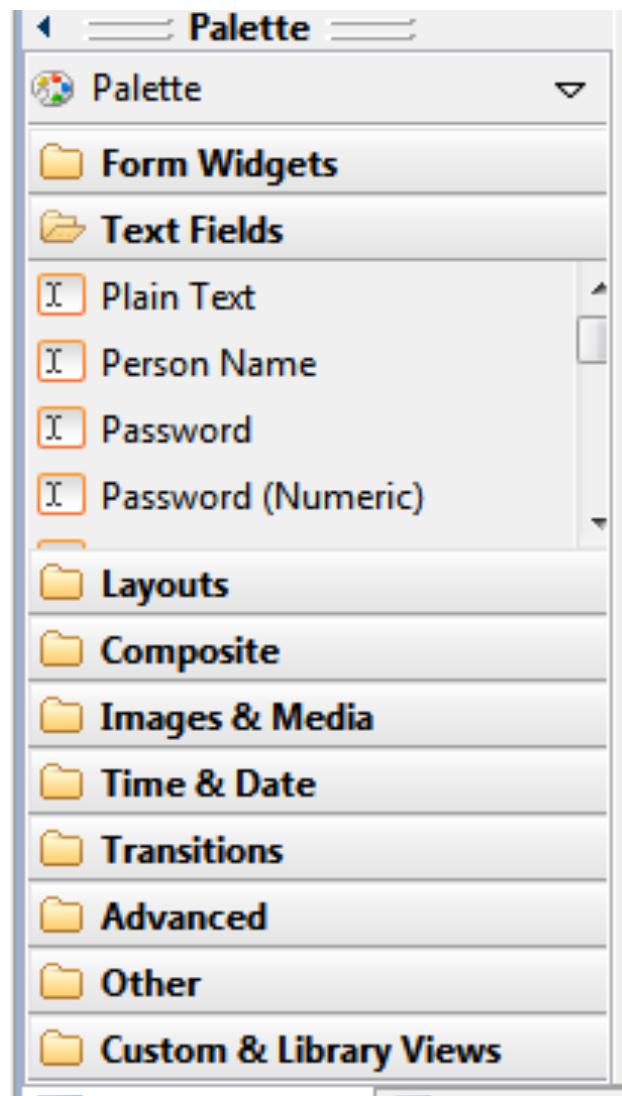
- Title Bar:** The title bar contains two tabs: "activity_main.xml" and "MainActivity.java".
- Code Area:** The main area displays the Java code for the `MainActivity`. The code includes imports for `com.example.hola2`, `android.os.Bundle`, and the `Activity` class. It overrides the `onCreate` and `onCreateOptionsMenu` methods. In the `onCreate` method, it calls `super.onCreate` and sets the content view to `R.layout.activity_main`. In the `onCreateOptionsMenu` method, it inflates the menu from `R.menu.activity_main`.
- Left Margin:** A vertical blue margin on the left side of the code area contains several small green triangle icons, likely indicating code completion or navigation points.

```
package com.example.hola2;
import android.os.Bundle;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

View

- Los objetos view son los componentes básicos con los que se construye la interfaz gráfica de la aplicación, análoga por ejemplo a los controles de Java o .NET. De inicio, Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

View



Service

- Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son exactamente iguales a los servicios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. activities) si se necesita en algún momento la interacción con del usuario.

Content Provider

- Un *content provider* es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los *content provider* que se hayan definido.

Broadcast Receiver

- Un *broadcast receiver* es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: “Batería baja”, “SMS recibido”, “Tarjeta SD insertada”, ...) o por otras aplicaciones (cualquier aplicación puede generar mensajes (*intents*, en terminología Android) broadcast, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

Widgets

- Los *widgets* son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (*home screen*) del dispositivo Android y recibir actualizaciones periódicas.
- Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal.

Intent

- Un *intent* es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones.
- Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

Controles Básicos en Android

- Tenemos 3 formas de utilizar controles y views en android
 - Drag and Drop en el visor gráfico y manipularlos mediante el cuadro de propiedades
 - Declararlos en el fichero xml del layout
 - Declararlos en el archivo Java

Control Button [API]

- Un control de tipo Button es el botón más básico que podemos utilizar. En el ejemplo siguiente definimos un botón con el texto “Púlsame” asignando su propiedad android:text. Además de esta propiedad podríamos utilizar muchas otras como el color de fondo (android:background), estilo de fuente (android:typeface), color de fuente (android:textcolor), tamaño de fuente (android:textSize), etc.

XML

```
<Button android:id="@+id/BtnBoton1"  
        android:text="Púlsame"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content" />
```

Eventos de un botón

- Aunque estos controles pueden lanzar muchos otros eventos, el más común de todos ellos y el que querremos capturar en la mayoría de las ocasiones es el evento `onClick`.
- Para definir la lógica de este evento tendremos que implementarla definiendo un nuevo objeto `View.OnClickListener()` y asociándolo al botón mediante el método `setOnClickListener()`. La forma más habitual de hacer esto es la siguiente:

```
final Button btnBoton1 = (Button) findViewById(R.id.BtnBoton1);
btnBoton1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View argo)
    {
        lblMensaje.setText("Botón 1 pulsado!");
    }
});
```

Control TextView [API]

- El control TextView se utiliza para mostrar un determinado texto al usuario. El texto del control se establece mediante la propiedad android:text. A parte de esta propiedad, las que establecen el formato del texto mostrado son las siguientes: android:background (color de fondo), android:textColor (color del texto), android:textSize (tamaño de la fuente) y android:typeface (estilo del texto: negrita, cursiva, ...).

```
<TextView android:id="@+id/LblEtiqueta"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:text="Escribe algo:"  
        android:background="#AA44FF"  
        android:typeface="monospace" />
```

- De igual forma, desde nuestro código podremos recuperar y establecer este texto mediante los métodos `getText()` y `setText(nuevoTexto)` respectivamente:

```
final EditText txtTexto = (EditText)findViewById(R.id.TxtTexto);
String texto = txtTexto.getText().toString();
txtTexto.setText("Chau mundo!");
```



Terminamos por Hoy